phprel – be as fast as you can imagine

SC P.B.L. WEB LABS SRL
CUI RO23495308
www.phprel.ro
office@phprel.ro
support@phprel.ro
0747 458 678 / 0747 227 437

Termeni si conditii

Informatiile din acest document se pot schimba oricand fara o notificare prealabila si nu constituie un angajament din partea SC P.B.L. WEB LABS SRL. Desi toate eforturile au fost facute pentru a asigura corectitudinea si completitudinea prezentului document, SC P.B.L. WEB LABS SRL nu poate fi facuta responsabila pentru nici o omisiune sau greseala. Nici o parte a acestui manual nu poate fi reprodusa sau transmisa sub nici o forma, electronic sau mecanic, inclusiv prin fotografiere, transcriere, inregistrare sau orice alt sistem de stocare a informatiilor, cu exceptia cazurilor in care acest lucru este realizat pentru a facilita utilizarea framework-ului phprel de catre persoana care a cumparat sau testeaza respectivul produs. Toate marcile folosite in acest document apartin posesorilor lor de drept. Toate informatiile, textele si imaginile apartin companiei SC P.B.L. WEB LABS SRL.

Versiune document: HLP 1.0.1 / 12.06.2008

Introducere

Documentul de fata isi propune sa va familiarizeze cu conceptele si metodele de lucru specifice framework-ului phprel, un framework bazat pe o idee noua care va ajuta sa scrieti aplicatii web ergonomice, sigure si performante intr-o maniera simpla si rapida. Obiectivul echipei phprel este sa va puna la dispozitie uneltele necesare implementarii unor site-uri moderne, cel putin "web 2.0", intr-un timp deseori uimitor de scurt, site-uri sustinute de tehnologii puternice precum A.J.A.X. sau "content caching". O atentie deosebita a fost acordata comunicarii dintre dumneavoastra si framework-ul ce interconecteaza uneltele create de noi, rezultatul fiind un "asistent virtual" care preia din sarcinile dumneavoastra si este atent la indicatiile dumneavoastra date prin intermediul a cateva linii "naturale" de cod. Sentimentul creat este unul de concizie si rapiditate, dublat de siguranta data usde un control total asupra oricarui proces: fara stres, fara impedimente, duceti la bun sfarsit un proiect care ar putea parea lung si obositor intr-un timp de cel putin trei ori mai scurt. Cu phprel, castigati timp la fiecare apasare a unei taste, fiindca asistentul phprel interpreteaza si foloseste orice informatie scrisa de dumneavoastra in sursele aplicatiei.

Documentul prezent a fost structurat in trei parti:

- primele doua sectiuni constituie un mini-exemplu practic in care veti pune la incercare afirmatiile precedente si va veti familiariza intuitiv cu cateva concepte centrale din phprel. Obiectivul principal al celor doua sectiuni este sa va obisnuiasca cu metodele de lucru caracteristice, nu sa va impresioneze: exemplele date nu sunt cele mai rapide sau mai importante, nu au fost alese special, si nu sunt cazuri particulare, iar phprel nu se limiteaza doar la crearea unor aplicatii care contin elementele prezentate in mini-exemplu. Veti folosi phprel in crearea oricarei aplicatii, indiferent de natura, complexitatea sau destinatia ei.
- urmatoarele trei sectiuni detaliaza facilitatile oferite de phprel, precum si structura si flexibilitatea sa. Sunt oferite informatii amanuntite, in maniera unui document de prezentare si "ajutor".
- sectiunea "Index" ofera o descriere exhaustiva dar succinta a tuturor entitatilor prezente in phprel, ordonate pe categorii si subcategorii, apoi alfabetic.

In sectiunile urmatoare, au fost folosite cateva conventii de denumire sau de scriere:

- un cuvant sau sintagma scrisa intre ghilimele, pe langa utilizarile specifice limbii romane, poate reprezenta un termen tehnic si/ sau un termen englezesc folosit ca atare, o valoare concreta (ca sir de caractere) atribuita unei entitati sau indica folosirea termenului ca denominativ.
- un cuvant, sintagma sau sir de caractere scris cu culoarea verde reprezinta dupa caz o entitate phprel noua sau un parametru/ valoare obligatorie.
- un cuvant, sintagma sau sir de caractere scris intre paranteze "( )" reprezinta o valoare care va fi inlocuita practic cu o alta valoare, conform indicatiilor date. Parantezele nu vor aparea in valoarea respectiva cu exceptia cazurilor in care este vorba de parantezele unei functii sau in cazurile in care se precizeaza explicit ca vor fi folosite paranteze. Constructia "(...)" desemneaza orice sir de caractere, echivaland cu "orice" sau "orice valoare".
- valorile, entitatile sau parametrii scrisi intre paranteze drepte "[ ]" nu sunt obligatorii si pot lipsi
- un cuvant, sintagma, parametru sau sir de caractere scris cu culoarea gri nu este obligatoriu si poate lipsi, sau nu este important pentru context

- anumite informatii relevante au fost scrise intr-un paragraf separat inceput cu "**Important**"

1. phprel la prima vedere

Aceasta sectiune si sectiunea urmatoare au rolul de a va familiariza cu phprel si cateva concepte esentiale, printr-un exemplu construit progresiv in fiecare subsectiune. Exemplul presupune ca aveti deja instalat phprel, sau ca aveti acces la un server pe care dispuneti de phprel. Este de preferat sa parcurgeti intregul exemplu si de fiecare data cand faceti o pauza si reveniti mai tarziu, sa pastrati sursele pe care le-ati scris pana in acel moment. Solutiile vor fi prezentate intuitiv, cu minimum de detalii necesare pentru a intelege ideile centrale, idei care vor putea fi apoi extinse si adaptate la situatii reale urmarind sectiunile ulterioare.

1.1. Concept

phprel este un framework unic, construit in jurul unui nou concept si menit sa accelereze cu adevarat procesul de dezvoltare web. O accelerare teoretica, rezultata din structurarea codului si din facilitarea mentenantei este inerenta oricarui framework si este discutabila. Pentru phprel, acest tip de accelerare nu constituie un obiectiv ci doar un efect secundar. Adevarata viteza data de phprel este una

reala, masurabila in timp de dezvoltare (sau mai degraba in cit de scurt este intervalul de timp): phprel preia concret din sarcinile programatorului, rezolvand situatii tipice consumatoare de timp, si lasandu-i programatorului libertatea de a se ocupa de problemele cu adevarat mari ale proiectului.

Nucleul phprel va asista in realizarea anumitor task-uri, in unele cazuri fara nici o interventie din partea dumneavoastra, alteori fiind nevoie doar de cateva indicatii – si totul fara a compromite in nici un fel controlul asupra aplicatiei, flexibilitatea sau design-ul. Asistenta este asigurata prin numeroasele observatii pe care phprel reuseste sa le faca in baza liniilor de cod scrise: conceptul central este acela ca totul este scris cu un anumit motiv, entitatile aplicatiei fiind interconectate la nivel logic dar si la nivel semantic, un "cuvant" folosit intr-un loc ca nume de variabila, de pagina, etc. nu este doar dintr-o coincidenta identic sau similar cu denumirea unei tabel, a unei alte variabile, etc. De asemenea, phprel este atent sa recunoasca tipare foarte des intalnite in programarea web, si atunci cand va aflati intr-o situatie cunoscuta sa ofere cit de mult ajutor este posibil, respectand totodata indicatiile dumneavoastra si pastrand flexibilitatea. Parcurgand exemplele din acest manual, veti descoperi foarte repede cum anume se aplica practic aceste concepte pentru a asigura o dezvoltare de cel putin trei ori mai rapida.

Trebuie stiut ca phprel nu a fost construit ca un framework, rezultand apoi o unealta de dezvoltare web foarte rapida, dimpotriva, a fost creat ca o unealta de dezvoltare web foarte rapida si integrat apoi ca framework. Aceasta observatie este importanta pentru a sublinia faptul ca phprel este un framework din lipsa unei alte categorii in care poate fii incadrat, dar este esential diferit de framework-urile existente deja: phprel este si un framework, dar diferenta majora o constituie abilitatea de a identifica si prelua din sarcinile programatorului, actionand din acest punct de vedere ca un asistent artificial pentru dezvoltarea aplicatiilor web.

In plus, phprel nu respecta tendinta actuala de design a pietei de framework-uri, nefiind orientat pe obiecte si nici bazat pe arhitectura "model-view-controller" (MVC). Exista si clase in phprel, la fel cum exista si o componenta de prezentare (care ar putea fi intr-un sens larg asimilata componentei "view"), o componenta de calcul (asa numitul "business logic", dar care cu greu ar putea fi asimilat unui controller, pentru ca in phprel conceptul central este "pagina", rezultand un business logic separat pentru fiecare pagina, care eventual foloseste elemente comune cu alte pagini – aceasta impartire se dovedeste foarte eficienta ca timp de procesare al paginii, pentru ca un controller complicat care ar acoperi multiple pagini ar putea ajunge la mii de linii de cod, care se includ destul de greu) si o componenta care descrie particularitati de acces la baza de date (asimilabila din nou intr-un sens larg componentei "model", dar cu greu, deoarece in arhitectura MVC modelul chiar intermediaza accesul la baza de date, in timp ce in phprel aceasta componenta este o simpla "descriere", nu un cod responsabil pentru acces, asa cum vom vedea), dar diferentele fata de arhitecturile OOP si MVC consacrate sunt majore. Explicatia este una simpla: phprel este orientat-pe-productivitate, mai degraba decat orientat-pe-obiecte, arhitectura fiind una adaptata pentru maximum de viteza in dezvoltare. Se promoveaza o arhitectura adaptata ea insasi la conceptele aflate in spatele web-ului, si nu invers: o arhitectura care ar incerca sa adapteze conceptele din spatele web-ului la tipare create artificial. Exista, bineinteles, o paradigma de programare in phprel: reflection-oriented programming, o solutie care a permis un stil de comunicare cu framework-ul extrem de concis si eficient. De asemenea, phprel nu poate fi incadrat in una din categoriile "push-based" (executie "dinspre cod spre design") sau "pull-based" (executie "dinspre design spre cod"), ci un hibrid inovativ al celor doua abordari.

Cele mai des intalnite elemente sau comportamente ale paginilor web au fost incluse in phprel pentru a oferi un suport cat mai complet in orice fel de aplicatie, de la solutii de tip Content Manangement System (CMS), Client Relationship Management (CRM) la magazine online (eCommerce) si Enterprise

Resource Planning (ERP). Tehnologii precum url rewrite, A.J.A.X., caching, multilanguage au fost implementate in variante inovative pentru a permite site-uri optimizate SE(O), o experienta placuta pentru utilizatorii finali, un timp de raspuns foarte bun inclusiv in conditii de trafic intens si dezvoltarea rapida inclusiv a site-urilor in multiple limbi. In sprijinul aplicatiilor complexe, phprel ofera suport trasparent pentru multiple baze de date si restrictionarea transparenta a accesului la anumite pagini sau informatii folosind grupuri de utilizatori. In plus, securitatea este standard in phprel: url-urile cu risc mare sunt validate inainte de executia paginii, formularele sunt revalidate automat in php inainte de procesarea datelor, query-urile sunt asigurate impotriva atacurilor de tip sql injection iar sesiunea este protejata impotriva atacurilor de tip session hijacking. Nu in ultimul rand, transmiterea datelor prin intermediul cererilor de tip A.J.A.X. este criptata. Toate acestea sunt oferite in plus fata de ajutorul semnificativ acordat la construirea formularelor, listelor, includerea diverselor elemente de pagina, scrierea codului javascript, construirea rapida a query-urilor, implementarea cosului de cumparaturi si preluarea comenzilor, precum si alte sarcini web.

phprel nu este un wizard de aplicatii web sau un Content Management System, nu contine modele sau tipare predefinite de site-uri, sau parti de cod gata scrise pentru a fi folosite. Orice aplicatie scrisa folosind phprel va trebui implementata de la zero, insa va fi scrisa intr-un timp mult mai scurt datorita ajutorului oferit de phprel pe parcursul dezvoltarii.

phprel isi propune sa fie primul pas in redefinirea termenilor de dezvoltare (web) si rapiditate (a dezvoltarii). Echipa phprel depune continuu efort pentru asigurarea unor mijloace inovative de scriere a aplicatiilor web si va promite realizarea aplicatiilor web intr-un timp de minim trei ori mai scurt, cu mai putin stres si intr-o atmosfera mai vesela, precum si un rezultat de care veti fi mandru.

1.2. Instalare

Phprel este un software proprietar comercializat de S.C. P.B.L. WEB LABS S.R.L. Romania, protejat de legea dreptului de autor si utilizabil doar in termenii unei anumite licente. Pentru a afla mai multe despre phprel sau alte servicii ale P.B.L. WEB LABS, sau pentru achizitionarea (ori, dupa caz, solicitarea gratuita a) unei licente phprel vizitati site-ul www.phprel.ro. Pentru a proteja produsul si a asigura respectarea licentei precum si pentru a proteja proprietatea intelectuala a autorului, phprel este distribuit pentru folosire intr-o varianta criptata si poate fi folosit doar dupa activarea produsului pe site-ul oficial.

Criptarea produsului este realizata folosind un software specializat cu o larga raspandire, Zend Guard, iar rularea pe un server web a surselor criptate presupune instalarea unei extensii php corespunzatoare, Zend Optimizer, disponibila gratuit la http://www.zend.com/en/products/guard/downloads. Nucleul phprel nu va putea fi utilizat fara Zend Optimizer iar P.B.L. WEB LABS S.R.L. nu are obligatia instalarii extensiei pe serverul web care gazduieste aplicatia dumneavoastra, nici punerea la dispozitie a unei solutii de folosire a phprel fara Zend Optimizer.

Trebuie stiut ca majoritatea serverelor de gazduire au instalate extensia Zend Optimizer, iar instalarea este un proces usor care nu dureaza mai mult de cateva minute si nu intra in conflict cu nici o alta aplicatie care este deja instalata pe respectivul server.

Dupa ce ati instalat Zend Optimizer sau v-ati asigurat ca aceasta extensie este deja instalata, se poate instala usor phprel. Fiind o aplicatie php destinata folosirii intr-un mediu web, phprel nu este impachetat intr-un installer, procesul de instalare rezumandu-se la patru pasi simpli:

1. Descarcati phprel de pe site-ul oficial, dezarhivati si copiati continutul corespunzator versiunii php (directorul "distribution") intr-un director prestabilit de pe server in care veti tine framework-ul
2. Copiati in radacina aplicatiei dumneavoastra continutul directorului "common/" si suprapuneti apoi continutul directorului "versions/front/" sau "versions/back/", in functie de varianta de phprel dorita (pentru mai multe detalii, consultati manualul la sectiunea 3. phprel in detaliu, subsectiunea 3.1. Nucleul phprel). Se poate deja incarca pagina web in browser.
   Directorul radacina al noii aplicatii ar trebui sa contina acum:
   - cfgs/
   - uploads/
   - web/
   - webdevel/
   - .htaccess
   - index.php
   - functions.(front/ back).php

   Pentru a va familiariza cu phprel, incarcati mai intai versiunea "front".
3. In cazul in care serverul este unul de dezvoltare (pe care veti realiza mai multe aplicatii web folosind phprel), modificati fisierul cfgs/smartlocate.inc.php pentru a indica catre directorul distributiei phprel, altfel copiati directoarele "core/" si "include/" in radacina aplicatiei dumneavoastra si editati fisierul cfgs/smartlocate.inc.php pentru a indica catre directorul curent (aceasta operatie presupune setarea valorii variabilei $Tcfg_distributionDirectory la sirul vid "").
4. Un refresh al paginii web va va indica un cod de activare al copiei phprel. In cazul in care ati copiat varianta "front", codul nu va aparea pe ecran, fiind vizibil doar in sursa html a paginii. Nu mai ramane decit sa introduceti codul impreuna cu numele de domeniu sau adresa ip la care este accesibila aplicatia pe site-ul oficial phprel, sectiunea clienti (http://www.phprel.ro/clienti-corporate/), si sa descarcati licenta in directorul "core/".

Dupa instalare, interfata Web Assistant va va ghida prin procedura de conectare a framework-ului la baza de date, va va cere permisiunea sa creeze tabelele interne phprel necesare aplicatiei dumneavoastra si va va invita sa creati un cont de administrare prin intermediul caruia veti avea acces la Web Assistant – o interfata web de mentenanta si dezvoltare oferita impreuna cu phprel (detalii in sectiunea 2. Elemente de baza, subsectiunea 2.3. Web Assistant – interfata web de management a aplicatiei).
Se poate instala phprel atat pe sisteme de operare Linux cat si Windows, pe servere http care indeplinesc simultan urmatoarele conditii:

- au instalate php, versiunea 4 >= 4.3 sau 5 (sau, conform specificatiilor cunoscute pana in prezent, php 6)
- ofera suport pentru url rewrite
- ofera suport pentru extensii php si extensia Zend Optimizer este instalata
- se pot conecta la un server de baze de date MySQL versiunea 3, 4 sau 5.

Configuratia recomandata este un server Linux, Apache 2, php 5.2, MySQL 4 sau 5.

1.3. Pagina web

Directorul web/ va contine toate sursele php, html, css, imaginile si fisierele aditionale care vor alcatui aplicatia dumneavoastra. Localizarea lor intr-un singur subdirector va permite sa efectuati in orice moment o sincronizare rapida si completa intre locatia temporara de dezvoltare si serverul live. Pentru a crea o noua pagina, trebuie creata componenta de prezentare ("design"-ul propriu-zis al paginii) si componenta de calcul (sursa php care calculeaza toate informatiile necesare "design"-ului). In phprel, o aplicatie web este formata din mai multe pagini, o pagina fiind specificata printr-un parametru GET "page" (sau, dupa caz, prin echivalentul lui folosind url rewrite – pentru detalii consultati sectiunea 2. Elemente de baza, subsectiunea 2.2. Url rewrite).

Unei pagini ii corespunde un php cu acelasi nume in directorul "web/pages/", si un html in directorul "web/templates/". Pagina care se incarca implicit pentru varianta "front" este "home" iar pentru varianta "back" este "main". Incercam mai intai modificarea paginii web implicite:

- modificam web/templates/home.html, introducand textul "O pagina phprel simpla" si reincarcam aplicatia in browser.
- la pasul doi, se poate de exemplu schimba clasa css a textului: includem textul introdus in tag-ul …
- **Important**: clasa css s-a aplicat pentru ca a fost incarcat implicit fisierul "web/css/default.css". In acest fisier se poate defini si/ sau modifica stilul css al aplicatiei. Pentru a adauga mai multe css-uri, precum si pentru a modifica "layout"-ul general al aplicatiei, se poate modifica fisierul web/templates/header.html.

O pagina este inclusa automat intr-un "layout" format dintr-un "header", un "content" - reprezentand continutul propriu-zis al paginii - si un "footer". Header-ul si footer-ul sunt incarcate din web/templates/header.html respectiv web/templates/footer.html. Vom modifica layout-ul pentru a afisa orice pagina pe centru:

- modificam web/templates/header.html, adaugand pe ultima linie
- modificam web/templates/footer.html, adaugand pe prima linie
- rezultatul va fi disponibil printr-un refresh al paginii

Pentru mai multe detalii consultati sectiunea 3. phprel in detaliu, subsectiunea 3.1.2. Pagini – componenta de prezentare (html) si de calcul (php).

1.4. Tag-uri de variabile si bucle

Motorul de template-uri inclus in phprel accepta in sursele html tag-uri de forma <&(...)>, care urmeaza a fi inlocuite cu anumite valori. Toate tag-urile phprel vor incepe cu caracterul "&" si vor urma un anumit format. Cele mai simple tag-uri sunt cele de variabile, care inlocuiesc direct valoarea unei variabile php in template. Pentru inceput, vom crea o variabila php a carei valoare o vom insera in template:

- modificam pages/home.php, adaugand:

```
if (!date('w') || date('w') >= 5)
        $zi = 'in weekend';
else
        $zi = 'in timpul saptamanii';
```

- modificam templates/home.html, adaugand:

```
<div class="text_negru">Acum suntem <&zi></div>
```

Variabila php $zi va fi inlocuita in template, rezultatul fiind vizibil printr-un refresh al paginii in browser. Inlocuirea s-a realizat pe baza denumirii, variabila $zi cautindu-se in "scope"-ul global. Pentru mai multe detalii despre tag-uri de variabile consultati sectiunea 3. phprel in detaliu, subsectiunile 3.1.2. Motor de template-uri si 3.3.2. Inlocuirea automata a tag-urilor phprel.

O alta categorie importanta de tag-uri phprel o constituie tag-urile de bucle. O bucla presupune repetarea unei secvente a template-ului pentru mai multe seturi de date. Un set va fii o linie dintr-un array, iar o linie va fi la randul ei compusa din mai multe variabile retinute ca array asociativ. Pentru demonstrarea acestui tag, folositi un client mysql pentru a edita tabelul back_users creat automat de Web Assistant la instalarea phprel (daca nu ati creat tabelele, puteti accesa interfata Web Assistant la "/phprel" in browser-ul dumneavoastra si Web Assistant se va oferi din nou sa creeze tabelele necesare):

- introducem mai intai in tabel trei utilizatori cu trei email-uri diferite, completand doar campurile "user" si "email"

```
admin, admin@webtest.com
owner, owner@webtest.com
guest, guest@webtest.com
```

- modificam pages/home.php, adaugand:

```
$useri = getData('back.users.all');
```

- modificam templates/home.html, adaugand:

```
<table>
<&loop "useri">
    <tr>
        <td><&user></td>
        <td><&email></td>
    </tr>
<&endloop>
</table>
```

Linia adaugata in php este folosita pentru a extrage in $useri un array de forma:

```
[0] => array(
            'user' => 'admin',
            'email' => 'admin@webtest.com',
    ),
[1] => array(
            'user' => 'owner',
            'email' => 'owner@webtest.com',
    ),
[2] => array(
            'user' => 'guest',
            'email' => 'guest@webtest.com',
    ),
```

Pentru mai multe detalii referitoare la construirea rapida a query-urilor si obtinerea rezultatelor, consultati sectiunea 3. phprel in detaliu, subsectiunea 3.2.4. Functii de scriere rapida a query-urilor si prelucrarea rezultatelor.

Continutul html inclus intre tag-ul <&loop> si corespondentul sau <&endloop> va fi repetat pentru fiecare linie a buclei, iar tag-urile <&user> si <&email> vor fi inlocuite cu valorile corespunzatoare din array pentru fiecare linie. Astfel, fiecarei linii din array ii va corespunde un <tr> in tabelul html, rezultand trei <tr>-uri, si fiecarui <tr> ii vor corespunde user-ul si email-ul liniei respective: pentru prima linie "admin" respectiv "admin@webtest.com", pentru a doua "owner" respectiv "owner@webtest.com", pentru a treia "guest" respectiv "guest@webtest.com".

Important: tag-ul de loop, ca si tag-ul de includere a elementelor paginii, contine in denumirea sa un spatiu si apoi o denumire intre ghilimele, rezultand formatul <&loop "(nume)">, dar formatul general <&(...)> este respectat.

Pentru mai multe detalii, consultati sectiunea 3. phprel in detaliu, subsectiunile 3.1.2. Motor de template-uri si 3.3.2. Inlocuirea automata a tag-urilor phprel.

1.5. Tag-uri conditionale

Template-urile pot fi ramificate folosind tag-uri conditionale, similare unui bloc conditional dintr-un limbaj de programare: <&if{(conditie)}>(...)<&endif> sau <&if{(conditie)}>(...)<&else>(...)<&endif>. Am mentionat in sectiunea precedenta formatul unui tag phprel ca fiind <&(...)>, de unde rezulta si formatul tag-ului conditional <&if{}> precum si formatele tag-urilor "else" si "endif". In plus fata de mentiunea anterioara, trebuie observat ca un tag phprel accepta acolo unde este cazul indicatii suplimentare incluse intre acolade.

Folosim un if in template pentru a stabili daca exista utilizatori in tabel:

- modificam templates/home.html, adaugand inainte de tag-ul <table>:

```
<&if{$useri}>
```

- si dupa tag-ul </table>:

```
<&else>
    <div class="text_negru">Nu exista useri.</div>
<&endif>
```

Nu s-a modificat nimic, deoarece conditia este validata, dar pentru a testa functionarea tag-ului conditional, folositi clientul mysql pentru a sterge liniile din tabelul "back_users".

Important: conditia if-ului este o conditie scrisa in limbaj php, si poate cuprinde orice conditie valida php, cu anumite exceptii in folosirea functiilor.

Pentru mai multe detalii, consultati sectiunea 3. phprel in detaliu, subsectiunea 3.3.5. Tag-uri conditionale.

1.6. Elemente incluse in pagina

Pentru a crea elemente de pagina reutilizabile, este suficient sa creati un html in directorul templates/ pe care apoi il puteti include in orice pagina folosind un tag de includere. Creem un element "stiri" pe care il includem in pagina principala:

- creem templates/stiri.html, adaugandu-i continutul:

```
Stiri web: phprel – un framework bazat pe un nou concept
```

- modificam templates/home.html, adaugand pe ultima linie:

```
<&include "stiri" />
```

Un "include" se comporta foarte similar cu o pagina: i se poate atasa un php corespondent in pages/ si poate orice tag phprel, inclusiv un nou "include". Singura exceptie o constituie statutul variabilelor definite in php-ul, o variabila fiind implicit locala sau auto-globalizata (spre deosebire de variabilele definite in php-ul atasat elementelor principale – "header", "content", "footer" – care sunt intotdeauna globale).

Pentru mai multe detalii referitoare la variabile in elemente de pagina, consultati sectiunea 3. phprel in detaliu, subsectiunea 3.3.6. Tag-uri de includere a elementelor.

Ca exemplu, definim un array pe care il inlocuim folosind bucle:

- modificam templates/stiri.html, adaugand in continuarea textului introdus:

```
<&loop "stiri">, <&titlu><&endloop>
```

- creem pages/stiri.php, cu continutul:

```
$stiri = array(
                0 => array(
                        'titlu' => 'Web 3.0',
                    ),
                1 => array(
                        'titlu' => 'Dezvoltare rapida',
                    ),
            );
```

Un refresh in browser va evidentia rezultatele. Elementul nou creat poate fi inclus in diferite pagini, ori de cate ori este nevoie.

1.7. Descriere simpla de tabel

Intre paginile unei aplicatii web si tabelele mysql exista o stransa legatura, iar intr-un tabel anumite proprietati se pastreaza de la o pagina la alta. Pentru a fi cat mai de folos, phprel trebuie sa inteleaga structura fiecarui tabel si cum se relationeaza intre ele tabelele, stiind astfel cum anume diferite tabele vor fi integrate in diferite situatii din diverse pagini. Este foarte important sa creati pentru fiecare tabel din baza de date o descriere prin care sa documentati logica tabelului, particularitatile si comportamentul fiecarui camp. Aceste descrieri pot fi create manual sau folosind interfata Web Assistant:

- folosind un client mysql, creem in baza de date un tabel "stiri", avand campurile "id, titlu, continut, data, activa":

```
CREATE TABLE `stiri` (
    `id` int(11) NOT NULL auto_increment,
    `titlu` varchar(255) default NULL,
    `continut` text,
    `data` bigint(20) default NULL,
    `activa` tinyint(4) default NULL,
    PRIMARY KEY  (`id`)
)
```

- accesam Web Assistant adaugand "/phprel" la url-ul aplicatiei web, introducem user-ul si parola selectate la instalare (sau, daca nu ati creat inca un cont de utilizator, veti putea sa il creati acum), si selectam "tabele (io)". Veti fi avertizat ca nu se poate folosi aceasta functie fara drept de scriere asupra directorului web/rules/tables/. Schimbati permisiunea directorului si asteptati cateva momente.
- Selectam tabelul "stiri" si apasam "creaza io". Web Assistant intuieste aproximativ modul de completare al fiecarui cimp, propunand o varianta ca punct de plecare. O parte din datele importante referitoare la un tabel vor fi:
    - campurile – folosite pentru a autocompleta query-urile, pentru a determina coloanele unei liste, pentru a determina tabelul folosit intr-o pagina de editare/ vizualizare precum si alte operatiuni automate
    - ordonarea predefinita – folosita pentru a autocompleta query-urile pe tabelul respectiv atunci cand omiteti sa specificati o ordonare
    - campul de validitate a afisarii – pentru anumite tabele, vor fi situatii in care anumite linii trebuie sa fie "inactive", ele nefiind afisate pe site. Prin $activeclause se poate specifica un camp cu valori 0 sau nevide, reprezentand o linie inactiva respectiv activa. Liniile inactive vor fi excluse automat din afisare.
    - campul de "denumire" – anumite tabele contin un camp ce denumeste fiecare linie, folosit de exemplu la popularea unui select-box, sau la afisarea intr-o lista. Folosind campul de denumire, phprel va putea selecta si afisa automat continutul necesar din respectivul tabel in diferite situatii
    - relationarea cu celelalte tabele – probabil cea mai importanta informatie despre un tabel este modul in care se relationeaza cu celelalte tabele, relatiile putand fi:
        - parent – child
        - linked – tabelele sunt in legatura unul cu celalalt, dar legatura poate sa existe sau nu, iar daca exista este de tip unu-la-unu
        - nomen – tabelul relationat este nomenclator pentru tabelul curent, campul relationat fiind substituit cu valoarea corespunzatoare din nomenclator
    - tipul html al anumitor campuri care nu sunt text si nu pot fi prezise cu usurinta de phprel sau care sunt checkbox

      Important: specificati mereu un camp care este checkbox pentru a va asigura ca la debifarea lui intr-o pagina de editare, valoarea este actualizata in baza de date.
    - functiile standard de input/ output – phprel propune o solutie unica de control a proceselor realizate automat sau la indicatiile dumneavoastra, bazata pe definirea unor functii cu denumire variabila, denumirea fiind relevanta pentru procesul si contextul tinta. In cazul operatiilor de intrare/ iesire pe tabele mysql, se pot defini functii standard pentru fiecare camp, functii care apoi pot fi suprascrise in diferite situatii. In exemplul de fata, campul "data" va tine data unixtime a aparitiei fiecarei stiri. Prin click pe "functii std input/ output", vom putea vizualiza o lista de campuri pentru care in mod obisnuit s-ar putea scrie functii i/o. Bifam input si output pentru data, si lasam completate campurile cu "unixtime" – Web Assistant a intuit deja ca acest camp va fi unixtime, in caz contrar am fi putut alege un alt tip de functie predefinit sau o functie "custom" pe care am fi descris-o mai tarziu in php.

- Apasam butonul "creaza io" si fisierul php corespondent al tabelului "stiri" va fi creat in directorul de descrieri ale tabelelor, web/rules/tables/. Incercati sa verificati continutul directorului, veti gasi acolo un nou fisier: io.stiri.php. Editandu-l in php, veti gasi descrierea tabelului asa cum ati configurat-o in Web Assistant.
  Important: descrierile de tabel sunt de asemenea surse php, deci destinate pentru a fi editate de dumneavoastra intr-un editor specializat. Web Assistant va ajuta doar la crearea initiala a descrierilor pentru a accelera timpul necesar inceperii proiectului si pentru a va oferi un punct de plecare pentru mai tirziu. Orice modificari ulterioare se vor face editand direct sursa php corespunzatoare tabelului. De asemenea, o descriere de tabel se poate crea fara Web Assistant prin crearea directa a fisierului de forma io.(tabel).php, unde (tabel) este numele tabelului in format phprel.

Formatul phprel de denumire al tabelului inlocuieste toate underscore-urile cu punct ("."). De exemplu, un tabel denumit in mysql "stiri_locale" va deveni "stiri.locale" in format phprel. Acest format va fi folosit in specificarea denumirii fisierului din web/rules/tables/ precum si in specificarea denumirilor tabelelor relationate (folosind $relate intr-un anumit io). In plus, formatul va putea fi folosit cu orice functie/ tag phprel care accepta denumiri de tabele.

Un model de descriere a unui tabel poate fi gasit in fisierul web/rules/tables/io.sampletable.php.

Pentru mai multe detalii referitoare la descrierile de tabele, consultati sectiunea 3. phprel in detaliu, subsectiunea 3.2.1. Descrierea tabelelor. Foarte mult din ajutorul pe care phprel reuseste sa il ofere se bazeaza pe observatiile pe care acesta le face in baza descrierilor de tabele, de aceea este foarte important ca orice tabel sa aiba o descriere cat mai completa.

1.8. Formulare

Probabil cel mai eficient ajutor acordat de phprel este la scrierea formularelor. Tag-urile in template se scriu prescurtat, iar liniile php de multe ori nici nu exista. In plus, validarea se face automat chiar si pentru validari complicate prin A.J.A.X. si se pot crea fara cod suplimentar formulare cu trimiterea datelor prin A.J.A.X. sau campuri select de tip parent-child cu repopularea optiunilor prin A.J.A.X. La revenire in pagina, dupa POST, campurile isi pastreaza automat valorile, la fel cum, in cazul paginilor de editare, isi preiau singure datele din MySQL. Iar la schimbarea paginilor sau ordonarii unei liste care are un formular de filtrare, campurile isi pastreaza valorile implicit. De asemenea, datele vor fi introduse automat in baza de date atunci cand totul este pregatit, iar redirect-ul se va calcula automat in lipsa unuia specificat de dumneavoastra. Si multe alte functionalitati.

Vom crea o pagina suplimentara la exemplul deja inceput, in care vom adauga stiri in tabela mysql creata anterior:

- creem pages/edit.stiri.php, fara continut
- creem templates/edit.stiri.html, adaugand continutul:

```
<&frm.form>
<table>
  <tr>
        <td class="text_negru">Titlu:</td>
        <td><&frm.titlu{*}></td>
  </tr>
```

```
    <tr>
        <td class="text_negru">Continut:</td>
        <td><&frm.continut></td>
    </tr>
    <tr>
        <td class="text_negru">Data:</td>
        <td><&frm.data{calendar}><img alt="Calendar" src="images/calendar.gif"
id="button_data" /></td>
    </tr>
    <tr>
        <td class="text_negru">Activa:</td>
        <td><&frm.activa></td>
    </tr>
    <tr>
        <td> </td>
        <td><&frm.goPost><&frm.submit{value="Salveaza"}></td>
    </tr>
</table>
<&endform>
```

- modificam templates/home.html, adaugand pe ultima linie un link catre noua pagina:
  `<div class="text_verde_2"><a href="edit.stiri/edit/0">Adauga stire</a></div>`
- Dupa un refresh al paginii, urmati linkul catre pagina de adaugare. Completati formularul cu date de test si apasati "Salveaza". Am adaugat formularului si validare pentru campul "titlu" (prin atributul "*", pentru detalii consultati sectiunea 3. phprel in detaliu, subsectiunea 3.4.9. Grupuri si validare) si va trebui sa completati obligatoriu o valoare. Pentru a verifica functionarea, folositi clientul mysql pentru a lista liniile din tabelul "stiri". Observati si formatul unixtime al campului "data".

Tag-ul de formular pus la dispozitie de phprel este de forma <&frm.(nume)>. Acest tag va fi folosit pentru a inlocui orice tag html referitor la formulare, precum <form>, <input>, <textarea>, <select>. Un formular va incepe fie cu <&frm.form> in cazul in care construim un formular principal, fie cu <&frm.form.(nume)>, unde (nume) poate fi orice denumire, in cazul in care construim un formular secundar. Acest tag va fi transformat in tag-ul html <form>. Vom incheia formularul cu <&endform> corespunzator tag-ului html </form>. Butonul de submit se va scrie folosind <&frm.submit> in cazul formularului principal, sau <&frm.submit.(nume form)> unde (nume form) este numele formularului secundar dat in denumirea tag-ului <&frm.form.(nume form)>, in cazul unui formular secundar. Pentru moment, vom presupune intuitiv cand un formular este principal si cand secundar.

Orice camp al formularului va fi scris cu un tag de forma <&frm.(nume camp)> sau, daca este cazul, <&frm.(nume camp){(atribute suplimentare)}>. Tag-ul urmeaza tiparul de tag phprel deja descris, incepand cu "&" si poate contine intre acolade indicatii suplimentare.

Important: phprel a intuit tabelul mysql la care se refera formularul, precum si tipul campurilor "titlu", "continut", "data" pe baza unor observatii. Acesti parametrii se pot specifica in cazul in care phprel nu reuseste sa determine unul dintre ei. Presupunerile facute de phprel, precum si construirea si executarea automata a query-ului dupa apasarea butonului de submit nu ar fi fost posibile fara descrierea tabelului stiri.

Suplimentar, pentru ca salvarea in baza de date respectiv citirea din baza de date sa se faca pentru formularul principal, link-ul trebuie sa contina o variabila GET "edit" (in cazul de fata, edit egal zero insemnand adaugare, edit egal cu o valoare pozitiva ar fi insemnat editarea liniei cu id egal cu valoarea respectiva) iar formularul trebuie sa contina un camp (implicit hidden) "goPost". Variabila GET este folosita pentru a determina tipul de query de actualizare (insert sau update) iar campul "goPost" este folosit ca indicatie a faptului ca doriti scrierea automata a query-urilor pentru respectivul formular.

Important: scrierea automata a query-urilor, completarea campurilor din baza de date, si inserarea sau modificarea valorilor in tabel se vor realiza doar in cazul formularului principal, acolo unde exista un $_GET['edit'] valid si un goPost.

Incercand un refresh al paginii dupa apasarea butonului "Salveaza", veti constata ca pagina a fost redirectionata automat in acelasi loc, fiind gata pentru o noua adaugare (acest lucru se deduce din faptul ca mesajul obisnuit care v-ar avertiza ca un formular va fi retrimis in cazul in care continuati refresh-ul nu apare).

Vom modifica functionarea paginii pentru a reveni la pagina anterioara dupa adaugarea unei stiri:

- modificam templates/home.html, schimband linkul de "adauga stire", noul link devenind:

```
edit.stiri/edit/0/?nav:back=<&nav.generate>
```

- "nav:back" este folosit pentru a indica linkul de intoarcere si este de folos in diverse situatii. <&nav.generate> este un tag de variabila special folosit pentru a genera linkul de intoarcere. Pentru mai multe detalii, consultati sectiunea 3. phprel in detaliu, subsectiunile 3.3.3. Variabile globale si 3.9. Navigare.
- Incercam sa adaugam o noua stire pentru a verifica rezultatul.

Pentru detalii referitoare la parametrul GET "secure" adaugat automat, consultati sectiunea 4. Securitate si performanta, subsectiunea 4.1.1. Validarea url-urilor. De asemenea, pentru detalii privind functionarea url rewrite-ului implicit si metode de a modifica url-ul, inclusiv ascunderea parametrului edit/0 din url (impreuna cu "secure"), consultati sectiunea 3.6. Url rewrite.

Nu in ultimul rand, pentru (multe) detalii privind formularele, consultati sectiunea 3. phprel in detaliu, subsectiunea 3.4. Formulare.

1.9. Liste

Foarte multe pagini, fie in backend fie in frontend folosesc listari ale informatiilor din tabele in diferite formate. In general pentru un proiect, design-ul listarilor este pastrat pe fiecare lista pentru coerenta, sau un numar finit de design-uri este folosit pentru a afisa un numar mare de liste. In phprel, listele se pot crea cu cateva linii si pot fi particularizate sa indeplineasca orice cerinte. Functionalitati precum ordonare dupa anumite coloane, paginare, detalii suplimentare prin A.J.A.X. la click pe o linie, adaugarea actiunilor clasice de vizualizare, editare, stergere, stergerea unei linii pentru care s-a actionat butonul de stergere, prevenirea stergerii in cazul in care linia are dependinte in alte tabele, filtrare pe baza unui formular, si multe altele sunt implementate automat de nucleul phprel. Listele de orice fel vor putea fi create usor, intr-o linie html, foarte rar cu mici alte indicatii din php.

Vom adauga la exemplul deja construit o lista simpla cu stirile adaugate in baza de date:

- modificam templates/home.html, adaugand inainte de linkul "adauga stire" continutul:

```
<table>
  <tr>
```

```
                    <td width="530">
                        <&lst.lista{table="stiri"; columns="Titlu, Continut, Data";}>
                    </td>
                </tr>
            </table>
```

- verificam rezultatul printr-un refresh al paginii

In phprel, o lista se creaza printr-un tag de lista, de forma <&lst.(nume){(indicatii)}>, unde (nume) este o denumire data listei, iar intre acolade se specifica detaliile privind lista. Acest tag este in concordanta cu tiparul de tag phprel <&(nume){(indicatii suplimentare)}> deja descris.
Am indicat tabelul principal pe baza caruia se construieste lista, precum si denumirea coloanelor care apar in listare. Ordinea liniilor in tabel este data de expresia specificata in $defaultorder pentru tabelul "stiri".

Important: campurile atasate fiecarei coloane au fost determinate automat de phprel folosind descrierea tabelului.

Lista afisata foloseste un "layout" (template html al listei) si un cadru ("frame", referindu-se la o variatie a aceluiasi design prin schimbare de stiluri, clase, culori si/ sau dimensiuni) deja scris si menit sa fie un punct de plecare pentru dumneavoastra in particularizarea design-ului listelor. De notat faptul ca design-urile sunt universal valabile pentru phprel si pot fi reutilizate de la un proiect la altul. Definirea unui cadru al aceluiasi layout se face in directorul web/settings/ printr-un fisier cu denumirea (nume cadru).frame.php, cadrul implicit fiind "default.frame.php". De asemenea, un layout este un template html, situat in web/templates/, cel implicit fiind disponibil in web/templates/list-layout.html, respectiv web/templates/list-pagination.html.

Vom modifca in continuare design-ul listei cu unul in stil phprel:

- modificam templates/home.html, adaugand la indicatiile listei urmatoarele atribute (atributele se vor desparti prin punct-si-virgula):
  layout="phprel"; frame="phprel"; align="lst.actions: right";
  Tag-ul de lista a devenit: <&lst.lista{table="stiri"; columns="Titlu, Continut, Data"; layout="phprel"; frame="phprel"; align="lst.actions: right"}>
- reincarcam pagina pentru a verifica rezultatul

Layout si frame au fost deja explicate. Atributul "align" specifica alinierea pentru fiecare coloana (liniile de date, nu si denumirile coloanelor, pentru denumirile coloanelor folositi "halign"), iar coloana actiunilor este denumita "lst.actions". Pentru a seta, de exemplu, align pe centru pentru titlu, noul atribut align ar arata:

```
align="lst.actions: right, titlu: center"
```

Important: indicatiile suplimentare pentru un tag, scrise intre acolade, au urmatoarea sintaxa:

- Indicatiile sunt formate din mai multe atribute, iar fiecare atribut are o valoare: atribut="valoare"; altatribut="altavaloare"; ...
- Sintaxa valorilor este sau un sir de caractere simplu, sau mai multe siruri de caractere despartite prin virgula, sau mai multe siruri de forma subatribut: subvaloare despartite prin virgula.
- In plus, se pot specifica mai multe subatribute pentru aceeasi valoare, printr-un sir de forma subatribut: altsubatribut: subvaloare
- Exemple:
  - table="stiri"; frame="phprel" (varianta simpla, atribut="valoare")
  - columns="Titlu, Continut, Data" (varianta atribut="valori separate prin virgula")

- align="lst.actions: right, titlu: center" (varianta atribut="valori de tip subatribut: subvaloare separate prin virgula)
- align="lst.actions: titlu: data: right" (varianta atribut="valori de tip subatribut: altsubatribut: subvaloare separate prin virgula)

Continuam modificarile, adaugand ordonare pentru coloanele "titlu" si "data":

- modificam templates/home.html, adaugand la tag-ul de lista atributul (separat de atributul precedent prin punct-si-virgula):

  sortablecolumns="titlu, data"

  Noul tag a devenit: <&lst.lista{table="stiri"; columns="Titlu, Continut, Data"; layout="phprel"; frame="phprel"; align="lst.actions: right, titlu: center"; sortablecolumns="titlu, data"}>

Important: sesizam ca nu sunt afisate in lista stirile pentru care nu s-a bifat campul "Activa". Daca nu aveti o astfel de stire in tabel, adaugati una pentru a verifica acest rezultat. Acest fapt este datorat expresiei indicate prin $activeclause din descrierea tabelului. De asemenea, denumirea coloanei "Titlu" este acum in *italics*, deoarece lista este ordonata implicit dupa "titlu", conform $defaultorder din descrierea tabelului. Pentru a ordona descrescator dupa "Titlu", se poate da click pe denumirea coloanei. La click pe coloana "Data", lista se va ordona crescator dupa "data". La un nou click, ordonarea va fi descrescatoare dupa "data".

Ultimul pas este asigurarea functionarii actiunilor:

- actiunea de stergere functioneaza deja, verificati stergand din lista una dintre stirile adaugate
- modificam templates/home.html, adaugand la atributele listei:

  editlink="edit.stire/"
- noul tag de lista este: <&lst.lista{table="stiri"; columns="Titlu, Continut, Data"; layout="phprel"; frame="phprel"; align="lst.actions: right, titlu: center"; sortablecolumns="titlu, data"; editlink="edit.stiri/"}>
- verificam rezultatul printr-un refresh si click pe editare in dreptul uneia dintre stiri.

Formularul scris anterior functioneaza deja ca formular de editare, datele au fost selectate automat si precompletate, iar la efectuarea unei modificari si click pe "Salveaza", query-ul de update se va construi si executa automat, apoi pagina se va redirectiona catre pagina anterioara.

Important: actiunile predefinite ca imagini se pot schimba, modificand in cfgs/advanced/assistant.inc.php variabilele $Tasi_setdefaultedit si $Tasi_setdefaultdel. De asemenea, orice actiune de pe o lista se poate schimba printr-o functie de control, si chiar ascunde in functie de linie pentru a nu permite anumite actiuni in anumite contexte.

Pentru mai multe detalii, consultati sectiunea 3. phprel in detaliu, subsectiunea 3.5. Liste.

2. Elemente de baza

Aceasta sectiune este o continuare a primei sectiuni in care ati facut cunostinta cu cateva concepte din phprel, prezentate intuitiv. In cazul in care nu ati parcurs sectiunea 1. phprel la prima vedere, este recomandat sa reveniti acum pentru a construi mini-exemplul phprel pe care il vom folosi in continuare.

2.1. Url rewrite

Pentru varianta "front", phprel dispune de o tehnologie inovativa de url rewrite, care va permite sa lucrati transparent cu url-uri de orice forma (perfect optimizate pentru motoarele de cautare). Fara nici o modificare in ".htaccess", si cu doar cateva linii php, ii veti indica nucleului phprel exact tiparul unui url anume, oricat de complicat ar fi. In plus, query-urile pentru transformarea din denumiri in id-uri se vor realiza automat, iar la cerere se vor realiza si query-urile de selectare a continutului corespunzator respectivului id. Tehnologia inovativa de url rewrite va permite sa specificati in ce mod vor fi interpretate partile url-ului:

- un url se va compune din mai multe parti separate prin slash-uri "/"
- in cazul in care mai multe parti concatenate reprezinta un concept, se pot concatena respectivele parti pentru a forma o singura parte
- o parte poate reprezenta:
    - denumirea paginii
    - denumirea unei variabile GET
    - valoarea unei variabile GET
- o variabila poate lipsi, fiind subinteleasa, caz in care in url va aparea doar valoarea – pentru acea valoare se poate specifica variabila destinatie
  Exemple:
    - http://.../produse/edit/0 are ca parti:
        - produse: denumirea paginii
        - edit: nume de variabila GET
        - 0: valoarea variabilei GET edit
    - http://.../lista/produse/categorie/Electronice
        - lista.produse: denumirea paginii, concatenare a partilor 0 si 1 a url-ului
        - categorie: nume de variabila GET
        - Electronice: valoarea variabilei GET categorie, in mod ideal transformata in id
    - http://.../produs/phprel
        - produs: denumirea paginii
        - phprel: valoare a unei variabile GET subinteleasa, de exemplu produs
    - http://.../Electronice
        - pagina poate fi subinteleasa ca fiind categorie
        - Electronice: valoare pentru o variabila GET subinteleasa, de exemplu categorie, in mod ideal transformata in id

Vom adauga un nou link in pagina de home, catre o pagina stire, care va primi si denumirea unei stiri:

- adaugam o noua stire in baza de date, cu titlul "phprel" si activa (folosind linkul "adauga stire" construit in sectiunea 1. phprel la prima vedere)
- modificam templates/home.html, adaugand pe ultima linie continutul:
  ```
  <div class="text_negru"><a href="stire/phprel">Vezi stirea</a></div>
  ```
- creem pages/stire.php fara continut
- creem templates/stire.html cu continutul:
  ```
  <&loop "stire">
  <div align="left" class="text_negru">
  ```

```
<table>
    <tr>
            <td class="text_negru" style="font-weight: bold;"><&titlu> <span
style="font-weight: normal;">(<&data>)</span></td>
    </tr>
    <tr>
            <td class="text_verde_2"><&continut></td>
    </tr>
</table>
        <a href="home">Inapoi</a>
</div>
<&endloop>
```

- modificam web/settings/rewrite.rules.php, adaugand in switch, inainte de "default":

```
case 'stire':

        $pag = $req[0];
        $offset = 0;

break;
```

- iar dupa switch, la finalul php-ului:

```
$assocsubstitute['stire'] = 'stiri.toid::globalize';
```

Switch-ul se face dupa $req[0], respectiv prima parte a url-ului, in cazul de fata "stire". $req este un array simplu cu partile url-ului. Stabilim pagina prin $pag, in cazul de fata $pag va fi de asemenea "stire". Indicam prin $offset incepand cu a cata parte a url-ului trebuie realizata procesarea variabilelor GET, in cazul de fata incepem cu aceeasi prima parte. Astfel $req[0] va fi interpretat ca denumire de variabila GET, rezultand variabila GET "stire". $req[1] va fi interpretat ca valoare a variabilei GET "stire", deci "phprel" va fi considerata valoarea variabilei stire. Dar folosind $assocsubstitute am precizat faptul ca variabila GET "stire" trebuie transformata intr-un id pe baza tabelului "stiri". Sintaxa folosita "stiri.toid" este compatibila cu sintaxa functiei phprel getData de acces rapid la baza de date. Pentru detalii, consultati sectiunea curenta, subsectiunea 2.2. Accesul rapid la baza de date si sectiunea 3. phprel in detaliu, subsectiunea 3.2.4. Functii de scriere rapida a query-urilor si prelucrarea rezultatelor. In final, folosind "globalize" solicitam nucleului phprel sa creeze o variabila globala cu aceeasi denumire a variabilei GET in care sa selecteze linia corespunzatoare id-ului rezultat.

Important: separatorul "::" este des folosit in phprel, pentru diferite situatii in care membrului stang ii trimitem ca valoare membrul drept.

- Un refresh al paginii ne va arata rezultatul. Stirea a fost selectata si afisata in pagina (variabila globala $stire creata in rewrite este inlocuita ca bucla – evident cu un singur rand – in continutul paginii).

Asadar, url-ul http://.../stire/phprel a fost interpretat dupa cum urmeaza:

- partile url-ului sunt "stire" si "phprel"
- "stire" se considera denumirea paginii, si in acelasi timp denumirea unei variabile GET
- "phprel" se considera valoarea variabilei GET "stire", transformata folosind tabelul "stiri" in id

Important: transformarea s-a facut folosind $namefield din descrierea tabelului.

- In plus, s-a creat variabila globala $stire continand linia din tabelul "stiri" avand id-ul rezultat in $_GET['stire']

Vom incerca in continuare o a doua varianta de rewrite:

- "stire" se considera denumirea paginii
- "phprel" se considera valoarea unei variabile subintelese, denumite "stire", si transformata in id apoi globalizata
- modificam settings/rewrite.rules.php, lasand ultima linie cu $assocsubstitute nemodificata, dar schimband case-ul pentru 'stire':

```
case 'stire':

    $pag = $req[0];
    $offset = 1;
    $vars[1] = 'stire';

break;
```

Pagina se specifica la fel. Diferenta consta in partea url-ului cu care se incepe procesarea variabilelor GET: pentru prima metoda, $offset a fost 0 (procesarea s-a inceput cu "stire"), acum $offset este 1, deci se incepe procesarea cu "phprel". Folosind $vars indicam faptul ca partea url-ului cu numarul dat, in cazul de fata 1, este valoare a unei variabile subintelese cu denumire data, in cazul de fata "stire". $_GET['stire'] va fi deci "phprel", dar prin $assocsubstitute am precizat ca variabila $_GET['stire'] va fi transformata in id pe baza tabelului "stiri", rezultand $_GET['stire'] = (id-ul stirii) si, in plus, am cerut sa se creeze variabila globala $stire. Metoda doi are acelasi efect, producand in final aceeasi pagina.

Prezentam o ultima metoda de a realiza acelasi lucru, pentru a documenta toate optiunile disponibile pentru rewrite:

- logica este similara metodei doi, dar acum vom face substitutia denumire – id in alt mod
- modificam settings/rewrite.rules.php, stergand ultima linie cu $assocsubstitute, si schimband case-ul pentru 'stire':

```
case 'stire':

    $pag = $req[0];
    $offset = 1;
    $vars[1] = 'stire';
    $substitute[1] = 'stiri.toid::globalize';

break;
```

Diferenta fata de metoda doi consta in modul de realizare a transformarii din denumire "phprel" in id a valorii pentru variabila $_GET['stire']. Substitutia nu mai este valabila global, pentru intreg rewrite-ul, ci este specifica situatiei/ paginii curente. Se pot astfel suprascrie comportamentele standard in anumite situatii, daca este cazul.

Vom rezolva un ultim exemplu fictiv, pentru clarificare:

- se considera url-ul http://.../lista/produse/Electronice
- partile lui sunt:
    - "lista", "produse", "Electronice"
- "lista" si "produse" trebuie concatenate in denumirea paginii "lista.produse"

- "Electronice" este o valoare pentru o variabila subinteleasa "categorie", pe care nu dorim sa o globalizam, ci doar sa setam $_GET['categorie'] = (id-ul categoriei cu denumire "Electronice"). Presupunem existenta unui tabel categorii pentru care deja am creat descrierea.
- In settings/rewrite.rules.php vom trece in interiorul switch-ului:

```
case 'lista':

    $pag = $req[0].'.'.$req[1];
    $offset = 2;
    $vars[2] = 'categorie';

break;
```

- iar la final, in afara switch-ului:

```
$assocsubstitute['categorie'] = 'categorii.toid';
```

Pentru mai multe detalii referitoare la url rewrite si modalitati de traducere a url-urilor folosind $vars, $assocsubstitute, $pag, etc. precum si la tehnici de transformare in id pentru situatii parent-child (cand denumirea nu determina unic un id, ci este necesar si valoarea unei a doua variabile GET reprezentand "parintele" denumirii), consultati sectiunea 3. phprel in detaliu, subsectiunea 3.6. Url rewrite.

Important: o data cele cateva linii de php scrise in settings/rewrite.rules.php, scrierea aplicatiei web se face obisnuit, ca si cum nu ar exista url rewrite, si url-ul ar fi fost dat in mod clasic. Pentru exemplul anterior, cel fictiv, cu categorii, am fi scris in pages/lista.produse.php:

```
$id = $_GET['categorie'];
```

ca si cum am fi accesat pagina printr-un link de forma "index.php?page=lista.produse&categorie=1".

2.2. Accesul rapid la baza de date

In phprel, scrierea unui query este sustinuta automat si simplificata pentru ca dumneavoastra sa castigati timp cu fiecare apasare a unei taste. Aveti la dispozitie cateva functii eficiente pe care le veti folosi pentru a selecta, insera, modifica si sterge date. De asemenea, procesele de citire si scriere a datelor in mysql sunt complet controlate prin functii de control cu denumire variabila.

Revenim la exemplul din sectiunea 1. phprel la prima vedere, subsectiunea 1.4. Tag-uri de variabile si bucle. Am folosit o linie pentru a citi utilizatorii din tabel:

```
$useri = getData('back.users.all');
```

Functia "getData" este folosita la selectarea si obtinerea datelor din mysql – phprel va construi un query de select conform indicatiilor, il va executa, va citi rezultatele intr-un array si va modifica toate campurile care sunt influentate de functii de control. Sintaxa functiei este:

- (array/ valoare) getData('(tabel in format phprel).(functie)[(indicatii suplimentare)]'[, (parametru al functiei solicitate)]);
- Tabelul se denumeste in format phprel similar cu denumirea lui in mysql, dar se inlocuieste underscore-ul cu punct. De exemplu, tabelul "stiri" este denumit in format phprel la fel, "stiri", dar tabelul "back_users" este denumit in phprel "back.users".
- Functia solicitata se va aplica tabelului si poate fi una dintre functiile predefinite:
  - all – va intoarce toate liniile tabelului (care vor corespunde indicatiilor suplimentare)

        - search – va intoarce liniile care corespund unei conditii "where" de cautare, specificata prin "parametrul functiei solicitate"
        - filter – va intoarce liniile care corespund unui criteriu de filtrare, specificat prin "parametrul functiei solicitate". Criteriul se aplica unui camp sau mai multor campuri din cele specificate in descrierea tabelului la $fiterfield.
        - line – va intoarce o singura linie corespunzatoare unui id specificat prin "parametrul functiei solicitate"
        - toid – va intoarce id-ul liniei specificate prin valoarea campului completat la $namefield in descrierea tabelului
        - toname – va intoarce valoarea campului completat la $namefield pentru un id specificat prin "parametrul functiei solicitate"
    - Indicatiile suplimentare pot contine unul sau mai multi dintre parametrii:
        - la functia solicitata se poate adauga: (x, y) – doua valori separate prin virgula, intre paranteze, semnificand parametrii de "limit" asa cum apar ei intr-un query
            Exemplu: getData('back.users.all(0,10)');
        - .order – specifica o ordonare alta decat cea implicita preluata din descrierea tabelului
        - .group – specifica gruparea datelor dupa un anumit camp
        - .recursive[*nivel] – datele vor fi selectate recursiv si din tabelele invecinate, pentru fiecare linie array-ul de rezultate fiind completat cu datele aditionale relevante din tabelele specificate folosind $relate in descrierea tabelului. Folosind asterix si un numar se poate preciza pana la ce nivel de adancime sa se selecteze recursiv datele.

    Nota: variabilele $namefield, $relate, $filterfield sunt specificate in descrierea tabelului respectiv, in directorul web/rules/tables/, fisierul io.(nume tabel).php.

Important: getData functioneaza doar pentru tabele pentru care exista descrierea corespondenta in web/rules/tables/, creata de dumneavoastra sau cu ajutorul interfetei Web Assistant. Pentru detalii suplimentare, revedeti sectiunea 1. phprel la prima vedere, subsectiunea 1.7. Descriere simpla de tabel, si sectiunea 3. phprel in detaliu, subsectiunea 3.2. Accesul la baza de date. De asemenea, o functie care depinda explicit de un parametru din descrierea tabelului nu va functia in lipsa acelui parametru din descriere.

Exemplul de mai sus, $useri = getData('back.users.all')**,** intoarce in variabila $useri un array de randuri (un rand fiind un array asociativ de tip (camp) => (valoare)) corespunzator tuturor liniilor din tabelul back.users.

Vom parcurge in continuare cateva exemple pentru a indica modul de functionare al functiei getData (exemplele presupun ca ati parcurs sectiunea anterioara si ati construit mini-exemplul de familiarizare cu phprel):

- Folosim linkul de "adauga stire" pentru a adauga o stire cu titlul "phprel" (daca nu exista deja), si cel putin patru stiri in total.
- Modificam pages/home.php, adaugand pe ultima linie urmatorul continut:

```
$date = getData('stiri.search', "titlu LIKE '%php%'");
print_v($date, true);
```

- Am folosit print_v pentru a afisa o variabila si a incheia imediat executia scriptului. Functia este public disponibila in functions.front.php respectiv functions.back.php pentru variantele "front" si "back". Pentru a parcurge rapid aceasta sectiune, nu vom mai parse-a fiecare rezultat in template, ne vom multumi sa afisam rapid efectul folosirii unei anumite functii.

- Reincarcam pagina pentru a observa rezultatul.
- Modificam din nou pages/home.php, comentand apelul getData anterior si adaugand sub el (inainte de linia cu apelul functiei "print_v"):

```
$date = getData('stiri.all(0,3).order(RAND())');
```

- Reincarcam pagina pentru rezultat. Au fost selectate trei stiri aleator.

Important: functiile de control standard corespunzatoare campurilor din tabel s-au aplicat automat. Pentru mai multe detalii referitoare la cum pot fi suprascrise comportamentele standard in diferite situatii sau pentru diferite pagini, precum si cum se pot selecta date fara a aplica functiile de control, in ce ordine de prioritate se aplica functiile de control, etc. consultati sectiunea 3. phprel in detaliu, subsectiunea 3. Functii de control, comportament standard, comportamente diferite pe pagina.

- Pentru un nou exemplu, adaugam la baza de date inca doua tabele:

```
CREATE TABLE `autori` (
        `id` int(11) NOT NULL auto_increment,
        `nume` varchar(255) default NULL,
        `reprezentant_al` varchar(255) default NULL,
        PRIMARY KEY  (`id`)
      );
CREATE TABLE `articole` (
         `id` int(11) NOT NULL auto_increment,
         `id_stire` int(11) default NULL,
         `id_autor` int(11) default NULL,
         `articol` varchar(255) default NULL,
         `text` text,
         `ord` int(11) default NULL,
         PRIMARY KEY  (`id`)
        );
```

- Si adaugam in tabela de autori doi autori cu doua valori diferite pentru campul "reprezentant_al", una dintre aceste valori fiind valoarea de test "Web 3.0 Magazine", iar in tabela de articole vom adauga trei articole, doua pentru autorul reprezentant al "Web 3.0 Magazine" si legate prin id unul de stirea cu numele "phprel", altul de o alta stire, si inca un articol pentru cel de-al doilea autor si legat de aceeasi stire "phprel". Un model poate fi:
  - INSERT INTO `autori` (`nume`, `reprezentant_al`) VALUES ('primul_nume', 'Web 3.0 Magazine'), ('al_doilea_nume', 'Alt_Magazine')
  - INSERT INTO `articole` (`id_stire`, `id_autor`, `articol`, `text`, `ord`) VALUES ('1', '1', 'Un articol despre phprel', 'Continutul articolului', '1'), ('2', '1', 'Un articol despre alta stire', 'Continutul articolului 2', '2'), ('1', '2', 'Un alt articol despre phprel', 'Continutul articolului 3', '3') – unde "id_stire" trebuie sa fie in loc de "1" id-ul stirii cu denumirea "phprel" iar in loc de "2" id-ul unei alte stiri, iar "id_autor" sa fie dupa caz id-ul primului autor introdus si al celui de-al doilea
- Stergem din directorul web/rules/tables/ descrierea veche a tabelului "stiri", generata automat din interfata Web Assistant, apoi incarcam Web Assistant intr-o pagina separata pentru a crea descrierile pentru toate trei tabelele. Aceasta functionalitate a fost explicata in sectiunea anterioara, subsectiunea 1. 7. Descriere simpla de tabel, iar pentru mai multe detalii consultati sectiunea 3. phprel in detaliu, subsectiunea 3.2.1. Descrierea tabelelor.

- Creem descrierile de tabele, selectand in Web Assistant, tabele (io), tabelele stiri, autori, articole si apasand "creaza io". Pentru tabelul stiri, nu uitati sa bifati functii std input/ output pentru campul "data".
- Modificam pages/home.php, comentand ultimul apel getData si adaugand unul nou:

  ```
  $date = getData('(articole, autori, stiri).search', "titlu LIKE '%php%' AND reprezentant_al='Web 3.0 Magazine' AND articol LIKE '%phprel%'");
  ```
- Pentru a urmari ce se intampla, trecem in modul de dezvoltare, adaugand la inceputul url-ului directiva "devel/". Presupunand ca url-ul ar fi fost "http://www.domeniu.ro/director/home", noul url va fi "http://www.domeniu.ro/director/devel/home". Vor fi afisate query-urile efectuate, si se poate constata ca phprel a realizat automat query-ul corect, join-ind tabelele corect si aplicand conditiile, completand campurile si ordonarea. Daca datele introduse au respectat exemplul sau un exemplu compatibil, ar trebui sa existe un rezultat al query-ului pe toate cele trei tabele care respecta conditiile.

Important: getData poate fi folosit pentru a selecta datele de pe multiple tabele, join-ul realizandu-se automat. Tabelele se vor preciza separate prin virgula si intre paranteze, sintaxa standard a functiei respectandu-se dar denumirea unui singur tabel se inlocuieste cu denumirile multiplelor tabele. Parantezele au rolul de a pastra semantica exprimarii:

- tabel.functie insemna ca functia solicitata se aplica tabelului
- functia in cazul mai multor tabele se aplica join-ului tabelelor considerat ca o singura "entitate rezultat", deci se aplica unei structuri: (...).functie, acea structura fiind specificata in interiorul parantezelor.

Este de asemenea **important** de remarcat faptul ca phprel poate sa join-eze tabele care nu sunt legate direct. Pentru doua sau mai multe tabele cerute de utilizator, phprel va descoperi (daca exista) calea de a le lega, si va executa functia solicitata:

```
$date = getData('(autori, stiri).search.group(autori.id)', "1");
```

- Adaugati acest nou apel il locul celui vechi si verificati rezultatul, urmarind si query-ul generat. Am selectat toti autorii care au scris un articol pentru cel putin o stire (grup dupa id-ul autorului). In mod automat, phprel a descoperit legatura intre tabelele solicitate si a realizat query-ul folosind si tabelul de articole, dar a selectat campurile doar din tabelele cerute.

Important: daca doua campuri se repeta ca denumire – si deci s-ar suprapune in rezultat, cel de-al doilea (si toate celelalte nume) care s-ar suprapune peste un camp deja extras vor fi prefixate cu denumirea tabelului. Campul rezultat va fi de forma (tabel)___(camp) si este un format standard pentru phprel de selectarea a campurilor in conditiile in care sunt folosite mai multe tabele in query. De notat ca toate campurile unice au fost extrase cu denumirile lor exacte, fara prefix. De aici se ajunge la concluzia ca ordinea in care sunt precizate tabelele conteaza, si, in plus, trebuie notat ca phprel acorda importanta (in diverse situatii, nu numai in cazul functiei getData) primului tabel dintr-o lista de tabele ca fiind tabelul "principal", in timp ce celelalte vor fi tabele "secundare" care aduc informatii suplimentare tabelului principal.

Urmatorul exemplu se bazeaza pe functia "filter", si o vom demonstra pe baza tabelului "articole" – pentru care Web Assistant a propus automat doua campuri pentru $filterfield, "id_stire" si "id_autor". Folosind clientul mysql, aflati id-ul stirii cu titlul "phprel" si retineti-l.

- Vom presupune ca acel id este 1 si vom scrie un nou apel getData in pages/home.php:

  ```
  $date = getData('articole.filter', $stire = 1);
  ```
  sau simplu: `$date = getData('articole.filter', 1);`

- Rezultatul disponibil la refresh este selectarea articolelor care apartin de stirea cu id-ul specificat.
- Pentru a selecta articolele corespunzatoare unei anume stiri si unui anume autor (respectiv cu un anume "id_stire" si un anume "id_autor"), vom scrie:

  ```
  $date = getData('articole.filter', array($stire = 1, $autor = 1));
  ```

  sau simplu: `$date = getData('articole.filter', array(1, 1));`
- Inlocuiti valorile pentru stire si autor conform tabelelor dumneavoastra, si verificati rezultatul in browser.
- Functia filter filtreaza dupa valorile campurilor specificate prin $filterfield, in cazul in care se specifica o singura valoare se va filtra dupa primul (sau unicul) camp, daca se specifica doua valori se va filtra dupa primele doua campuri, apoi cu trei valori primele trei campuri etc. in functie de cate campuri sunt specificate in $filterfield. Orice camp poate fii specificat in $filterfield, cu conditia, bineinteles, sa apartina tabelului.
- Ultimul exemplu este pentru a demonstra functioarea functiei "line", lasandu-va pe dumneavoastra sa incercati, daca este cazul, functiile "toid" si "toname". Vom presupune din nou ca exista o stire cu id 1 in tabelul "stiri":

  ```
  $date = getData('stiri.line', 1);
  ```

Urmatoarea functie disponibila pentru acces rapid la date este functia "select", care construieste un query "SELECT". Sintaxa functiei este urmatoarea:

- (mysql result) `select ($campuri, $tabele, $where, $order = '', $group = '', $limit = '', $having = '')`

Ordinea parametrilor nu este ordinea reala dintr-un query mysql select, parametrii fiind rearanjati in functie de probabilitatea utilizarii pentru un maxim de eficienta in situatiile reale. De asemenea, parametrul $campuri se poate omite completand sirul vid '', si phprel va selecta campurile corespunzatoare tabelelor precizate. Tabelele se specifica separate prin virgula, iar ceilalti parametrii trebuie sa fie exact asa cum ar fi ei intr-un query mysql obisnuit. De exemplu:

```
select('', 'stiri', '1')
```

Dar vom dori sa dispunem direct de liniile rezultate, ca urmare vom scrie:

```
$date = fetch(select('', 'stiri', '1'));
```

Afisand continutul variabilei $date cu print_v, vom constata ca au fost selectate toate liniile din tabelul stiri, si rezultatele au fost citite automat sub forma unui array de randuri. Citirea rezultatelor se realizeaza cu functia fetch, care are urmatoarea sintaxa:

- (array de randuri/ array asociativ/ valoare) `fetch($result[, $return_as_array])`
- In cazul in care parametrul $return_as_array este false sau nespecificat, functia va intoarce un array de randuri daca s-au selectat mai multe campuri si au rezultat mai multe linii, un array asociativ simplu corespunzator unui singur rand daca s-au selectat mai multe campuri dar a rezultat o singura linie sau o valoare corespunzatoare valorii unicului camp daca s-a selectat un camp si a rezultat o singura linie. Daca parametrul $return_as_array este specificat ca true, functia va intoarce un array de randuri indiferent de rezultat (inclusiv pentru un rezultat vid, va intoarce un array fara elemente).

Exemple:

- Exemplul specificat anterior va intoarce un array de randuri, pentru ca tabelul "stiri" are mai multe linii. Array-ul de randuri va corespunde liinilor din tabel, si va contine pentru fiecare rand toate campurile tabelului.

$date = fetch(select('', 'stiri', '1', '', '', '0, 1')); - va intoarce un array asociativ simplu corespunzator unicului rand selectat (am folosit LIMIT 0,1).

- $date = fetch(select('titlu', 'stiri', '1', '', '', '0,1')); - va intoarce direct titlul primei stiri din tabel (considerata in ordinea ascendenta a titlurilor, daca s-a pastrat $defaultorder propus de Web Assistant in descrierea tabelului).

Functia fetch nu va aplica automat functiile de control stabilite de dumneavoastra, selectand datele exact asa cum apar in tabele. Pentru aplicarea functiilor de control, rezultatul intors de "fetch" se va trimite catre functia "alter" de modificare a rezultatelor:

```
$date = alter(fetch(select('', 'stiri', '1'), true));
```

In acest ultim exemplu, campul "data" nu va mai fi in format unixtime, ci va fi in format d.m.Y.

In plus, pentru obtinerea rapida (intr-o singura linie) a unei valori dorite, se pot folosi functiile:

- (valoare) getfield ($camp, $array) – intoarce valoarea campului solicitat corespunzatoare primului (sau singurului) rand din array-ul dat.
- (array asociativ/ valoare) shift ($array) – intoarce primul rand din $array, ca array asociativ, in cazul in care $array este un array de randuri sau valoarea primului camp din $array in cazul in care $array este un array asociativ.

Exemple:

```
$titlu = getfield(getData('stiri.all(0,1)'));
$line = shift(getData('stiri.all(0,1)'));
$id = shift($line);
```

Nu in ultimul rand, pentru a insera, modifica sau sterge linii se poate folosi functia writeData (similara functiei getData):

- (id insert/ true/ false) writeData('(tabel in format phprel).functie', $parametru_functie_solicitata);
- Unde tabelul este un tabel pentru care exista descriere, scris in format phprel (underscore-urile din denumire sunt inlocuite cu puncte), iar functie poate fi:
  - insert – folosita pentru inserarea uneia sau mai multor linii in tabel, $parametru_functie_solicitata fiind un array asociativ reprezentand un rand de inserat (din care cheia primara, daca exista, va fi ignorata – se presupune o cheie primara auto-incrementata, in caz contrar va trebui sa folositi un query obisnuit) sau un array de randuri de inserat (din care, de asemenea, cheia primara va fi ignorata)
  - update – folosita pentru a modifica una sau mai multe linii deja existente, $parametru_functie_solicitata fiind un array asociativ reprezentand un rand sau un array de randuri, linia sau liniile din tabel asupra carora se opereaza modificarile fiind specificate prin cheie primara (ca si componenta in array).
  - mixed – folosita pentru a insera sau modifica linii, dupa cum cheia primara lipseste sau nu, $parametru_functie_solicitata fiind un array asociativ sau un array de randuri
  - delete – folosita pentru a sterge o linie, $parametru_functie_solicitata fiind sau valoarea unei chei primare numerice sau o conditie "where" de stergere.

Exemple:

```
writeData('stiri.insert', array('titlu' => 'phprel - framework de dezvoltare rapida web', 'continut' => '...', 'data' => date('d.m.Y')));
```

- adaugam o noua stire

```
writeData('stiri.insert', getData('stiri.line', 1));
```

- dublam o stire, in cazul de fata cea cu id 1, sau mutam continut dintr-un tabel in altul

writeData('stiri.update', array('id' => 1, 'titlu' => 'phprel – framework de dezvoltare rapida web 3.0')); - modificam o stire, in cazul de fata cea cu id 1
writeData('stiri.delete', $id = 1); sau writeData('stiri.delete', 1); - stergem o stire, in cazul de fata cea cu id 1

Important: in phprel, o stergere se realizeaza recursiv, pentru tabelele descrise de tip parent-child, pentru a nu pastra informatii nefolositoare in tabele. De asemenea, pentru tabelele de tip "linked" legaturile se vor desface (seta la zero) automat. Pentru mai multe detalii, consultati sectiunea 3. phprel in detaliu, subsectiunea 3.2. Accesul la baza de date.

In final, trebuie notat faptul ca in acele situatii in care un query trebuie totusi scris obisnuit, este de preferat ca functiile mysql_query si mysql_insert_id sa fie inlocuite cu functiile run_query si get_insert_id specifice phprel, aceasta metoda oferindu-va cateva avantaje:

- phprel va afiseaza automat query-urile executate in pagina precum si detalii despre ele atunci cand sunteti in modul de dezvoltare
- phprel urmareste query-urile pentru a decide cand pagina dumneavoastra s-a modificat si nu mai poate fi incarcata din cache
- phprel urmareste query-urile si poate opera modificari asupra lor pentru a corespunde indicatiilor dumneavoastra privind identitatea tabelelor (se poate specifica faptul ca un tabel folosit cu o anumita denumire in aplicatie corespunde de fapt unui alt tabel real mysql sau unei parti dintr-un tabel real mysql)
- se poate, in plus, specifica rapid baza de date pe care se executa query-ul respectiv

Pentru detalii suplimentare referitoare la functiile de acces rapid, functiile de control a operatiilor pe baza de date, descrierile de tabel si procesele automatizate in phprel, consultati sectiunea 3. phprel in detaliu, subsectiunea 3.2. Accesul la baza de date. De asemenea, pentru a afla mai multe despre selectarea automata a bazei de date in cazul proiectelor pe mai multe baze de date, precum si alte facilitati oferite de phprel pentru proiectele multi-database, consultati sectiunea 5. Elemente avansate, subsectiunea 5.1. Suport pentru multiple baze de date.

2.3. Web Assistant – interfata web de management a aplicatiei

Pentru phprel, fiecare secunda sau apasare de tasta in minus reprezinta inca un mic succes care contribuie la scopul final al framework-ului de a va ajuta sa castigati (foarte mult) timp. Performanta phprel este in primul rand masurata in orele castigate de dumneavoastra la un proiect. De aceea, orice mic detaliu al procesului de dezvoltare web a fost evaluat si integrat intr-un mod intuitiv si inovativ in phprel. Pentru un plus de ajutor in dezvoltare, aveti la dispozitie o interfata web de management a aplicatiei, care va ajuta de exemplu in scrierea descrierilor de tabel, in adaugarea de tabele sau campuri mysql (pentru a nu mai fi nevoie sa va conectati la aplicatii precum cpanel, plesk sau clienti mysql pentru orice modificare minora), in crearea politicilor de acces la aplicatie, configurarea cache-ului, monitorizarea performantei site-ului dumneavoastra si chiar actualizarea datelor din tabele.

Web Assitant este accesibil adaugand "/phprel" in browser la radacina aplicatiei dumneavoastra. In cazul unei aplicatii noi, veti fi ghidat prin pasii de configurare a conexiunii la baza de date, crearea tabelelor interne phprel si crearea unui cont de acces la Web Assistant.

Pentru a accesa interfata, introduceti utilizatorul si parola stabilite la prima vizita a interfetei. Conturile de acces sunt de asemenea disponibile in tabelul "phprel_back_users" din baza dumneavoastra

de date. Meniurile si mesajele variaza in functie de ce optiuni ati selectat pentru aplicatia dumneavoastra (din cfgs/advanced/optimizers.inc.php – pentru mai multe detalii consultati sectiunea 4. Securitate si performanta, subsectiunea 4.2.3. Optimizarea nucleului prin excluderea componentelor nefolosite) si de varianta folosita: "front" sau "back".

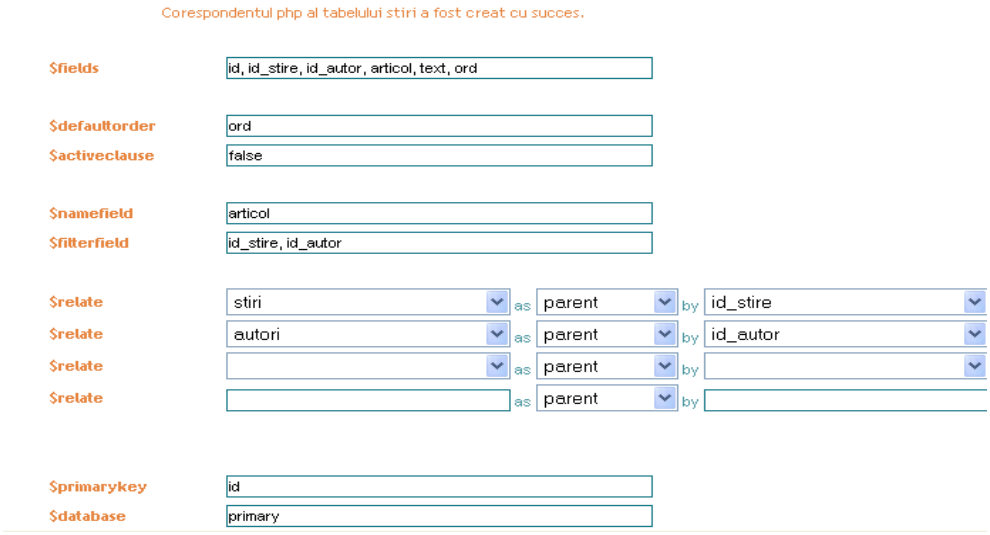
2.3.1. Sectiunea "Baze de date"

In aceasta sectiune a interfetei Web Assistant, se pot crea descrierile de tabel si se pot adauga campuri sau tabele in baza de date.

Web Assistant va ajuta doar sa creati descrieri pentru tabele care nu au deja o descriere. In cazul in care doriti sa modificati o descriere de tabel, editati fisierul corespunzator din web/rules/tables/. In plus, crearea descrierilor nu poate fi realizata daca directorul web/rules/tables/ nu are setate permisiuni de scriere. Web Assistant va va avertiza automat de acest lucru, si va astepta sa setati permisiunile corespunzator, apoi va va afisa o lista de tabele din baza de date care nu au corespondent php.

Pentru a crea descrierile, urmati linkul "tabele (io)", apoi bifati tabelele dorite si apasati "creaza io". Pentru a selecta toate tabelele recomandate de phprel, folositi link-ul "toate (recomandate)". Nu vor fi bifate tabele care par a fi folosite de phprel si nu necesita descriere si tabele care prin denumire indica faptul ca sunt nefolosite (precum "old_*"). Chiar si pentru acestea se pot crea descrieri, bifandu-le dumneavoastra in lista. Ordinea in care tabelele apar in lista este o ordine de creare a descrierilor recomandata de phprel in functie de particularitatile fiecarui tabel.

In timpul procesului de creare a descrierilor, denumirea tabelului pentru care se creaza un io este afisata in subtitlul paginii, iar in cazul in care doriti sa omiteti crearea descrierii pentru tabelul afisat, se poate urma linkul "urmatorul tabel" din explicatia oferita de Web Assistant.



2.3.1.a. Crearea tabelului de "articole" folosit anterior in sectiunea curenta

Web Assistant va intui anumite detalii referitoare la logica tabelului, completand campurile cu o varianta "de pornire" pe care dumneavoastra nu trebuie decat sa o imbunatatiti pentru a reflecta intocmai logica tabelului. In imagine (2.3.1.a.), se observa varianta propusa de Web Assistant pentru tabelul "articole" folosit in subsectiunea anterioara: ordonarea implicita a fost dedusa corect, de asemenea si

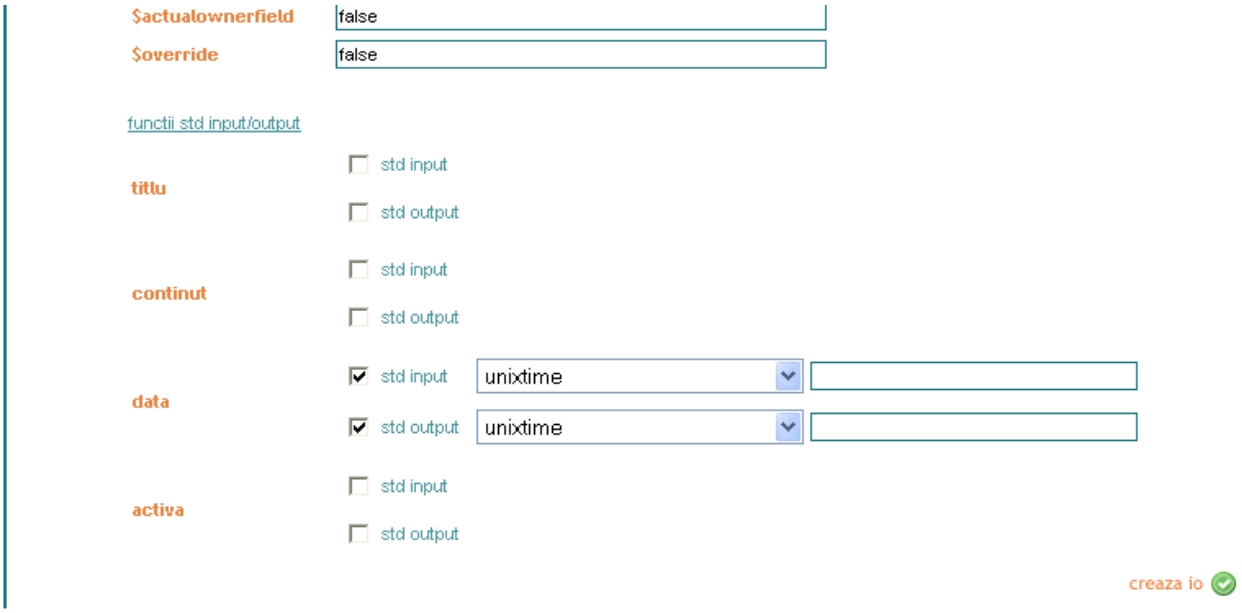
campul de denumire ($namefield) al tabelului, filtrele rapide au fost stabilite pentru campurile "id_stire" si "id_autor", s-a intuit corect legatura tabelului cu tabelele de stiri si autori prin cele doua campuri si s-a completat cheia primara.

Pentru o lista completa de campuri/ variabile care alcatuiesc o descriere de tabel, consultati sectiunea 3. phprel in detaliu, subsectiunea 3.2.1. Descrierea tabelelor, sau editati fisierul web/rules/tables/io.sampletable.php pentru un model de descriere de tabel.

Propunerile facute de Web Assistant se pot modifica sau trece cu valoare "false", dupa caz. Campurile completate in formular vor fi apoi scrise in sintaxa php in fisierul corespunzator tabelului care va fi creat in directorul web/rules/tables/. Valorile pe care le completati trebuie sa aiba in vedere formatul final al fisierului si sintaxa lui (php). In cazul relationarii de alte tabele, vi se da atat posibilitatea completarii unei linii $relate initial vide, dar cu select-uri precompletate cu tabele si campuri probabile, cat si posibilitatea scrierii tabelului si campului si alegerii tipului de relatie. In cazul scrierii denumirii tabelului, aceasta va fi obligatoriu in format phprel (underscore-urile vor fi inlocuite cu puncte).

Important: in cazul aplicatiilor de tip "backend", trecerea parametrului "secure" pe true va asigura un nivel avansat de control al accesului la datele tabelului. Pentru mai multe detalii, consultati sectiunea 5. Elemente avansate, subsectiunea 5.4.3. Restrictionarea accesului la informatiile din tabele. In plus, Web Assistant va adauga pentru dumneavoastra in tabelul mysql campurile necesare asigurarii restrictionarii accesului. Aceasta optiune nu este necesara in mod normal, fiind folosita decat in cazul aplicatiilor cu un nivel de securitate ridicat.

Pentru a adauga functii standard de input/ output, apasati pe "functii std input/ output" si bifati campurile pentru care doriti functii precum si tipul de functie implicita dorit (sau tipul "custom" pentru a scrie dumneavoastra functia).



2.3.1.b. Adaugarea functiilor standard de input/ output pentru campul "data"
al tabelului "stiri" folosit in sectiunea anterioara

Web Assistant va propune implicit un tip de functie in momentul in care bifati std input sau output pentru un anumit camp. Daca presupunerea lui a fost gresita, se poate alege alt tip de functie. Variantele de functie "input" sunt:

- custom – veti scrie dumneavoastra functia in php-ul rezultat, sau daca este o functie dintr-o singura linie, scrieti return-ul corespunzator in campul din dreapta select-ului.
- password – Web Assistant va genera functia necesara introducerii unei parole in baza de date ca hash md5
- unixtime – Web Assistant va genera functia necesara introducerii unei date calendaristice prin conversie in unix timestamp
- currency – Web Assistant va genera functia necesara introducerii in baza de date a unei sume (de bani), din formatul romanesc cu zecimale separate prin virgula in formatul mysql cu virgula mobila, in care zecimalele sunt separate prin punct.
- upload – Web Assistant va genera functia necesara pentru upload-ul unui fisier si memorarea denumirii in campul inserat in mysql. Aceasta functie presupune sau modificari ulterioare din partea dumneavoastra pentru a specifica locatia de upload, sau crearea unui director uploads/(nume tabel)/ in radacina aplicatiei dumneavoastre, cu permisiuni de scriere.
- upload+thumbs – Web Assistant va genera functia necesara pentru upload-ul de imagini si, daca este cazul, generarea automat de thumbnail-uri. In campul aditional se pot specifica formatul thumbnail-urilor, dupa sintaxa:
    - (nume): (dim. X)x(dim. Y) – implicit scale
    - (nume): scale (dim. X)x(dim. Y)
    - (nume): crop (dim. X)x(dim. Y)
    - se pot specifica mai multe thumb-uri, separandu-le prin virgula: (nume): (dim. X)x(dim. Y), (altnume): crop (dim. X)x(dim. Y), (sialtnume): (dim. X)x(dim. Y)

  Thumbnail-urile presupun existenta directoarelor uploads/(nume tabel)/(nume thumb)/

Pentru mai multe detalii referitoare la functiile standard de input/ output, consultati sectiunea 3. phprel in detaliu, subsectiunea 3.2.3. Functii de control, comportament standard, comportamente diferite pe pagina, iar pentru detalii referitoare la upload si thumbnail-uri, consultati sectiunea 3. phprel in detaliu, subsectiunea 3.4. Formulare, 3.4.12. Upload de fisiere si imagini, generarea thumbnail-urilor.

Variantele de functii "output" sunt:

- custom – veti scrie dumneavoastra functia in php-ul rezultat, sau daca este o functie dintr-o singura linie, scrieti return-ul corespunzator in campul din dreapta select-ului
- unixtime – Web Assistant va genera functia necesara afisarii unei date din format unixtime in format "d.m.Y" (implicit) sau formatul specificat prin campul aditional
- currency – Web Assistant va genera functia necesara afisarii unei sume (de bani) din format cu zecimale separate prin punct in format romanesc cu zecimale separate prin virgula.

Important: functiile de input/ output se aplica automat in toate procesele de selectare si/ sau afisare a datelor, in cazul in care nu se interzice acest lucru explicit sau prin metode de selectie "bruta". De aceea configurarea corecta a functiilor este foarte importanta, deoarece veti castiga foarte mult timp la afisare: cu fiecare pagina scrisa care foloseste tabelul pe care tocmai il descrieti, veti castiga timp pentru ca veti selecta datele direct in formatul in care va doriti, fara sa mai fie nevoie sa interveniti local asupra array-ului rezultat.

Exemple de functii std input/ output:

- "custom", cu campul aditional: return strtoupper(substr($val, 0, 1)).strtolower(substr($val, 1, strlen($val)));
- "upload+thumbs", cu campul aditional: small: 100x100, banner: 500x100

Cand descrierea este completa si corecta, apasati "creaza io" pentru a solicita interfetei Web Asssistant sa scrie php-ul corespunzator in directorul de reguli si sa treaca la urmatorul tabel.

Cele doua meniuri de adaugare tabel si adaugare camp pot fi folosite intuitiv sau citind detaliile oferite de Web Assistant in pagina. Tabelele vor fi adaugate la baza de date specificata, iar campurile se vor specifica prin denumire, tip si lungime. Pentru economisirea timpului, unele tipuri au deja incluse lungimi (pentru lungimile frecvente) si nu necesita completarea campului de lungime. Pentru specificarea mai multor campuri la crearea unui tabel, folositi butonul de adaugare "+". In cazul meniului de adaugare a unui singur camp, se poate selecta intre tabelele care au deja descriere, iar campul va fi adaugat atat in baza de date cat si la descriere.

2.3.2. Sectiunea "Traduceri"

In cazul in care ati optat pentru suportul multilingv oferit transparent de phprel, Web Assistant va ajuta sa traduceti template-urile aplicatiei dumneavoastra precum si sa adaugati o noua limba in baza de date. Pentru mai multe detalii referitoare la aplicatii in multiple limbi dezvoltate folosind phprel, consultati sectiunea 5. Elemente avansate, subsectiunea 5.2. Suport pentru multiple limbi.

Sectiunea "continut site" va lista "etichetele" (textele din template-uri care trebuie traduse) impreuna cu traducerea lor in fiecare limba. O eticheta care nu are traducere intr-o anume limba va fi inlocuita pe site cu traducerea ei in limba principala. Pentru a seta limba principala si limbile aditionale, editati cfgs/advanced/core.settings.inc.php, $Tcfg_defaultLanguage si $Tcfg_allLanguages.

Se poate filtra dupa o denumire partiala a etichetelor, dupa pagina sursa (pagina in care a fost folosita pentru prima data eticheta), pagina destinatie (* - reprezinta orice pagina care nu a fost particularizata; o eticheta poate fi tradusa diferit de standard pentru o anume pagina destinatie). De asemenea, se poate lista etichete traduse complet (in toate limbile), traduse incomplet (eticheta nu are traducere cel putin intr-o limba; atentie: nu neaparat in limba selectata) sau toate etichetele.

Actualizarea traducerii unei etichete se realizeaza pentru limba aleasa in filtru. La salvarea unei traduceri, daca nu ati modificat pagina destinatie, traducerea deja existenta va fi actualizata cu noua traducere introdusa, in caz contrar o noua intrare pentru pagina destinatie aleasa va fi adaugata in tabel impreuna cu traducerea specificata (traducerea pentru pagina destinatie initiala, pe care s-a apasat "editare" va ramane neschimbata).

La afisarea in pagina, se va selecta prima traducere gasita, in ordinea:

- traducerea in limba curenta, pentru pagina (destinatie) curenta
- traducerea in limba curenta, pentru orice pagina (*)
- traducerea in limba principala, pentru pagina curenta
- traducerea in limba principala, pentru orice pagina (*)

  Important: etichetele vor fi adaugate automat pe masura ce sunt descoperite de phprel, traducerea initiala fiind chiar textul etichetei si aplicata pentru orice pagina (*), in limba principala

La adaugarea unei noi limbi, Web Assistant va indica tabelele si campurile pentru care se va adauga o noua limba, si va recomanda sa realizati un backup al bazei de date inainte de a continua procedura. O data noua limba stabilita si butonul de "adauga" apasat, Web Assistant va adauga toate campurile necesare si va va comunica rezultatul.

2.3.3. Sectiunea "Acces aplicatie"

In cazul in care dezvoltati o aplicatie de tip "backend" si ati optat pentru restrictionarea accesului pe baza grupurilor de utilizatori, Web Assistant va ajuta sa stabiliti grupurile, utilizatorii, si politicile de acces. Pentru mai multe detalii referitoare la restrictionarea accesului, consultati sectiunea 5. Elemente avansate, subsectiunea 5.4. Grupuri de utilizatori. O data sectiunea parcursa, optiunile puse la dispozitie in Web Assistant vor fi folosite usor, intuitiv sau pe baza explicatiilor puse la dispozitie in paginile Web Assistant.

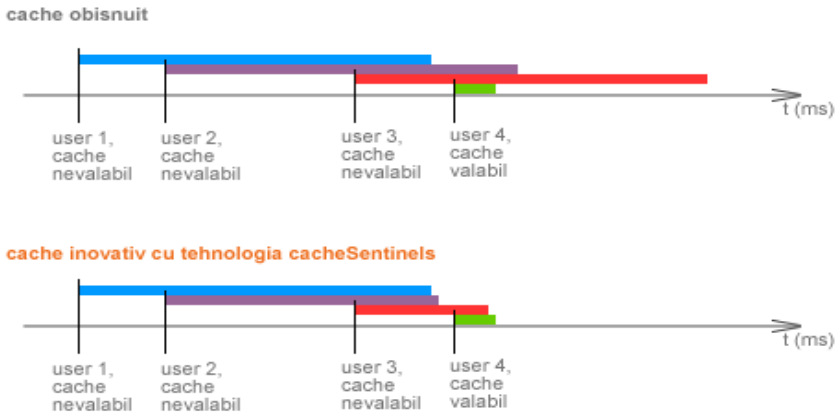
2.3.4. Sectiunea "Cache"

Daca ati optat pentru imbunatatirea performantei folosind tehnologia inovativa de cache adaptiv si cacheSentinels disponibila cu phprel, se poate monitoriza performanta aplicatiei dumneavoastra si configura comportamentul cache-ului prin intermediul interfetei Web Assistant.

In sectiunea "Performanta", dupa alegerea directorului de cache, se pot urmarii timpii necesari generarii paginii in diferite situatii, numarul de incarcari de fiecare tip, timpul mediu de generare si in functie de caz, tendinta de crestere sau imbunatatire a timpului. Directoarele de cache sunt unice pe fiecare sectiune a aplicatiei dumneavoastra (frontend, backend, eventual alte backend-uri) si se pot si configura din cfgs/advanced/engine.settings.inc.php. Se poate cauta dupa denumirea partiala a paginii, sau filtra sa nu apara async-uri (async-urile fara pagina vor aparea catalogate la pagina implicita, "home" sau "main"). In plus, se poate selecta perioada (ziua in curs, ultimele 7 zile, etc.). In functie de perioada, se va calcula tendinta fata de perioada anterioara de lungime similara.

"Hits" reprezinta numarul de afisari din cache (succese), impreuna cu timpul mediu de generare a unei afisari din cache si timpul ocupat de validarea si obtinerea rezultatului din cache. De exemplu: "9 / 0.00957 / 0.0085" inseamna ca pagina a fost afisata din cache de 9 ori, timpul de generare mediu pentru fiecare din cele 9 afisari a fost de aproximativ 9.5 milisecunde, din care 8.5 milisecunde a reprezentat timpul petrecut pentru a include si valida cache-ul precum si pentru a obtine pagina din baza de date.

"Misses" reprezinta numarul de afisari fara cache (esecuri), cand cache-ul nu continea inca pagina ceruta sau continutul s-a constatat a fi nevalabil, impreuna cu timpul de generare a paginii si timpul ocupat de generarea cache-ului (si constatarea initiala a nevaliditatii continutului). De exemplu: "1 / 0.1197 / 0.0138" inseamna ca pagina a fost generata complet o singura data, in aproximativ 120 milisecunde, din care aproximativ 14 milisecunde au fost folosite pentru a determina faptul ca pagina trebuie generata complet si pentru a genera la sfarsit continutul cache-ului.



2.3.4.a. Tehnologia cacheSentinels

"Partial hits" reprezinta numarul de afisari combinate – unul sau mai multe elemente de pagina au fost preluate din cache (in cazul configurarii cache-ului individual pe elemente) in timp ce majoritatea paginii a fost regenerata, impreuna cu timpul mediu de generare a intergii pagini, si timpul ocupat de procesele de caching.

"Sentinel hits" reprezinta numarul de afisari din cache in cazul actionarii automate a tehnologiei cacheSentinels, impreuna cu timpul necesar generarii paginii si timpul ocupat de procesele de caching. Tehnologia cacheSentinels permite aplicatiei dumneavoastra sa raspunda intr-un timp foarte bun chiar si unui numar mare de cereri simultane. In cazul site-urilor cu trafic intens, in situatiile in care mai multi utilizatori acceseaza simultan aceeasi pagina si cu cateva clipe in urma cache-ul a devenit nevalabil, folosind un cache obisnuit se va regenera intreaga pagina pentru toti utilizatorii. Cu cacheSentinels insa, executiile simultane vor comunica intre ele si pagina se va genera complet doar pentru un singur utilizator, ceilalti preluand-o de la el. Se castiga astfel timp si resurse, numarul celor care pot accesa site-ul fara o degradare a performantei crescand semnificativ. Tehnologia este implementata folosind puncte de "frontiera" intre etapele esentiale ale executiei, la fiecare trecere de la o etapa la alta comunicandu-se cu firul de executie cel mai avansat. Daca la punctul de verificare ("santinela") se constata ca se pot prelua date de la un alt fir de executie, etapele urmatoare sunt suspendate iar generarea preia informatiile lipsa. Se pot obtine timpi de executie mai buni cu 30-60%, si o economisire de memorie de 50-75% pentru fiecare fir "aproape simultan" exceptandu-l pe primul.

Pentru site-uri cu trafic mic, cacheSentinels e posibil sa nu fie folosit niciodata sau sa fie folosit foarte rar.

"Imbunatatire" este rezultatul unui calcul efectuat de Web Assistant care ia in considerare numarul de esecuri, numarul de succese complete, partiale sau folosind cacheSentinels si timpii de generare medii pe care ii compara cu timpul de generare al paginii in cazul in care cache-ul ar fi fost dezactivat (se calculeaza un timp real de generare, nu timpul de "miss" pentru ca acel timp include in el si anumite calcule efectuate de cache). In urma acestui calcul, Web Assistant va estimeaza castigul in secunde pentru fiecare noua afisare de pagina (incluzand statistic si cazul unui nou "miss") fata de situatia in care cache-ul ar fi fost dezactivat. Pentru a calcula castigul total, se poate inmulti imbunatatirea cu numarul total de vizite.

"Medie" reprezinta timpul mediu de generare a paginii pe perioada selectata. Implicit, lista este ordonata descrescator dupa medie, pentru a scoate in evidenta cele mai lente pagini, unde se poate eventual optimiza incarcarea.

"Tendinta" va indica cu cat a crescut sau a scazut timpul de generare fata de perioada anterioara de aceeasi lungime cu cea selectata.

In sectiunea "Configurari", se pot stabili politici de caching partial al elementelor de pagina (prin introducerea denumirii elementului, a tipului de cache – care insa este restrictionat doar la cache-ul static bazat sau pe numar de afisari sau pe timp de valabilitate si particularizarea valabilitatii). In cazul in care pe o pagina cache-ata, se constata la un moment dat ca trebuie actualizat cache-ul, elementele de pagina care se regasesc pe lista de caching partial vor fi (in masura in care conditiile introduse de dumneavoastra pentru fiecare element sunt valabile in respectiva situatie) "injectate" direct in pagina, renuntandu-se la portiunile de cod care calculeaza acel element si economisindu-se timp, resurse si acces la baza de date.

In plus, se pot specifica pagini care nu respecta setarea globala de caching, specificandu-se individual tipul de cache si parametrii pentru fiecare pagina. Tipurile de cache sunt:

- statice:
    - un numar fix de vizite stabilit prin "Expira la"
    - un numar fix de minute stabilit prin "Expira la"
    - un numar fix de ore stabilit prin "Expira la"
    - un numar fix de zile stabilit prin "Expira la"
- adaptive:
    - maxim F vizite stabilite prin "Expira la" cu monitorizarea activa a modificarilor efectuate in baza de date. Cache-ul va fi actualizat in situatia in care o valoare din baza de date folosita pe pagina este schimbata sau stearsa, precum si in situatia in care sunt introduse noi valori in baza de date de care pagina actuala va depinde. Daca nu este descoperita o astfel de modificare, cache-ul este valabil F vizite, dupa care este actualizat.
    - maxim F vizite stabilite prin "Expira la" cu monitorizarea interna a modificarilor efectuate asupra bazei de date . Cache-ul va fi actualizat in situatia in care prin intermediul aplicatiei dumneavoastra (in frontend sau oricare backend) a fost modificata sau stearsa o valoare de care depinde pagina, precum si in situatia in care au fost introduse prin intermediul aplicatiei dumneavoastra valori de care pagina actuala va depinde. Daca nu este descoperita o astfel de modificare, cache-ul este valabil F vizite, dupa care este actualizat.
    - cache de tip "circuit inchis" care teoretic poate sa nu expire niciodata, cu monitorizarea activa a bazei de date. Cache-ul va fi actualizat doar in cazul descoperirii unei modificari in baza de date care afecteaza pagina.
    - cache de tip "circuit inchis" cu monitorizarea interna a bazei de date. Cache-ul va fi actualizat doar in cazul descoperirii unei modificari care afecteaza pagina, efectuate prin intermediul aplicatiei asupra bazei de date.

  Important: in cazul cache-ului adaptiv de tip maxim F, dupa un esec sau un succes (s-au atins F vizite fara descoperirea unei modificari), cache-ul se va adapta, calculand automat un nou F pentru ciclul urmator, mai mic in cazul unui esec, mai mare in cazul unui succes. Procentul cu care va creste F este specificat prin primul camp al "Adaptarii", iar procentul la care scade F este specificat prin al doilea camp (sau poate fi determinat automat).

Cea mai dinamica varianta de cache, care va descoperi cu cea mai mare precizie modificarile si va mentine continutul actualizat cu cea mai mare probabilitate este cache-ul adaptiv cu maxim F vizite si monitorizarea activa a bazei de date. Un cache foarte performant indicat pentru folosirea in medii "live",

este cache-ul adaptiv cu maxim F vizite si monitorizarea interna a bazei de date sau, dupa caz, cache-ul de tip "circuit inchis" cu monitorizarea interna a bazei de date.

Pentru a sterge complet continutul tabelelor de caching, indiferent de directorul de cache sau de sectiunea aplicatiei (frontend, backend), folositi sectiunea "Reset".

Pentru mai multe detalii referitoare la tipurile de cache, configurari, alte conditii care determina actualizarea cache-ului si strategii de alegere a cache-ului potrivit, consultati sectiunea 4. Securitate si performanta, 4.2.1. Caching adaptiv si tehnologia cacheSentinels.

2.3.5. Sectiunea "Mentenanta"

Web Assistant va pune la dispozitie o sectiune de intretinere a aplicatiei, prin intermediul careia se poate actualiza baza de date fara un client mysql, se pot sterge informatiile invechite din tabelele phprel si se poate activa modul de urgenta de afisare exclusiva din cache.

Tabelele phprel stocheaza informatii referitoare la cache, pastrarea datelor de la o sesiune la alta, validarea url-urilor si suportul pentru multiple limbi. Anumite inregistrari sunt generate automat sau indirect la cereri realizate de aplicatia dumneavoastra, iar in timp unele devin invechite, ne mai fiind folosite. Din Web Assistant, liniile nefolositoare pot fi sterse automat, prin folosirea functiilor de "Actualizare" sau "Actualizare agresiva". Indicatorul afisat va arata numarul estimat de linii care urmeaza sa fie sterse in modul "obisnuit" respectiv "agresiv" precum si varia numarului total de linii din tabelele temporare in ultimele doua zile. O variatie pozitiva mare, exceptand cele din prima saptamana de la lansarea site-ului si situatiile in care se dezvolta noi sectiuni pe site, inseamna o generare intensiva de linii temporare care poate fi un indiciu privind un atac asupra site-ului sau o configurare necorespunzatoare a cache-ului.

Actualizarea nu este necesara decat dupa o perioada foarte mare de timp (de exemplu, atunci cand indicatorul arata un castig de cateva zeci de mii de linii) iar site-ul va functiona in parametrii si fara aceasta actualizare. In plus, in functie de tipul de aplicatie dezvoltat, unele date temporare ar putea fi inca folosite. Actualizarea "obisnuita" va sterge toate liniile temporare mai vechi de 14 zile, in timp ce cea agresiva va sterge toate liniile mai vechi de 2 zile. Cea de-a doua varianta este recomandata doar in cazul in care tabelele s-au "aglomerat" intr-o perioada scurta de timp, cauzand o degradare vizibila a performantei aplicatiei si este recomandat ca aceasta actualizare sa fie insotita de masuri aditionale, cum ar fi reconfigurarea cache-ului sau prevenirea acceselor rau intentionate.

Sectiunea de "Actualizare date" va asigura accesul rapid la baza de date. Selectati un tabel din lista (tabelul trebuie in mod obligatoriu sa aiba descrierea corespunzatoare in php, pentru detalii revedeti sectiunea 1. phprel la prima vedere, subsectiunea 1.7. Descriere simpla de tabel), si apasati "cauta". Vor fi afisate toate liniile tabelului cu posibilitatea de filtrare, denumirile coloanelor si valorile fiind afisate intutiv de Web Assistant asa cum probabil doriti sa le vizualizati. De exemplu, daca un tabel "articole" este relationat de un tabel "stiri" si un tabel "autori" (precum in exemplul din sectiunea curenta, subsectiunea anterioara), Web Assistant va afisa coloanele sub denumirea de "Stire" si "Autor" (in loc de "id_stire" si "id_autor") iar valorile vor fi titlul stirii respectiv numele autorului. Se pot, de asemenea, edita, sterge sau adauga linii, iar editarea se va face in acelasi mod intuitiv (selectand de exemplu titlul stirii relationate dintr-un select cu stiri, in locul introducerii unui id).

Important: Pentru anumite relationari mai complexe, Web Assistant s-ar putea sa nu intuiasca formatul asa cum ar fi de dorit. In plus, filtrul functioneaza doar pe campurile tabelului selectat, si nu vor functiona pe valorile afisate automat de Web Assistant din tabelele relationate.

Sectiunea "Afisare din cache" va permite functionarea site-ului exclusiv din cache, indiferent de validitatea continutului. Solutia este disponibila pentru situatii critice in care continutul surselor site-ului a fost modificate neautorizat sau urmeaza sa realizati o schimbare pe parcursul careia utilizatorii vor fi afectati. Web Assistant va da si posibilitatea introducerii unui mesaj de eroare care va fi afisat in cazul accesarii unor resurse nedisponibile in cache, si de indata ce situatia s-a remediat se poate comuta inapoi in modul "live" printr-un simplu click.

2.4. Cereri asincrone javascript + xml (A.J.A.X.)

Pentru a asigura o experienta fluenta a utilizatorilor pe site-ul dumneavoastra, deseori va trebui sa incarcati sau schimbati continut pe pagina prin cereri ulterioare (asincrone), realizate in fundal, catre server. Si pentru a asigura o experienta unica a utilizatorilor pe site-ul dumneavoastra, phprel a redefinit modul in care veti scrie codul din spatele acestor cereri.

O cerere A.J.A.X. este preluata de nucleul phprel si trimisa unui php responsabil de respectiva cerere ("handler"), in cadrul caruia raspunsul este construit usor sub forma unor indicatii privind modificarile care trebuie sa survina in pagina, apoi acel raspuns este trimis inapoi nucleului phprel care il scrie in format xml si il trimite browser-ului. La destinatie, raspunsul xml este despachetat de o extensie javascript a nucleului phprel, care interpreteaza raspunsul si realizeaza modificarile in pagina. Ar ramane de rezolvat doua operatii: construirea si integrarea in pagina a cererii propriu-zise (sub forma de cod javascript) si scrierea "handler"-ului. Pentru prima operatie, phprel oferta un suport concis si puternic, cererea fi integrata in pagina tastand doar cateva indicatii (sau in anumite cazuri, fara nici o modificare a paginii). A doua operatie trebuie scrisa in totalitate de dumneavoastra, dar stilul inovativ si suportul oferit de phprel in scrierea anumitor situatii tipice intalnite in php face ca un handler de complexitate medie sa fie scris in doua-trei minute.

Continuam mini exemplul inceput in sectiunea anterioara, adaugand la home.html un nou link (eventual in acelasi din cu linkul "Vezi stirea" construit anterior):

```
<br /><a href="articole/edit/0">Adauga articol</a>
```

Si creem pagina de adaugare a unui articol:

- creem articole.php (in pages), fara continut, si articole.html (in templates), cu urmatorul continut:

```
<&frm.form>
<table>
        <tr>
                <td class="text_negru">Adauga pentru stirea:</td>
                <td>
                        <input type="text" name="id_stire" id="id_stire" value=""
onkeyup="<&async{getstire.id_stire.id_stire}>" />
                </td>
        </tr>
        <tr>
```

```
                <td class="text_negru">Stiri posibile:</td>
                <td>
                        <div id="feedback" class="text_verde_2">
                                -
                        </div>
                </td>
        </tr>
</table>
<&endform>
```

- pentru campul "id_stire" am construit o cerere asincrona folosind tag-ul <&async{...}>, tag care respecta formatul tag-urilor speciale phprel, precizat in sectiunea anterioara. Creem acum handler-ul specificat in tag, getstire.php, in directorul web/handlers/, cu urmatorul continut:

```
/* input */

$call = $_GET['call']; //expected call is typed data
$target = $_GET['target']; //expected target is data field

/* end input */


/* processing */

if ($call)
{
        $d = fetch(select('titlu', 'stiri', "titlu LIKE '".$call."%'"));
        if ($d && !is_array($d))
                $flds[$target] = $d;
        else
        if (is_array($d))
        {
                $divs['feedback'] = '';
                foreach ($d as $i => $stire)
                        $divs['feedback'] .= '<a href="javascript: void(0);"
onclick="javascript:
document.getElementById(\".$target.'\').value=\".$stire['titlu'].'\"'>'.$stire['titlu'].'</a><br
/>';
                }
                else
                        $divs['feedback'] = '-';
        }
        else
                $divs['feedback'] = '-';
```

```
/* end processing */


/* output */

$content = $divs;
$displays = array();
$selects = array();
$fields = $flds;
$calls = array();

/* end output */
```

Handler-ul este "apelat" cu doi parametrii, primul denumit "caller", in cazul de fata fiind "id_stire", iar al doilea denumit "target", in cazul de fata fiind, de asemenea, "id_stire". Astfel, s-a construit intreaga cerere: getstire.id_stire.id_stire. Se cere handler-ul "getstire", caller-ul fiind "id_stire" si target-ul fiind "id_stire". Handler-ul va primi doi parametrii: un "call", valoarea caller-ului, si un "target" identic cu target-ul specificat. Logica provine de la un model simplificat de cerere A.J.A.X. in care valoarea ("call") unui element din pagina ("caller") determina modificarea (generata in handler) unui alt element de pagina ("target"). Aceasta insa ramane doar o conventie, un handler fiind capabil sa modifice oricate elemente din pagina, iar cererea poate fi trimisa simultan cu valorile mai multor elemente. Pentru detalii, consultati sectiunea 3. phprel in detaliu, subsectiunea 3.7. Cereri asincrone javascript + xml (A.J.A.X.). In cazul de fata, caller-ul fiind "id_stire" (elementul html din pagina cu id="id_stire"), valoarea trimisa handler-ului prin "call" va fi ceea ce dumneavoastra introduceti in camp. Target-ul va fi acelasi camp "id_stire", folosit pentru a completa numele stirii pe baza primelor caractere introduse de dumneavoastra.

Pentru a ilustra modul de functionare al handler-ului, asigurati-va ca aveti trei stiri in tabelul de "stiri" care incep cu "phprel", de exemplu: "phprel", "phprel - framework de dezvoltare rapida web 3.0" si "phprel - un nou concept". Introduceti apoi in camp valoarea "php" si asteptati cateva momente. La "Stiri posibile" au aparut cele trei stiri. Un click pe una dintre stiri va completa campul "id_stire" cu denumirea respectiva. Continuati sa scrieti in camp, pentru a obtine valoarea "phprel - f". Campul va fi completat automat cu denumirea "phprel – framework de dezvoltare rapida web 3.0", singura stire care se potriveste caracterelor deja introduse. Ati obtinut astfel un mecanism simplu de autocompletare a unor campuri cu valori din baza de date. Acesta este doar un exemplu demonstrativ si poate fi imbunatatit.

Un handler contine trei parti, o sectiune de input, o sectiune de procesare a cererii si o sectiune de raspuns. In general, prima sectiune este standard. Pentru a construi rapid un handler, duplicati fisierul samplehandler.php, denumiti-l corespunzator si modificati-l pentru a indeplini sarcina propusa. In sectiunea de "input", se obtin parametrii cererii: call-ul si target-ul. In sectiunea de "output", se construieste raspunsul sub forma unor array-uri:

- $content – array de forma id_div => continut prin care se schimba continutul unuia sau mai multor div-uri.
- $displays – array de forma id_div => true/ false prin care se schimba vizibilitatea unuia sau mai multor div-uri in pagina (concret, phprel foloseste style.display = 'block' respectiv style.display = 'none')

- $selects – array de forma id_select_box => optiuni prin care se schimba optiunile unuia sau mai multor select-uri. Pentru detalii privind modul de construire al optiunilor, consultati sectiunea 3. phprel in detaliu, subsectiunea 3.4.7. Construirea optiunilor unui select.
- $fields – array de forma id_camp => valoare prin care se schimba valoarea unuia sau mai multor campuri din pagina.
- $calls – array de forma nume_functie => valoare parametru, prin care se apeleaza diverse functii (care in mod necesar au un unic parametru sau nu au nici un parametru caz in care "valoare parametru" va fi completat cu sirul vid) in momentul in care rezultatul cererii asincrone s-a intors si a fost procesat

Sectiunea de procesare realizeaza, evident, construirea raspunsului pe baza datelor de intrare. In cazul de fata, daca s-a introdus o valoare in camp, se selecteaza stirile care incep cu valoarea introdusa. In cazul in care a rezultat o singura stire, campul specificat prin "target" se completeaza cu titlul gasit. Se observa metoda de realizare a array-ului $fields: se construieste un array temporar, de exemplu denumit $flds, care apoi este atribuit in sectiunea de "raspuns" variabilei $fields. Aceasta metoda pastreaza bine delimitata si clar structurata (sub forma a patru linii) sectiunea de raspuns. In cazul in care s-au gasit mai multe stiri, se schimba continutul div-ului "feedback" cu o "lista" de stiri posibile pe care se poate "apasa" pentru a completa campul. Metoda este aceeasi: se construieste un array temporar $divs care este apoi atribuit variabilei $content. In orice alt caz, se schimba continutul div-ului "feedback" in "-".

De notat faptul ca am folosit tag de <input> si nu <&frm.(nume)> doar in scop demonstrativ pentru a ilustra functionarea tag-ului <&async{...}> care este inlocuit cu o secventa de cod javascript. In locul tag-ului <input type="text" name="id_stire" id="id_stire" value="" onkeyup="<&async{getstire.id_stire.id_stire}>" /> se poate scrie: <&frm.id_stire{text; async="onkeyup: getstire.id_stire"}>. Pentru mai multe detalii, consultati sectiunea 3. phprel in detaliu, subsectiunea 3.4.10. Formulare si cereri asincrone javascript + xml (A.J.A.X.).

Continuam prin construirea unui handler care modifica optiunile unui select-box:

- modificam articole.html, adaugand la formular urmatorul continut:

```
<tr>
        <td class="text_negru">Agentie:</td>
        <td><&frm.agentie{options="$agentii"; async="getautor.id_autor"}></td>
</tr>
<tr>
        <td class="text_negru">Autor:</td>
        <td><&frm.id_autor></td>
</tr>
<tr>
        <td class="text_negru">Titlu:</td>
        <td><&frm.articol></td>
</tr>
<tr>
        <td class="text_negru">Continut:</td>
        <td><&frm.continut></td>
</tr>
<tr>
        <td class="text_negru">Ordine:</td>
```

```
            <td><&frm.ord{text; defaultvalue="$ord"}></td>
        </tr>
        <tr>
            <td> </td>
            <td><&frm.goPost><&frm.submit{value="Adauga"}></td>
        </tr>
```

- modificam articole.php, adaugand continutul:

```
        $d = fetch(select('reprezentant_al', 'autori', '1', 'reprezentant_al', 'reprezentant_al'),
    true);
        for ($i=0; $i<count($d); $i++)
            addOpt($d[$i]['reprezentant_al'], $d[$i]['reprezentant_al']);
        $agentii = addEmptyopt(retOpt());

        $ord = fetch(select('MAX(ord)', 'articole', '1'))+1;
```

- Pentru a ilustra folosirea cererilor asincrone pentru a repopula un select-box in functie de context, ne imaginam ca vrem sa simplificam alegerea unui autor pentru un articol, selectand mai intai revista sau agentia pentru care el scrie, reprezentata prin campul "reprezentant_al". Adaugam deci la formular un camp "agentie" cu optiunile calculate in php printr-un query si folosind functiile de construire manuala a optiunilor unui select-box (pentru mai multe detalii, consultati sectiunea 3. phprel in detaliu, subsectiunea 3.4.7. Construirea optiunilor unui select). Adaugam apoi pe campul "agentie" o cerere asincrona catre handler-ul "getautor" avand target-ul "id_autor" (campul de selectie al autorului). De notat faptul ca nu am folosit sintaxa de async clasica: handler.caller.target, ci o sintaxa simplificata: handler.target. In acest caz se subintelege caller-ul ca fiind campul pe care s-a adaugat cererea asincrona. In final, scriem un query pentru a obtine cea mai probabila valoare pentru campul "ord" de ordonare a articolelor, valoare retinuta in variabila $ord si specificata ca valoare intiala pentru campul <&frm.ord>.
- Creem handler-ul getautor.php in web/handlers/, cu continutul:

```
    /* input */

    $call = $_GET['call']; //expected call is "reprezentant_al"
    $target = $_GET['target']; //expected target is "id_autor" select-box

    /* end input */


    /* processing */

    $autori = addEmptyOpt(getData('autori.searchoptions', "reprezentant_al='".$call."'"));

    /* end processing */


    /* output */
```

```
$content = array();
$displays = array();
$selects = array($target => $autori);
$fields = array();
$calls = array();

/* end output */
```

- Dupa un refresh al paginii, formularul este functional si poate adauga articole in baza de date (cu exceptia faptului ca nu completeaza corect "id_stire", din cauza faptului ca acel camp ar trebui sa fie un id si nu o denumire de stire asa cum este in acest exemplu).
- Selectati o agentie pentru a restrange autorii doar la agentia respectiva. De notat faptul ca phprel a completat in mod implicit select-box-ul "id_autor" cu autorii din baza de date (si chiar a determinat tipul campului ca fiind select). Pentru mai multe detalii privind modul in care phprel va ajuta sa construiti si sa populati select-box-uri, precum si situatiile de tip parent-child in care phprel realizeaza automat popularea prin A.J.A.X. fara nici o linie de cod, consultati sectiunea 3. phprel in detaliu, subsectiunea 3.4.6. Campuri select de tip parent-child.

Vom solutiona simplu problema "id_stire", printr-o functie standard de input:

- modificam web/rules/tables/io.articole.php, adaugand continutul:

```
function std___input___articole___id_stire ($val)
{
	$d = getfield('id', getData('stiri.line', $val));
	if ($d != $val)
		return getfield('id', getData('stiri.search', "titlu='".$val."'"));
	else
		return $val;
}
```

- Notam de asemenea faptul ca exemplul a fost unul menit sa ilustreze cererile asincrone, ar fi fost mai practic in cazul de fata sa scriem tag-ul pentru "id_stire" la modul: <&frm.id_stire> si sa alegem stirea dintr-un select-box construit automat de phprel, fara sa mai fie nevoie de functia standard de input.

Ultimul exemplu este incarcarea unei pagini folosind A.J.A.X:

- modificam header.html, adaugand pe ultima linie continutul:

```
mini exemplu phprel
```

- Acest text serveste drept portiune a site-ului care nu se modifica, pentru a sesiza schimbarea paginii doar in zona de site destinata "continutului". In situatiile reale, zonele care raman neschimbate vor fi, evident, mult mai ample.
- Modificam home.html, schimband linkul vechi de "Vezi stirea" - Vezi stirea cu un link construit prin intermediul unei variabile de template corelata cu o functie php disponibila in phprel (functia a()):
  - <&a(stire/phprel)>Vezi stirea
- Reincarcam pagina "home" si urmam linkul "Vezi stirea" pentru a vedea rezultatul.
- In situatia in care incarcarea paginii dureaza mai mult decat un timp prestabilit (configurat standard la 100ms dar editabil din cfgs/advanced/core.settings.inc.php), un gif animat de

"incarcare" va aparea pana la incarcarea continutului, altfel pagina va fi incarcata direct, ca si cum era deja disponibila pe calculatorul client. Astfel, utilizatorii care dispun de o conexiune la internet foarte rapida vor naviga fara intreruperi, in timp ce utilizatorii care folosesc conexiuni mai lente vor fi atentionati de faptul ca pagina inca se incarca.

Pentru mai multe detalii referitoare la cererile asincrone, consultati sectiunea 3. phprel in detaliu, subsectiunea 3.7. Cereri asincrone javascript + xml (A.J.A.X.).

2.5. Cos de cumparaturi si preluarea comenzilor

In phprel, suportul pentru realizarea unui cos de cumparaturi pe site-ul dumneavoastra a fost redefinit, majoritatea operatiilor fiind indicate direct din template si realizate automat fara a fi necesara vreo linie de cod php. Datorita descrierilor de tabel si flexibilitatii inerente modulelor phprel, sistemul de cos de cumparaturi poate fi interconectat cu orice tabel de utilizatori si orice tabel de produse.

Pentru exemplul urmator, va trebui sa modificati tabelul deja creat de phprel, "produse" (tabelul a fost creat la accesarea interfetei web de management a aplicatiei, Web Assistant), adaugand campurile denumire (varchar) si pret (float). Introducem apoi date de test in tabelele folosite:

```
INSERT INTO front_users (user, pass, status) VALUES('demo', MD5('demo'), 1)
INSERT INTO produse (denumire, pret) VALUES('un produs', '100'), ('alt produs', '230')
```

Continuam prin crearea unei noi pagini, denumita "produse", astfel:

- creem in web/pages/ fisierul produse.php, fara continut
- creem in web/templates/ fisierul produse.html, cu urmatorul continut:

```
<table width="80%">
   <tr>
            <td class="text_verde_2">
                  <&if{!getUser()}>
                  <&if{$Tshp_cartLoginError}><div class="text_verde_2">User si/sau
parola incorecte.</div><&endif>
                  <&frm.form.login>
                  <table>
                        <tr>
                              <td class="text_negru">User:</td>
                              <td><&frm.user{text}></td>
                        </tr>
                        <tr>
                              <td class="text_negru">Parola:</td>
                              <td><&frm.pass{pass}></td>
                        </tr>
                        <tr>
                              <td> </td>
                              <td><&frm.goLogin>
<&frm.submit.login{value="Login"}></td>
                        </tr>
                  </table>
```

```
                <&endform>
                <&else>
                        Bun venit, <&getUserDetails('user')>! <a
href="home/logout/1">Logout</a>
                <&endif>
        </td>
        <td>
                <table>
                        <&loop "produse">
                        <tr>
                                <td class="text_negru"><&denumire>, <&pret> EUR,
<a href="produse/<&cart.add{<&id>}>">adauga in cos</a></td>
                        </tr>
                        <&endloop>
                </table>
        </td>
  </tr>
  <tr>
        <td class="text_negru" valign="top">
                <&if{cartItems()}>
                        Aveti <&cart.items> <&if{cartItems() ==
1}>produs<&else>produse<&endif> in cos, in valoare de <&cart.total> EUR.
                <&else>
                        Nu aveti produse in cos.
                <&endif>
        </td>
        <td class="text_negru">
                <&if{cartItems()}>
                        <table width="100%">
                                <&loop "cart.contents">
                                <tr>
                                        <td
class="text_negru"><&product['denumire']></td>
                                        <td class="text_negru"><&quantity></td>
                                        <td class="text_verde_2"
align="right"><&totalline></td>
                                        <td class="text_negru"><a
href="produse/<&cart.more{<&product['id']>}>">+</a> <a
href="produse/<&cart.less{<&product['id']>}>">-</a> <a
href="produse/<&cart.remove{<&product['id']>}>">sterge</a></td>
                                </tr>
                                <&endloop>
                                <tr>
                                        <td> </td>
```

```
                                        <td> </td>
                                        <td class="text_verde_2"
        align="right"><&cart.total></td>
                                        <td class="text_verde_2">EUR</td>
                                    </tr>
                                </table>
                            <&else>
                                 
                            <&endif>
                        </td>
                    </tr>
                </table>
```

- modificam home.html, adaugand in ultimul div (dupa linkul de "Adauga articol"), continutul:
  ```
  <br /><a href="produse">Vezi produsele</a>
  ```
- Incarcam pagina "home" si urmam linkul "Vezi produsele" pentru a vedea rezultatul.

Pagina de produse contine acum: un formular de login pentru utilizatorii inregistrati ai magazinului, o lista cu toate produsele din baza de date care se pot "adauga in cos", un "mini-cos" care specifica numarul de produse si valoarea totala si o lista a produselor din cos.

Urmarind sursa, veti descoperi o serie de elemente noi care va ajuta sa construiti rapid un magazin online:

- functia getUser() intoarce id-ul utilizatorului logat
- variabila globala $Tshp_cartLoginError este (in urma unui proces de login automat) 1 daca utilizatorul sau parola sunt incorecte sau 2 daca respectivul cont nu a fost inca activat (corfirmed = 0).
- campul hidden special "goLogin", similar lui "goPost", care solicita nucleului phprel sa logheze user-ul trimis prin POST.
- functia getUserDetails() care intoarce, in lipsa unui parametru, un array asociativ cu linia din tabel corespunzatoare utilizatorului logat, sau campul specificat ca parametru de pe linia respectiva
- linkul cu parametru $_GET['logout'] = 1, in cazul de fata "produse/logout/1", care solicita nucleului phprel sa delogheze utilizatorul curent
- functia cartItems() care intoarce numarul de produse din cos
- tag-urile speciale <&cart.(functie){...}>, care respecta formatul tag-urilor speciale phprel, descris in sectiunea anterioara, cu urmatoarele utilizari:
  - <&cart.items> - este inlocuit cu numarul de produse din cos
  - <&cart.total> - este inlocuit cu suma totala de plata pentru produsele din cos
  - <&cart.add{(id_produs)[, (cantitate)]}> - este inlocuit cu o parte de URL care va solicita nucleului phprel sa adauge produsul cu id-ul specificat in cos. Optional, se poate preciza si cantitatea. In cazul in care produsul exista deja in cos, cantitatea va fi crescuta.
  - <&cart.more{(id_produs)[, (cantitate)]}> - este inlocuit cu o parte de URL care va solicita nucleului phprel marirea cantitatii din cos a produsului specificat prin id. In cazul in care cantitatea nu este specificata, marirea se va realiza cu o unitate. Produsul trebuie sa existe deja in cos pentru ca operatiunea sa aiba loc.

        - <&cart.less{(id_produs)[, (cantitate)]}> - similar cu <&cart.more{...}> dar solicita scaderea cantitatii din cos a produsului specificat.
        - <&cart.remove{(id_produs)}> - similar celorlalte tag-uri de acest tip, dar solicita scoaterea (stergerea) produsului specificat prin id din cos.
    - bucla cu denumire speciala "cart.contents", care este automat inlocuita cu produsele din cos. Liniile si campurile selectate sunt cele apartinand tabelului cart_sessions (sau echivalentului lui specificat in cfgs/advanced/core.settings.inc.php), si datele produselor, disponibile pentru fiecare linie in array-ul asociativ "product".

Cosul de cumparaturi ofera de asemenea suport pentru adaugarea automata a datelor de livrare si a comenzilor in tabelele corespondente, precum si afisarea comenzilor efectuate de un utilizator. Pentru mai multe detalii, consultati sectiunea 3. phprel in detaliu, subsectiunea 3.8. Cos de cumparaturi si preluarea comenzilor.

In plus, am folosit o serie de elemente deja prezentate, insa de data aceasta in contexte mai complicate:

- functii in cadrul conditiei unui <&if{...}>. Se poate folosi orice expresie in limbaj php in interiorul conditiilor unui if, dar cu anumite limitari introduse in principal pentru pastrarea organizarii logice a framework-ului. De exemplu, variabilele folosite sunt implicit considerate globale, si nu se poate folosi variabila $GLOBALS, in plus, doar o lista restransa de functii php pot fi folosite in conditii si nu se permite operatorul de egalitate – pentru a asigura pastrarea oricaror calcule in componenta php a paginii. Pentru detalii, consultati sectiunea 3. phprel in detaliu, subsectiunea 3.3.5. Tag-uri conditionale.
- un form secundar, <&frm.form.login>. Pentru diferentele dintre form-ul principal si form-uri secundare, consultati sectiunea 3. phprel in detaliu, subsectiunea 3.4.4. Formular principal si formulare secundare.
- o bucla fara array-ul corespunzator in php, <&loop "produse">. In cazul in care bucla nu are un corespondent in php si nu este o bucla speciala (de exemplu, de afisare a continutului cosului de cumparaturi), phprel isi va da seama daca numele buclei este un nume de tabel. In caz afirmativ, phprel va incerca restrangerea, pe baza parametrilor GET sau POST, a liniilor din tabel care probabil se doresc a fi afisate. Daca nici o conditie nu se potriveste tabelului, similar cazului de fata, toate liniile tabelului vor fi afisate.
- tag-uri reprezentand variabile globale sau componente ale unei bucle, care specifica valoarea unei chei a unui array. In cazul de fata, <&product['id']> inseamna inlocuirea valorii cheii "id" a cheii "product" a liniei curente din &loop. Sintaxa este identica unei sintaxe php corespunzatoare. Se poate, prin urmare, afisa o variabila globala de forma <&variabila['cheie 1'][5]['cheie 2']['cheie 3']>.
- tag-uri imbricate, de exemplu <&cart.more{<&product['id']>}>. In acest caz, se inlocuiesc tag-urile progresiv, din interior spre exterior, pana la rezolvarea tuturor tag-urilor. Sintaxa ar fi similara unei situatii din php de forma: $cartmore[$product['id']]. Se pot folosi tag-uri imbricate in orice situatii, inclusiv in <&if{...}>-uri. De exemplu, in interiorul unui &loop are sens sa folosim <&if{<&product['id']> == 5}>.
  Important: variabilele imbricate care ar putea intoarce valori sir de caractere ar trebui delimitate prin apostroafe. De exemplu, <&if{'<&product['denumire']>' == 'un produs'}>.

La autentificarea cu un user si o parola, sesiunea de cos de cumparaturi este implicit inregistrata ca apartinand respectivului utilizator. In plus, la logout, produsele adaugate in cos nu vor mai fi disponibile (sesiunea precedenta se considera a fi privata, nedisponibila public dupa logout).

Pentru mai multe detalii despre recuperarea unei sesiuni de cos la login, adaugarea discount-urilor sau a altor taxe/ bonus-uri precum si alte detalii privind sistemul de cos de cumparaturi, consultati sectiunea 3. phprel in detaliu, subsectiunea 3.8. Cos de cumparaturi si preluarea comenzilor.

3. phprel in detaliu

In sectiunile precedente, am construit impreuna un mini exemplu prin care dumneavoastra ati facut cunostinta cu anumite concepte centrale din phprel. Pentru a va familiariza mai mult cu phprel si a afla multe alte mijloace prin care phprel va ajuta in dezvoltarea web, continuati cu sectiunea curenta si sectiunile 4. Securitate si performanta, 5. Elemente avansate in care fiecare detaliu implementat in phprel este explicat pe larg. Alternativ, daca ati mai folosit phprel sau ati mai parcurs sectiunile detaliate intr-o ordine oarecare, aveti la dispozitie sectiunea 6. Index, care enumera exhaustiv si explica pe scurt fiecare "entitate" din phprel.

3.1. Nucleul phprel

In centrul framework-ului se afla o serie de module interconectate care sunt specializate sa va ofere un anumit tip de ajutor. Aceste module sunt construite folosind algoritmi rapizi sau foarte rapizi (de tip n patrat sau n), care asigura un timp de executie foarte bun chiar daca phprel trebuie sa faca foarte multe calcule suplimentare fata de orice framework tipic, calcule ce servesc la a va oferi un ajutor real in dezvoltarea aplicatiei propuse. In plus, codul phprel este compact, minimizand problema de performanta cauzata de includerea de cod php intr-o aplicatie (in php, avand in vedere tipul de limbaj – limbaj de "scripting", interpretat – si pasii prin care o sursa php este inclusa – validare sintactica, transformare in "opcodes", executie – includerea surselor mari se face intr-un timp relativ mare). O inovatie adusa de phprel este incarcarea dinamica, tehnologie descrisa in sectiunea 4. Securitate si performanta, subsectiunea 4.2.2. Incarcarea dinamica a modulelor.

Exista doua variante de phprel, cu functionalitati similare, una folosita pentru aplicatii de tip backend, denumita "back" si alta folosita pentru aplicatii de tip frontend, site-uri web, denumita "front". Diferenta intre cele doua consta in accesul la aplicatie (pentru varianta "back", restrictionat prin user si parola, pentru varianta "front" liber), url rewrite (functional doar in varianta "front") si functii avansate de securitate pe baza grupurilor de utilizatori (disponibile doar in varianta "back"). Toate celelalte module sunt disponibile in ambele variante.

3.1.1. Modulele phprel

Modulele care alcatuiesc nucleul (core) phprel indeplinesc urmatoarele functii:

- motorul de template-uri – realizeaza incarcarea continutului html si inlocuirea efectiva a tag-urilor speciale phprel
- manager-ul de query-uri – intermediaza query-urile intre aplicatie si bazele de date pentru a completa anumite detalii ale query-ului si pentru a oferi feedback-ul necesar altor module phprel (de exemplu cache-ului) si dumneavoastra (debugging)
- biblioteca centrala – cuprinde functiile care realizeaza operatiile de baza necesare altor module si asigura anumite procese izolate, care nu se incadreaza in nici un alt modul
- manager-ul setarilor – reglementeaza parametrii de functionare a celorlalte module
- accesul la baza de date – construieste si autocompleteaza query-uri, faciliteaza selectarea rapida a datelor si intermediaza comunicarea intre celelalte module si tabelele mysql.
- constructorul de formulare – construieste tag-uri html de formulare complet functionale, pe baza specificatiilor exterioare, venite din partea dumneavoastra sau de la phprel
- constructorul de liste – construieste liste in format html complet functionale, pe baza specificatiilor exterioare, venite din partea dumneavoastra sau de la phprel
- constructorul de cereri asincrone (A.J.A.X.) - construieste cod javascript functional de comunicare asincrona cu serverul pe baza specificatiilor dumneavoastra sau a nucleului phprel, raspunde cererilor asincrone pe baza handler-elor, si pune la dispozitie operatii fundamentale legate de A.J.A.X. si handler-e celorlalte module
- cosul de cumparaturi – se ocupa de functionarea sesiunii de cumparaturi si de preluarea comenzii, pe baza indicatiilor venite in principal de la nucleul phprel, dar si din partea dumneavoastra
- manager-ul de sesiune – se ocupa de securizarea sesiunii curente si de stocarea informatiilor de la o sesiune la alta si de pe un calculator pe altul (sesiuni persistente)
- manager-ul securitatii linkurilor – valideaza url-urile cu risc ridicat ca fiind autentice si ataseaza automat linkurilor aplicatiei un identificator pentru validare
- url rewrite – interpreteaza url-ul potrivit specificatiilor, alege pagina destinatie si construieste parametrii GET
- multilanguage – traduce template-urile pe baza datelor introduse in tabelul de traduceri, seteaza si schimba limba curenta, pune la dispozitie functii de traducere din php, descopera etichete netraduse si adauga noi intrari in tabelul de traduceri, sterge traducerile invechite
- cache – retine continutul unei anumite pagini pentru afisare rapida si economisirea resurselor la accesele ulterioare. Foloseste tehnologii inovative de adaptare, cacheSentinels si caching partial al elementelor de pagina, monitorizeaza sursele aplicatiei si baza de date pentru a determina eficient momentul regenerarii paginii.
- manager-ul securitatii informatiilor – o extensie a securitatii standard, modifica transparent paginile si query-urile pentru a proteja datele la care un grup de utilizatori nu are acces si pentru a restrictiona accesul la resurse.
- assistant – se foloseste de indicatiile date de dumneavoastra si de propriile predictii, deductii si presupuneri rezultate in urma analizei surselor aplicatiei pentru a prelua din sarcinile programatorului, realizand anumite operatii automat, facilitand realizarea rapida a altor operatii si completand anumite secvente de cod ale programatorului. Assistant-ul este cel care ofera ajutorul concret, fiind un modul special care imbina si foloseste dupa caz toate celelalte module.
- extensia functiilor de baza – pune la dispozitia dumneavoastra cateva functii folositoare

Toate aceste module, cu trei exceptii (motorul de template-uri, biblioteca centrala si manager-ul setarilor) pot fi activate sau dezactivate in functie de caz si de constrangerile legate de performanta sau consum de resurse. In mod obisnuit, nucleul phprel va optimiza automat folosirea modulelor pentru a va oferi maximum de functionalitate cu minimum de resurse consumate. Standard, phprel este configurat sa foloseasca toate modulele cu exceptia celui de multilanguage si a celui de securitate extinsa (in cazul unei aplicatii cu mai multe limbi sau care necesita un inalt grad de securitate bazat pe grupuri de utilizatori, va trebui sa activati modulul respectiv prin realizarea unei setari o singura data la inceputul proiectului). In plus, modulul de url rewrite nu este disponibil decat in varianta "front", iar cel de securitate extinsa doar in varianta "back".

3.1.2. Pagini – componenta de prezentare (html) si de calcul (php)

Conceptul central intr-o aplicatie construita folosind phprel este acela de "pagina", corespunzator unei pagini web accesate din cadrul aplicatiei. O pagina poate avea multiple continuturi, in functie de exemplu de parametrii get, post sau session, dar va pastra acelasi design. De exemplu, pentru afisarea a treizeci de produse din baza de date, se va construi o singura pagina "produs", care va afisa unul din acele produse, in functie de un parametru get. Nu fiecare varianta, produsa pentru fiecare produs, este considerata "pagina" ci tiparul in sine de afisare a unui produs este o "pagina". De fapt, conceptul de "pagina" reprezinta ceea ce dumneavoastra in mod intuitiv intelegeti prin "pagina web" la un nivel de programare, nu al utilizatorului final.

In phprel, o pagina are doua componente: o componenta html de prezentare care reprezinta pagina din punct de vedere visual, al design-ului si o componenta php care efectueaza toate manipularile de date, calculele si procesele (colectiv "calculul") necesare pentru concretizarea design-ului html intr-o forma finala afisabila unui utilizator. Un exemplu ar putea fi o lista de persoane cu date de contact: componenta de calcul ar selecta persoanele din baza de date, in timp ce componenta de prezentare ar contine tabelul html in care vor fi afisate persoanele. Procesul propriu-zis de completare a tabelului va fi realizat de asemenea de componenta de calcul (dar in phprel aceasta "legatura" este realizata in aproape toate cazurile automat).

Unei pagini ii corespund asadar doua fisiere: un php si un html. Acestea vor avea denumirea paginii pe care o reprezinta si vor fi create respectiv in directoarele web/pages/ si web/templates/. Html-ul atasat paginii, reprezentand componenta de prezentare, se considera "template"-ul paginii. O data cele doua fisiere create, pagina poate fi accesata adaugand la url-ul aplicatiei "?page=(denumire pagina)".

Denumirea nu va contine extensia ".php" sau ".html" si poate contine un subdirector (de exemplu: "?page=magazin/produse", unde magazin este un subdirector creat de dumneavoastra atat in web/pages/ cat si in web/templates/ si care contine produse.php respectiv produse.html - dar aceasta practica este categoric descurajata, in mod normal neexistand o justificare pertinenta pentru a folosi subdirectoare) dar nu poate, din motive de securitate, sa contina referinte la directoare "parent" (de forma "../"). Nucleul phprel va include in mod automat php-ul si html-ul cerut prin parametrul GET "page".

Important: in sursele aplicatiei dumneavoastra, pentru a afla sau folosi pagina curenta, folositi variabila globala $page si nu $_GET['page'], aceasta din urma reprezentand doar "pagina ceruta" si nu "pagina afisata". Motive pentru care pagina afisata poate sa difere de pagina ceruta sunt: pagina ceruta nu exista si modulul de url rewrite este activat ceea ce va determina ca pagina afisata sa fie prima pagina ("home") sau accesul pe pagina ceruta este restrictionat.

In ceea ce priveste structurarea paginilor in categorii, subcategorii, este recomandat sa folositi denumiri de pagina de tip "(categorie).(denumire)" sau "(categorie).(subcategorie).(denumire)", de exemplu "magazin.produse", folosirea directoarelor nefiind necesara.

Important: se recomanda ca denumirile sa fie formate din cuvinte separate prin punct, mai degraba decat underscore.

Evident, orice afisare se va face prin intermediul componentei de prezentare, afisarile "parazite" efectuate direct din php (de exemplu, prin "echo 'parasite output';") fiind suprimate in afara modului de dezvoltare (consultati sectiunea curenta, subsectiunea 3.1.5. Modul de dezvoltare). Aceasta masura este luata pentru a stabili o structura flexibila dar clara, structura care se demonstreaza a fi de un real folos in proiectele mari sau proiectele la care lucreaza mai multe persoane.

In plus, design-ul unei pagini merge pe modelul header – content – footer, html-ul corespunzator unei pagini fiind incarcat in "content" si "impachetat" in header si footer. Aceste doua elemente de layout sunt prezente de asemenea in web/templates/ si vor fi modificate dupa caz. Cele trei elemente (header, content, footer) sunt elementele principale ale paginii si sunt incarcate automat de nucleul phprel.

Important: exista web/pages/header.php si web/pages/footer.php corespunzatoare elementelor "header" si "footer" similar elementului "content" caruia ii corespund si un php si un html. Toate trei elementele sunt incarcate in "scope"-ul global al executiei php, in timp ce elementele secundare care impart suplimentar pagina vor fi incarcate intr-un scope local. Avand in vedere ca Assistant-ul realizeaza orice inlocuire automata de tag-uri doar din scope-ul global, acest lucru este important pentru functionalitatea paginii. De exemplu, o variabila php definita in header.php va putea fi inclocuita si in header, content, footer si in elemente secundare create de dumneavoastra. La fel este cazul variabilelor definite in content sau footer, dar o variabila definita in php-ul atasat unui element secundar se va comporta diferit: in cazul in care dumneavoastra permiteti Assistant-ului functia de "auto-globalizare" (activata standard), o variabila definita intr-un php al unui element secundar va putea fi inlocuita automat in html-ul atasat acelui element (si numai respectivul html). In cazul in care o variabila trebuie inlocuita in alte elemente sau functia de "auto-globalizare" nu este activa, va trebui sa specificati variabila ca globala. Pentru mai multe detalii despre Assistant si "auto-globalizare", consultati sectiunea curenta, subsectiunea 3.3. Assistant.

Componentele html ale header-ului, content-ului si footer-ului sunt incluse de radacina html a site-ului, web/templates/index.html in care se pot face de asemenea modificari de "layout" (aplicatiile simple nu vor necesita, in general, schimbarea index-ului). Index-ul are ca php atasat insasi "index.php" din radacina aplicatiei, care este responsabil pentru includerea intregului nucleu phprel si care centralizeaza toate paginile aplicatiei.

3.1.3. Motor de template-uri

Stim deja ca orice pagina sau element de pagina are doua componente: un php si un html. Html-urile, sau componenta de prezentare, sunt incarcate, manipulate si afisate folosind un motor de template-uri eficient, parte a nucleului phprel. Modulul incarca template-uri html, si executa modificari asupra lor in doua moduri:

- prin inlocuirea de tag-uri simple (corespunzatoare unei variabile, sau tag-uri speciale phprel precum cele de cos de cumparaturi – pentru detalii consultati sectiunea curenta, subsectiunea 3.8. Cos de cumparaturi si preluarea comenzilor)

- prin inlocuirea de bucle (corespunzatoare unor array-uri de tip rand => array asociativ).

In plus, modulul insereaza continutul unui alt element in elementul curent daca intalneste un tag de incluziune.

Astfel, cele trei tipuri de tag-uri permise de motor-ul de template sunt:

- <&variabila> - denumit in continuare "tag de variabila globala"
- <&loop> ... <&cheie1> ... <&cheie2> ... <cheie3> ... <&endloop> - denumit "bucla"
- <&include "(element)" /> - denumit "tag de incluziune"

Toate tag-urile phprel se incadreaza in una din aceste trei categorii. Folosind Assistant-ul, unele tag-uri derivate din primele doua pot capata anumite intelesuri. De exemplu, Assistant-ul se va descurca sa inlocuiasca variabile componente ale unui array, de exemplu <&variabila['cheie1'][0]['cheie2']>. Tag-ul, din punct de vedere structural, se afla in prima categorie. In plus, se pot inlocui tag-uri reprezentand functii: <&str_replace('phprel', 'phprel – web 3.0', $a)> care pot contine variabile php sau chiar variabile de template: <&str_replace('phprel', 'phprel- web 3.0', '<&a>')>. Pentru mai multe detalii, consultati sectiunea curenta, subsectiunea 3.3. Assistant.

Obiectul care contine template-urile curente este variabila globala $parser. In mod obisnuit, operatiile de inlocuire si/ sau citire de continut vor fi realizate automat de Assistant sau alte module ale nucleului phprel, dar variabila se poate folosi si manual prin metodele urmatoare:

- $parser->retrievefile('(element)') – va intoarce continutul elementului specificat, asa cum este el in momentul apelului
- $parser->loadfile('(element)', '(fisier.html)') – va incarca continutul fisierului specificat (care in mod obligatoriu contine si extensia, de regula ".html") in elementul specificat. Un tag de incluziune <&include "(element)" /> va include continutul elementului asa cum se regaseste el in $parser si nu in web/templates/, deci oricarui tag de incluziune ii corespunde un apel la $parser->loadfile (realizat in mod automat de Assistant).
- $parser->parsevar('(element)', '(variabila)'[, (valoare)]) – va inlocui variabila speficata (mai exact tag-ul <&(variabila)>) cu valoarea data. In lipsa unei valori, variabila va fi cautata in scope-ul global.
- $parser->parseloop('(element)', '(bucla)'[, (array)]) – va inlocui bucla specificata (mai exact fragmentul html continut intre <&loop "(bucla)"> si <&endloop>) cu randurile array-ului dat. In cazul in care nu se specifica un array, va fi cautat in scope-ul global un array cu numele buclei.
- Pentru rapiditate, se poate folosi functia parsevar('(variabila)', (valoare)[, '(element)']) care va inlocui in elementul curent (de exemplu, daca functia este apelata in "header.php", va inlocui in "header") variabila specificata cu valoarea respectiva. In cazul in care este necesara inlocuirea intr-un alt element, acesta se poate specifica totusi ca un al treilea parametru.
- Similar, se poate utiliza functia parseloop('(bucla)', (array)[, '(element)']).

In practica insa, probabil nu veti folosi aceste functii niciodata, iar cea mai probabila situatie in care veti folosi manual obiectul $parser va fi la construirea unui handler de cereri asincrone javascript + xml (A.J.A.X.) care trebuie sa intoarca un element de pagina (folosind $parser->retrievefile). Toate incluziunile, inlocuirile de variabile si de bucle vor fi realizate automat de Assistant pentru dumneavoastra pe baza unei corelatii de denumire sau pe baza unei denumiri speciale si/ sau a unei corelatii logice. De exemplu, pentru a inlocui variabila "<&var>" din header.html, este suficient sa definiti in header.php o variabila "$var = 'valoare';". Pentru mai multe detalii, consultati sectiunea curenta, subsectiunea 3.3. Assistant.

O variabila de forma <&(denumire)> poate contine in denumire orice caracter mai putin semnul ">" (inlocuit, daca este cazul, prin [lt] sau [ltoe] pentru "<=") si semnul ";" (inlocuit, daca este cazul, prin [sc]). [lt] abreviaza sintagma "less than", [ltoe] "less than or equal" iar [sc] "semi-colon". Astfel, variabile speciale interpretate de Assistant precum tag-urile conditionale se pot construi pornind de la acest tipar:

- <&if{(conditie)}> ... <&endif> sau
- <&if{(conditie)}> ... <&else> ... <&endif>

Conditia nu va putea contine asadar caracterele ">" sau ";", in cazul in care ele sunt necesare trebuind sa fie inlocuite cu [lt], [ltoe], [gt].

O bucla se defineste in modul urmator: <&loop "(denumire)"> ... <&cheie1> ... <&cheie2> ... <&endloop>. Denumirea nu poate contine caracterul apostrof. Este permisa imbricarea tag-urilor de variabila globala in denumirea unui loop: <&loop "subcategorii_<&id>">. Tag-urile interioare ale buclei trebuie sa fie chei ale fiecarui rand din array-ul care se inlocuieste. Un exemplu ar putea fi un array de forma: array (0 => array ('cheie1' => 'valoare1', 'cheie2' => 'valoare2',), 1 => array ('cheie1' => 'valoare1', 'cheie2' => 'valoare2',). Fiecare rand va fi inlocuit in continutul buclei si adaugat la rezultatul final. Cu toate ca o anumita cheie poate lipsi din randurile ulterioare, primul rand trebuie sa contina toate cheile folosite in loop. Buclele pot fi imbricate pe oricate nivele: <&loop "(bucla1)"> ... <&loop "(bucla2)"> ... <&endloop> ... <&endloop>.

O variabila inlocuita poate fi atat o referinta la o cheie simpla, "<&(denumire)>", cat si o referinta la o cheie a unui (sub)array al randului, de exemplu "<&(denumire)[(cheie1)]['(cheie2)']>".

Important: buclele contin, pe langa cheile array-ului dat pentru inlocuire, cateva chei speciale adaugate de motorul de template-uri phprel:

- row – inlocuit cu numarul randului curent (de la zero la N-1).
- row+1 – inlocuit cu numarul randului curent (de la 1 la N). "row+1" este o denumire conventionala, nu exista "row+2", "row+3" sau alta operatie matematica.
- odd – inlocuit cu 1 pentru randurile impare (primul rand in acest caz se considera ca are indice 1 si este deci impar) si cu 0 pentru cele pare
- (denumire loop).row
- (denumire loop).row+1
- (denumire loop).odd
- Acestea pot fi folosite in tag-uri conditionale sau afisari ale numarului de ordine. Evident, fiind chei inlocuite ale unei bucle, vor aparea in html sub forma unor tag-uri, de exemplu:

  <&row+1>

  sau

  <&produse.row+1> (unde "produse" este numele buclei)
- Elementele speciale pot fi prefixate de numele buclei in cazul imbricarii mai multor bucle.

In plus, un array fara elemente - array() - va determina inlocuirea buclei cu "nici un rand" astfel incat tot ceea ce este cuprins intre <&loop> si <&endloop> va fi exclus din template.

Un tag de incluziune se scrie in modul urmator: <&include "(element)" /> sau <&include "(element)">. Denumirea elementului nu va contine o extensie (se subintelege ".html") si nu va contine caracterul apostrof.

Important: Un tag de variabila si un tag de incluziune poate sa apara de oricate ori in template, in timp ce un tag de bucla poate apare o singura data (trebuie sa fie unic) sau cel putin sa fie unic pe fiecare element de pagina.

3.1.4. Structura de directoare si smartLocate

In phprel, toate fisierele care determina continutul aplicatiei se afla in subdirectorul web/. Acest lucru faciliteaza sincronizarea intre un server de dezvoltare si cel "live", precum si dezvoltarea in general prin accesul rapid la fisierul cautat (intr-o structura de directoare aglomerata, care contine atat directoare de continut cat si directoare ale framework-ului si/ sau ale altor biblioteci necesare, gasirea unui fisier poate deveni complicata).

Directorul web/ contine standard urmatoarele subdirectoare si nu este recomandata modificarea lor sau adaugarea de noi directoare:

- css/ - contine fisierele ".css" ale aplicatiei
- handlers/ - contine php-urile atasate handler-elor de cereri asincrone javascript + xml (A.J.A.X.)
- images/ - contine imaginile folosite in aplicatiei
- otherincs/ - poate contine orice alte php-uri create de dumneavoastra sau din surse "third party" care trebuie incluse si folosite in anumite situatii.
- pages/ - contine php-urile atasate fiecarei pagini si fiecarui element de pagina
- rules/ - contine regulile mysql specifice fiecarei pagini, impreuna cu functiile specifice de i/o, metoda de procesare a formularelor pe pagina si descrierile de tabele (in subdirectorul tables/). Pentru mai multe detalii, consultati sectiunea curenta, subsectiunea 3.2. Accesul la baza de date.
- settings/ - contine setari referitoare la url rewrite, formulare, liste, template-uri si constate folosite in proiect
- templates/ - contine template-urile (".html") atasate fiecarei pagini si fiecarui element de pagina, precum si template-urile de liste si cateva template-uri speciale

In plus, directorul radacina al aplicatiei dumneavoastra va contine urmatoarele subdirectoare:

- cfgs/ - grupeaza toate configurarile posibile pentru phprel si conexiunea la baza de date
- uploads/ - este de preferat sa contina toate upload-urile efectuate de aplicatie, eventual structurate dupa caz in subdirectoare
- webdevel/ - contine o structura similara directorului web/ si este folosit in modul de dezvoltare. Pentru detalii, consultati sectiunea curenta, subsectiunea 3.1.5. Modul de dezvoltare.
- In cazul in care nucleul phprel si directorul de "include" (continand aplicatiile "third-party" si cele javascript incluse in phprel) nu au fost "legate" de o alta locatie pe disc, directorul radacina va contine si subdirectoarele core/ (nucleul phprel) si include/

Directorul radacina contine trei fisiere:

- .htaccess
- index.php – care centralizeaza executia aplicatiei si lanseaza nucleul phprel
- functions.front.php/ functions.back.php (in functie de varianta aleasa) – un fisier obligatoriu initial fara continut, in care dumneavoastra veti scrie functii general valabile pentru aplicatie si care vor fi incluse automat de nucleul phprel

In phprel, chiar si cel mai mic detaliu care ar putea ingreuna dezvoltarea web a fost inlaturat. Beneficiati de tehnologia smartLocate pentru ca centralizarea subdirectoarelor de continut in directorul web sa nu va afecteze deloc: in sursele dumneavoastra, nu va trebui sa scrieti (aproape niciodata) "web/". In mod automat, phprel va indentifica situatiile in care unul din directoarele specificate trebuie prefixat de "web/". De exemplu, intr-un template al unei pagini este suficient sa scrieti:

```
<img alt="phprel" src="images/phprel.jpg" />
```

si phprel va inlocui in sursa finala valoarea atributului src cu "web/images/phprel.jpg". La fel si pentru css-uri, si chiar directorul "include". Pentru includerea manuala a template-urilor folosind obiectul global $parser (consultati sectiunea curenta, subsectiunea 3.1.3. Motor de template-uri), nu va trebui sa precizati decat denumirea fisierului html, fara directorul "web/templates/".

In plus, smartLocate permite existenta a doua versiuni paralele a aceluiasi fisier – una "live", una "in curs de dezvoltare" – in directoarele web/ respectiv webdevel/ si va alege automat versiunea corespunzatoare situatiei in care va aflati. Pentru mai multe detalii, consultati sectiunea curenta, subsectiunea 3.1.5. Modul de dezvoltare. Pentru ca smartLocate sa functioneze in totalitate si pentru a folosi optiunea de caching inclusa in nucleul phprel, orice "include" suplimentar din php, neefectuat automat, va trebui facut folosind functia locate($numefisier), de asemenea fara a preciza directorul "web/". De exemplu:

```
include(locate('otherincs/somefile.php'));
```

sau

```
require_once(locate('pages/somepage.php'));
```

SmartLocate face de asemenea posibila includerea nucleului si a aplicatiilor third-party dintr-un singur loc centralizat, pentru toate aplicatiile de pe serverul respectiv. Acest lucru poate fi util in cazul serverului dumneavoastra de dezvoltare, dar si in cazul unui site web care contine atat o sectiune frontend cat si o sectiune backend: directoarele core/ si include/ vor putea sta doar in radacina frontend-ului, iar backend-ul le va include de acolo. La fel si pentru directorul "uploads/", care in cazul variantei backend poate fi legat de directorul "uploads/" al frontend-ului.

Configurarea functiei smartLocate se realizeaza din cfgs/smartlocate.inc.php folosind parametrii:

- $Tcfg_distributionDirectory – de obicei, singura setare care trebuie modificata, indicand calea spre directorul "distributiei" phprel care cuprinde directoarele "core/" si "include/"
- $Tcfg_smartincludedir – array care specifica pentru fiecare director (dat prin cheie), directorul corespondent real. De obicei, nu trebuie modificat sau adaugat vreun element la acest array.
- $Tcfg_smartincludedevel – array care specifica pentru fiecare director (dat prin cheie), directorul corespondent in situatia in care aplicatia este incarcata in modul de dezvoltare. De obicei, nu trebuie modificat.
- $Tcfg_smartincludefile – array care specifica pentru un fisier (dat prin cheie) fisierul corespondent real. Poate fi adaugat in fisierul de configurare in cazul in care anumite fisiere au un alt corespondent pe disc.

In cazuri exceptionale, folosind aceste configurari smartLocate poate fi folosit pentru a indica o locatie cu totul noua pentru directoarele aplicatiei, de exemplu pentru a comuta intre o versiune mai noua si o versiune mai veche a aceluiasi site.

3.1.5. Modul de dezvoltare

In phprel, feedback-ul necesar corectarii erorilor si imbunatatirii paginii este la indemana dumneavoastra in orice moment.

(...)

Modul de dezvoltare se activeaza in versiunea "front" adaugand "devel/" la url-ul curent, dupa url-ul radacina si inainte de parametrii optional adaugati. De exemplu, pentru url-ul

"http://www.domain.ro/shop/someproduct" url-ul in modul de dezvoltare va fi: "http://www.domain.ro/devel/shop/someproduct". In versiunea "back", adaugati un parametru GET denumit "devel" cu valoarea "1". De exemplu, pentru "http://www.domain.ro/admin/index.php?page=main" url-ul va deveni "http://www.domain.ro/admin/index.php?page=main&devel=1". Nucleul phprel va modifica automat linkurile din pagina pentru a pastra modul de dezvoltare pe parcursul navigarii.

In plus, atunci cand modul de dezvoltare este activat, se incarca in locul fisierului standard de configurare cfgs/config.inc.php, fisierul de configurare cfgs/advanced/devel.config.inc.php care poate fi eventual modificat pentru a folosi o alta baza de date. Nu in ultimul rand, se pot introduce modificari in sursele site-ului fara a afecta modul "live": daca un fisier care contribuie la continutul paginii curente are corespondent in directorul "webdevel/", atunci in modul de dezvoltare acel fisier va fi incarcat in locul celui standard din directorul "web/". Utilizatorii obisnuiti ai site-ului vor continua sa navigheze pe versiunea initiala de site, in timp ce dumneavoastra sunteti liber sa efectuati noile modificari chiar pe serverul "live" si testa in fiecare moment functionarea lor. Aceasta functie poate fi utila si pentru a prezenta clientilor doua versiuni paralele ale aceleiasi pagini. De exemplu, daca se incarca pagina "produse", in mod evident vor exista fisierele "web/pages/produse.php" si "web/templates/produse.html". La crearea fisierului "webdevel/templates/produse.html" si activarea modului de dezvoltare, fisierul php inclus va ramane in continuare "web/pages/produse.php" in timp ce template-ul inclus va fi "webdevel/templates/produse.html". Aceasta schimbare este valabila chiar si pentru fisiere ".css" (din css/) sau imagini (din images/).

3.1.6. Configurari de baza

Nucleul phprel a fost construit pentru flexibilitate. Fiecare proces, functie sau operatie indeplinita automat sau asistat de phprel poate fi intrerupta, modificata sau controlata. In plus, pentru maximum de adaptare si pentru a va oferi un framework modelat in mod unic dupa aplicatia pe care o construiti, aveti la dispozitie o serie de configurari de la elementare pana la avansate.

Vom incepe cu setarile de baza, in primul rand caracteristice oricarei aplicatii web, apoi cele necesare pentru a construi un proiect cu phprel. La inceperea unui nou proiect, se va configura mai intai conexiunea la baza de date:

- in cfgs/config.inc.php, se completeaza variabilele:
  - $Tcfg_dbhost – adresa serverului mysql, de regula 'localhost'
  - $Tcfg_dbuser – utilizatorul mysql
  - $Tcfg_dbpass – parola mysql
  - $Tcfg_db – denumirea bazei de date

Orice aplicatie web are o adresa URL radacina, care se poate configura de asemenea in cfgs/config.inc.php, dar in mod normal este determinata automat. Pentru a completa manual adresa, modificati valoarea variabilei $websiteURL.

In plus, $Tcfg_DEBUG stabileste daca mesajele de debugging sunt afisate sau nu. Pentru un server "live", aceasta setare trebuie trecuta pe 0, chiar si in configurarea modului de dezvoltare.

O data conexiunea cu baza de date stabilita, trebuie setata calea catre directorul "distributiei" phprel, setare care se face in fisierul de configurare a functiei smartLocate:

- in cfgs/smartlocate.inc.php, se completeaza:

- $Tcfg_distributionDirectory - in cazul in care ati copiat directoarele core/ si include/ in directorul radacina al aplicatiei, modificati $Tcfg_distributionDirectory pentru a indica acest lucru, atribuindu-i valoarea vida ". Daca directorul se afla intr-o alta locatie (in cazul serverelor de dezvoltare, de exemplu), indicati calea completa catre director prin valoarea respectivei variabile. Valoarea trebuie sa fie urmata de caracterul "/".

Aplicatia se poate deja incarca in browser. Se recomanda incarcarea interfetei Web Assistant pentru crearea tabelelor interne phprel si stabilirea datelor de acces la interfata. Aceasta etapa se poate realiza intuitiv iar Web Assistant va pune la dispozitie toate informatiile necesare.

Nu in ultimul rand, se recomanda setarea unor variabile specifice proiectului:

- in web/settings/constants.inc.php, se modifica variabilele:
  - $Tcfg_projectname – denumirea site-ului sau a aplicatiei, folosita atat pentru afisari cat si pentru securitatea sesiunii si a altor informatii.
  - $Tcfg_from_mail – adresa de email de pe care se vor trimite mail-uri atunci cand folositi functia de trimitere mail-uri a nucleului phprel
  - $Tcfg_from_name – numele cu care se semneaza mail-urile trimise folosind functia inclusa in nucleul phprel

Atunci cand adaugati propriile dumneavoastra constante, se recomanda sa le adaugati in web/settings/constants.inc.php.

In plus, se pot edita anumite setari implicite pentru a corespunde proiectului dumneavoastra. In mod normal, aplicatia functioneaza in parametrii fara modificari efectuate asupra acestor setari:

- in cfgs/advanced/core.settings.inc.php exista o serie de setari legate comportamente standard ale nucleului phprel. Unele setari vor fi tratate intr-o sectiune ulterioara de configurari avansate. Momentan, daca este cazul, se pot modifica urmatoarele variabile:
  - $Tcfg_backUsersTable – denumirea tabelului folosit pentru logarea utilizatorilor sectiunii "backend" a aplicatiei. Modificarea acestei denumiri va trebui efectuata impreuna cu modificarea denumirii descrierii respectivului tabel, web/rules/tables/io.back.users.php.
  - In cazul unei aplicatii in limba engleza sau intr-o alta limba, traduceti mesajele standard de validare automata a formularelor, folosind:
    - $Tfrm_Vloading – mesajul de asteptare la validarea unui formular
    - $Tfrm_Vempty – mesajul de eroare pentru un camp care trebuie obligatoriu completat. Se poate folosi simbolul special "{label}" – inlocuit automat cu "label"-ul/ numele real/ denumirea campului.
    - $Tfrm_Vother – mesajul de eroare pentru un camp care trebuie sa aiba obligatoriu un anumit format. Se pot folosi simbolurile speciale "{label}" si "{type}" – tipul obligatoriu al campului, asa cum este el specificat in web/settings/form.settings.inc.php.
    - $Tfrm_Vasync – mesajul de eroare in cazul campurilor validate printr-o cerere asincrona javascript + xml (A.J.A.X.). Se poate folosi simbolul special "{message}" – inlocuit cu mesajul propriu-zis intors de cerere.
    - $Tfrm_Verrors – "cadrul" in care sunt afisate erorile. Se va folosi simbolul "{errors}" – inlocuit cu mesajele de eroare rezultate in urma validarii, la care se poate adauga orice alta informatie/ secventa html.

    - $Tfrm_Vdiv – in cazul in care nu veti specifica dumneavoastra un alt div de eroare, phprel va adauga automat div-ul de eroare asa cum este el specificat prin aceasta variabila. Id-ul propriu-zis al div-ului va fi inlocuit de phprel cu un id real si nu trebuie modificat.
    - $Tfrm_form_post_async_ok – mesajul afisat dupa trimiterea datelor unui formular printr-o cerere asincrona javascript + xml (A.J.A.X.)

  Pentru mai multe detalii, consultati sectiunea curenta, subsectiunea 3.4. Formulare.
- $Tcfg_translationsTable – in cazul in care construiti o aplicatie in mai multe limbi, modulul multilanguage oferit de phprel va folosi un tabel intern cu denumirea specificata prin aceasta variabila. In mod obisnuit, nu este necesar sa schimbati valoarea implicita.
- $Tcfg_defaultLanguage – limba principala in cazul unui proiect multilanguage, in forma prescurtata la doua caractere.
- $Tcfg_allLanguages – toate limbile disponibile in aplicatie, in forma prescurtata la doua caractere, separate prin virgula.
- $Tshp_usersTable – denumirea tabelului folosit pentru logarea utilizatorilor din "frontend". Ca si in celelalte cazuri de denumiri de tabel, nu trebuie neaparat modificata valoarea variabilei. In cazul schimbarii tabelului, corelati schimbarea si cu denumirea descrierii web/rules/tables/io.front.users.php.
- $Tshp_productsTable – denumirea tabelului de produse atunci cand implementati un cos de cumparaturi pe site
- $Tshp_cartFields['price'] – denumirea campului de pret din tabelul de produse.
- $Tshp_cartTransportFields – lista de campuri separate prin virgula, care trebuie copiate din tabelul de produse in tabelul de sesiune de cumparaturi. Includeti aici toate campurile referitoare la produs relevante pentru comanda, care trebuie pastrate exact asa cum erau ele in momentul comenzi. Campuri cu aceleasi denumiri trebuie create in tabelul de sesiuni de cumparaturi.
- $Tshp_sessionTable – denumirea tabelului intern phprel care pastreaza sesiunile de cumparaturi.
- $Tshp_ordersTable – denumirea tabelului care inregistreaza comenzile. In cazul modificarii denumirii, modificati si denumirea descrierii corespunzatoare din web/rules/tables/.
- $Tshp_shipmentTable – denumirea tabelului de adrese de livrare, atasate comenzilor.
- $Tshp_autoClearCartAfterCheckout – daca este setat pe "true", cosul de cumparaturi va fi golit automat dupa procesarea comenzii.
- $Tshp_redirectAfterLogin – daca este setat pe "false", in urma procesului de login realizat automat de phprel, user-ul va fi redirectionat catre aceeasi pagina. In cazul precizarii unui URL, user-ul va fi redirectionat catre acel URL.
- $Tshp_redirectAfterLogout – similar variabilei precedente, dar referitoare la procesul de logout automat.
- $Tcfg_useSession_withSecure – setata pe true, variabila comunica modulului phprel responsabil pentru validarea linkurilor cu potential risc sa foloseasca in generarea codului de validare si id-ul de sesiune php pentru sporirea securitatii. In mod obisnuit, aceasta masura suplimentara nu este necesara.

- $Tcfg_keysTable – denumirea tabelului intern phprel responsabil pentru stocarea cheilor aleatoare folosite pentru a generarea codurile de validare a url-urilor cu potential risc.
- $Tcfg_secure – array asociativ care specifica (prin cheie) parametrii GET considerati cu risc ridicat si validati automat

Pentru mai multe detalii consultati sectiunea 4. Securitate si performanta, subsectiunea 4.1.1. Validarea url-urilor.

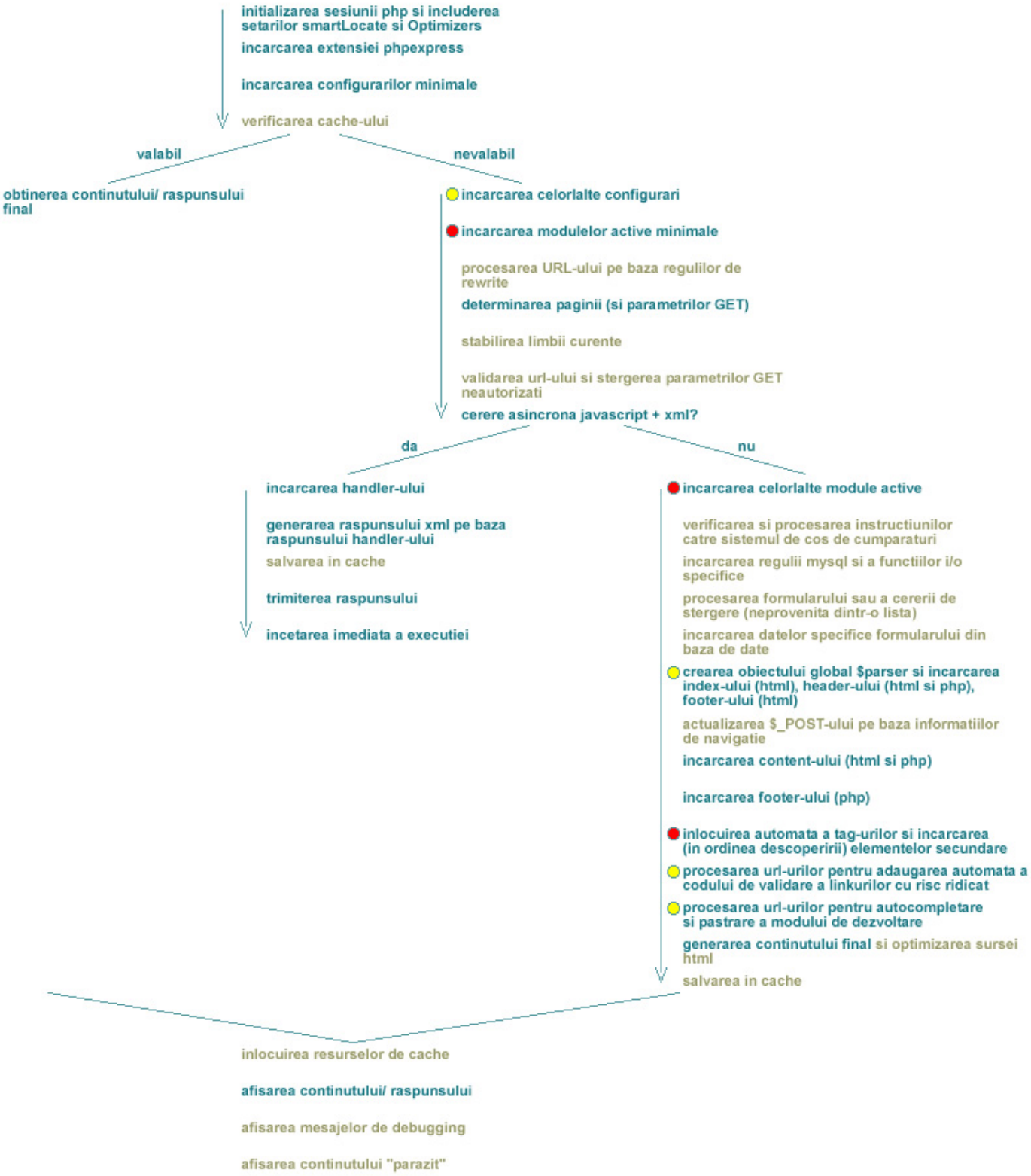
- $Tmem_table – denumirea tabelului intern phprel responsabil pentru stocarea datelor de la o sesiune la alta
- $Tcfg_accessTablesPrefix – prefixul denumirilor tabelelor interne phprel responsabile pentru gestionarea grupurilor de utilizatori si politicilor de acces.

Pentru detalii in legatura cu celelalte setari disponibile in cfgs/core.settings.inc.php precum si alte setari disponibile in phprel, consultati sectiunea 5. Elemente avansate, subsectiunea 5.6. Configurari avansate.

3.1.7. Schema de functionare simplificata

Pentru o viziune de ansamblu asupra succesiunii proceselor in nucleul phprel, consultati schema din imagine. Etapele indicate cu gri nu sunt intotdeauna parcurse, fiind omise atunci cand modulul care se ocupa de respectivul proces este inactiv (in cazul cache-ului, rewrite-ului, multiplelor limbi, validarea url-urilor, cosului de cumparaturi, regulilor mysql, optimizarea sursei html si debugging) sau contextul nu necesita realizarea respectivei etape (in cazul procesarii instructiunilor catre cosul de cumparaturi, incarcarea regulii mysql, procesarea si/ sau incarcarea formularului principal, stergerea din tabel, actualizarea $_POST-ului, inlocuirea resurselor de cache). Procesele marcate printr-un indicator galben sunt consumatoare medii de resurse in timp ce procesele marcate printr-un indicator rosu sunt consumatoare intensive de resurse.

Pentru detalii privind incarcarea dinamica a modulelor, consultati sectiunea 4. Securitate si performanta, subsectiunea 4.2.2. Incarcarea dinamica a modulelor. Procesele denumite in schema sunt explicate pe larg in sectiunea curenta, sectiunea 4. Securitate si performanta si, eventual, sectiunea 5. Elemente avansate.


initializarea sesiunii php si includerea setarilor smartLocate si Optimizers
incarcarea extensiei phpexpress
incarcarea configurarilor minimale
verificarea cache-ului
valabil
nevalabil
obtinerea continutului/ raspunsului final
incarcarea celorlalte configurari
incarcarea modulelor active minimale
procesarea URL-ului pe baza regulilor de rewrite
determinarea paginii (si parametrilor GET)
stabilirea limbii curente
validarea url-ului si stergerea parametrilor GET neautorizati
cerere asincrona javascript + xml?
da
nu
incarcarea handler-ului
generarea raspunsului xml pe baza raspunsului handler-ului
salvarea in cache
trimiterea raspunsului
incetarea imediata a executiei
incarcarea celorlalte module active
verificarea si procesarea instructiunilor catre sistemul de cos de cumparaturi
incarcarea regulii mysql si a functiilor i/o specifice
procesarea formularului sau a cererii de stergere (neprovenita dintr-o lista)
incarcarea datelor specifice formularului din baza de date
crearea obiectului global $parser si incarcarea index-ului (html), header-ului (html si php), footer-ului (html)
actualizarea $_POST-ului pe baza informatiilor de navigatie
incarcarea content-ului (html si php)
incarcarea footer-ului (php)
inlocuirea automata a tag-urilor si incarcarea (in ordinea descoperirii) elementelor secundare
procesarea url-urilor pentru adaugarea automata a codului de validare a linkurilor cu risc ridicat
procesarea url-urilor pentru autocompletare si pastrare a modului de dezvoltare
generarea continutului final si optimizarea sursei html
salvarea in cache
inlocuirea resurselor de cache
afisarea continutului/ raspunsului
afisarea mesajelor de debugging
afisarea continutului "parazit"

3.2. Accesul la baza de date

Cateva detalii privind selectarea, inserarea, modificarea sau stergerea datelor din tabelele mysql au fost explicate in sectiunea 2. Elemente de baza, subsectiunea 2.2. Accesul rapid la baza de date. Aceasta sectiune completeaza informatiile respective si detaliaza descrierile de tabele precum si un concept de baza in phprel, acela de "regula mysql" (rule).

3.2.1. Descrierea tabelelor

Asa cum am stabilit deja in sectiunea 1. phprel la prima vedere, subsectiunea 1.7. Descriere simpla de tabel, intre paginile unei aplicatii si tabelele mysql exista o stransa legatura. In plus, un tabel mysql isi pastreaza aproximativ aceeasi logica de functionare indiferent de pagina. Pentru ca phprel sa fie cat mai de ajutor in construirea paginilor, multe dintre presupunerile si asistenta acordata se vor baza pe legaturile intre pagini si tabele si este foarte important ca phprel sa "inteleaga" logica fiecarui tabel. In acest sens, fiecarui tabel i se creaza o "descriere" care explica dupa un anumit tipar logica tabelului si modul in care se relationeaza cu alte tabele si adauga particularitati ale tabelului care nu se incadreaza in respectivul tipar. Descrierile de tabel sunt fisiere php create in web/rules/tables/, avand denumiri de forma "io.(nume tabel in format phprel).php", unde (nume tabel in format phprel) este denumirea tabelului in care toate caracterele underscore "_" au fost inlocuite cu caractere punct ".". Pentru a crea o descriere de tabel se poate folosi interfata Web Assistant sau se poate duplica fisierul web/rules/tables/io.sampletable.php, caruia apoi i se schimba denumirea si continutul in functie de caz. Fisierul serveste drept punct de plecare care contine toate configurarile posibile, acestea fiind:

- $fields – campurile tabelului separate prin virgula. Acestea vor fi selectate in orice query construit automat sau autocompletat de phprel. Desi nu este recomandat, daca este necesar se pot adauga aici chiar si alte expresii mysql valide, nu doar campurile tabelului. De exemplu, pentru un tabel cu campurile id, nume, prenume, functie se poate defini fields ca:

  ```
  $fields = 'id, nume, prenume, functie';
  ```
  sau
  ```
  $fields = "id, nume, prenume, CONCAT(prenume, ' ', nume) AS fullname, functie,
  functie AS duplicat_functie";
  ```
- $defaultorder – campul sau campurile dupa care se ordoneaza automat atunci cand nu este specificata o ordonare intr-un query. De exemplu:

  ```
  $defaultorder = 'nume ASC, prenume ASC';
  ```
- $activeclause – folosita pentru a exclude din selectare anumite linii "inactive" sau "invizibile" sau "neconfirmate" sau "ascunse". Toate query-urile vor fi completate automat cu o conditie de forma "status>0" daca $activeclause = true sau, in cazul in care $activeclause specifica un anumit camp al tabelului, cu "(camp)>0". De exemplu,

  ```
  $activeclause = true;
  ```
  sau
  ```
  $activeclause = 'este_vizibil';
  ```
- $namefield – campul care are rol de "denumire" a liniei, sau expresia mysql cu rol de denumire a liniei. Nu poate contine identificatorul mysql "AS". De exemplu:

```
$namefield = 'titlu';
```
sau
```
$namefield = "CONCAT(prenume, ' ', nume)";
```

- $filterfield – un camp sau o serie de campuri pe care se aplica de obicei conditii. Va fi folosit in combinatie cu getData('(nume tabel).filter', (valori campuri)) pentru filtrare rapida. Pentru a specifica mai multe campuri $filterfield va fi un array simplu definit array('camp1', 'camp2', ...); iar valorile folosite in getData vor fi similar construite sub forma unui array('valoare1', 'valoare2', ...); De exemplu:
```
$filterfield = 'id_categorie';
```
sau
```
$filterfield = array('id_categorie', 'id_subcategorie');
```
- $relate – indica modul in care se relationeaza tabelul de alte tabele. Pentru phprel, un tabel se poate relationa de un alt tabel in trei moduri: parent – child, printr-o legatura optionala, ca nomenclator. Daca un tabel are un alt tabel relationat ca "parent", in mod obligatoriu va exista un camp in tabelul curent care face legatura intre cele doua tabele pe baza de chei primare. O linie din tabelul "parent" va putea corespunde mai multor linii din tabelul "child". Legatura va fi indicata si in sens invers, prin "child" (fara camp de legatura). O legatura optionala se comporta in principal ca o legatura unu-la-unu, tabelele situandu-se pe acelasi nivel, legatura putand sa existe sau nu, iar unul dintre tabele va trebui sa aiba un camp care stabileste legatura pe baza cheilor primare, in timp ce celalalt nu are camp. O legatura de tip nomenclator este intr-un singur sens de la un camp reprezentand o cheie la un tabel nomenclator din care se va substitui valoarea cu denumirea ei. Astfel:
  - parent – are ca relatie complementara "child", are camp, desemneaza o relatie de tip unu-la-n, unde n poate fi si 0. In cazul unei stergeri recursive, atunci cand o linie parent este stearsa vor fi sterse toate liniile child.
  - linked – are ca relatie complementara tot "linked", una dintre ele fiind cu camp, cealalta fara camp, desemneaza o relatie de tip unu-la-zero sau unu-la-unu, dar in cazuri speciale poate desemna si o relatie unu-la-n care nu se doreste a fi parent-child. In cazul unei stergeri recursive, atunci cand o linie linked este stearsa, pe linia aflata in legatura cu ea din tabelul complementar, campul de legatura se va seta la valoarea 0.
  - nomen – nu are relatie complementara, are camp, desemneaza o relatie de tip unu-la-unu in care exista mereu corespondent, corespondentul fiind un nomenclator prin care o valoare simbolica a unui camp ii corespunde o denumire concreta. In cazul unei stergeri recursive, nu se va intampla nimic pe o astfel de legatura.
  - child – complementul relatiei "parent", nu are camp.

  La selectarea recursiva a datelor, pentru campurile legate ca nomenclator valoarea se va substitui pur si simplu in camp, in timp ce pentru celelalte legaturi vor fi selectate liniile corespondente din tabelul complementar si adaugate la array-ul rezultat sub forma unei noi chei cu denumirea tabelului si valoare un array de randuri corespunzatoare liniilor selectate.
  Sunt doua forme de a preciza relationarea unui tabel cu tabelul curent:
  - $relate['(nume tabel relationat)'] =,relatie).by.(camp)' sau, in cazul relationarilor fara camp $relate['(nume tabel relationat)'] = 'as.(relatie)'. De exemplu:
```
$relate['categorii'] = 'as.parent.by.id_categorie';
$relate['produse'] = 'as.child';
```

    - $relate['(nume tabel relationat).as.(relatie).by.(camp)'] = true sau, in cazul relationarilor fara camp, $relate['(nume tabel relationat).as.(relatie)'] = true. De exemplu:
        ```
        $relate['categorii.as.parent.by.id_categorie'] = true;
        $relate['produse.as.child'] = true;
        ```
- $type – precizeaza tipul de camp html corespunzator unui camp din tabel. Folosit pentru a preciza tipul campului atunci cand phprel nu il intuieste corect si dumneavoastra nu doriti sa-l specificati in fiecare formular in parte. In plus, are o utilizare speciala pentru campurile specificate ca fiind checkbox, ajutand la "debifarea" (schimbarea in tabel dintr-o valoare pozitiva in 0) a campurilor de tip checkbox din formulare construite si procesate cu ajutorul phprel. Pentru o functionare corecta, campurile checkbox se vor preciza intotdeauna prin $type, chiar daca phprel intuieste corect tipul lor. De exemplu:
    ```
    $type['status'] = 'checkbox';
    ```
- $preventblankupdate – indica nucleului phprel sa nu schimbe niciodata valoarea unui camp in sirul vid. Acest lucru este in primul rand util in cazul campurilor de parola, in formulare de modificare a datelor de utilizator (de exemplu), atunci cand comportamentul standard al paginii in cazul in care parola nu a fost introdusa (utilizatorul voia doar sa schimbe alte date, nu si parola) trebuie sa fie pastrarea parolei asa cum este acum in tabel (si nu inlocuirea cu sirul vid). De exemplu:
    ```
    $preventblankupdate['parola'] = true;
    ```
    Important: phprel verifica daca valoarea trimisa este vida doar dupa ce a procesat campul printr-o eventual functie de control a datelor de intrare. Un camp initial vid a carui valoare este schimbata printr-o functie std i/o sau o functie i/o specifica paginii nu va fi considerat vid.
- $virtual – phprel va permite sa definiti tabele virtuale, pe care puteti lucra ca si cum ar fi tabele adevarate. Astfel, unui tabel din phprel poate sa-i corespunda un alt tabel mysql sau o parte de tabel (stabilita printr-o conditie). Un exemplu ar fi un tabel mysql "entitati" care contine atat categorii cat si subcategorii de produse, diferenta constand intr-o valoare nula sau nenula a unui camp "id_parent". Am putea crea doua tabele virtuale, categorii si subcategorii, legate de acelasi tabel "entitati" care sa se comporte ca si cand ar fi tabele separate de categorii si subcategorii. In acest caz, in web/rules/tables/io.categorii.php avem:
    ```
    $virtual['entitati'] = "id_parent='0'";
    ```
    Iar in web/rules/tables/io.subcategorii.php avem:
    ```
    $virtual['entitati'] = 'id_parent>0';
    ```
    Important: daca partea de tabel se defineste prin mai multe campuri, $virtual va permite doar conjunctia conditiilor simple (doar identificatorul "AND" folosit pentru a lega conditiile pe fiecare camp). Daca tabelul doar indica spre alt tabel mysql, fara nici o conditie, folositi "1" in loc de conditie.
- $primarykey – cheia primara a tabelului. In phprel, se recomanda ca fiecare tabel sa aiba o cheie primara denumita "id". In cazul in care tabelul nu se poate construi cu un camp "id" cheie primara, specificati prin $primarykey denumirea cheii primare:
    ```
    $primarykey = 'codprodus';
    ```
- $database – denumirea bazei de date caruia apartine tabelul. Baza de date principala (cea configurata in cfgs/config.inc.php) se specifica prin 'primary', orice alta baza de date indicata trebuie specificata in cfgs/advanced/more.dbs.inc.php. Pentru mai multe detalii, consultati sectiunea 5. Elemente avansate, subsectiunea 5.1. Suport pentru multiple baze de date.

- $secure - in cazul in care ati activat modulul avansat de securitate pe baza de grupuri de utilizatori, phprel restrictioneaza transparent in baza unor politici accesul utilizatorilor la liniile anumitor tabele. Astfel, daca un tabel este restrictionat folosind $secure = true, se poate stabili o politica de acces care sa limiteze un utilizator sa utilizeze doar liniile introduse de el, sau de grupul lui (sau la toate liniile). Nici o modificare sau efort din partea dumneavoastra nu va fi necesar, phprel va modifica activ toate query-urile efectuate pe tabelul respectiv, adaugand conditii de asa maniera incat rezultatele sa apartina doar respectivului utilizator sau grupului sau. In cazul in care un grup are drepturi depline asupra tabelului, query-urile nu vor fi modificate in nici un fel. Tabelele restrictionate trebuie sa contina, pe langa campurile proprii, urmatoarele campuri folosite de phprel pentru a asigura securitatea informatiilor din tabel:
  - ownerid – id-ul utilizatorului care detine linia respectiva
  - groupid – id-ul grupului principal caruia apartine utilizatorul respectiv
  - createdon – data si ora (in format unixtime) la care linia a fost creata
  - modifiedid – id-ul utilizatorului care a modificat ultima data linia
  - lastchange – data si ora (in format unixtime) la care linia a fost ultima oara modificata
- $actualownerfield – in cazul in care accesul la tabel este restrictionat (prin $secure = true), se poate specifica un camp care este adevaratul camp ce indica utilizatorul detinator al liniei (un camp al tabelului care ar trebui sa corespunda cu "ownerid"). Corespondenta se va face apoi automat, "ownerid" preluand valoarea campului $actualfield si doar utilizatorii cu drepturi depline asupra tabelului vor putea introduce linii "in numele alcuiva", atribuind o valoare diferita pentru campul specificat prin $actualownerfield. De exemplu:

  ```
  $actualfield = 'id_user';
  ```
- $override – in cazul in care accesul la tabel este restrictionat, utilizatorii vor avea dreptul la liniile stabilite prin politica de securitate plus liniile care respecta conditia data prin $override. Conditia poate fi orice expresie mysql si va reprezenta anumite linii sau categorie de linii din tabel care trebuie sa fie vizibile indiferent de cat de restrictiva este politica de securitate. De exemplu:

  ```
  $override = "ispublic='1'";
  ```
  sau
  ```
  $override = 'id IN (1,2,3)';
  ```
  Pentru mai multe detalii referitoare la restrictionarea accesului pe baza grupurilor de utilizatori, consultati sectiunea 5. Elemente avansate, subsectiunea 5.4. Grupuri de utilizatori.
- functii standard de control cu denumire variabila – folosite pentru a stabili comportamentul standard al anumitor campuri din tabelul respectiv. Pentru mai multe detalii, consultati sectiunea curenta, subsectiunea 3.2.3. Functii de control, comportament standard, comportamente diferite pe pagina.

De observat faptul ca nu este necesar sa prefixati campurile de denumirea tabelului in $defaultorder, acest lucru fiind realizat automat de phprel. La descrierea relationarilor folosind $relate, denumirile de tabele (referitoare la tabelele relationate) vor fi mereu in format phprel, avand toate underscore-urile inlocuite cu caractere punct ".". Se recomanda ca si denumirile paginilor (si implicit a php-urilor din web/pages/ si a html-urilor din web/templates/) sa fie compuse tot din cuvinte separate prin punct. Nucleul phprel va face mai usor legatura intre o pagina si un tabel daca folositi aceasta conventie de denumire pentru ambele entitati. Conventia in sine isi propune sa contureze tabelul ca o entitate sau subentitate caruia i se vor aplica anumite functii (consultati sectiunea curenta, subsectiunea 3.2.3. Functii

de scriere rapida a query-urilor si prelucrarea rezultatelor, functia phprel getData). De exemplu, tabelului "produse" i se va aplica o functie de cautare scriind "produse.search", si respectand conventia si denumind tabelele printr-o logica simpla de apartenenta, atunci cand vom cauta printre imaginile atasate unui produs, imaginile fiind de cele mai multe ori stocate intr-un tabel de forma "produse_imagini", vom putea scrie "produse.imagini.search". Se observa ca se creaza un anumit tipar de forma entitate, subentitate, actiune, dar acest lucru va fi posibil daca si tabelele mysql vor fi denumite dupa o anumita conventie. Un tabel avand o anumita denumire (X) va determina ca toate tabelele subordonate lui sau care aduc informatii aditionale liniilor lui sa fie denumite (X)_(nume secundar). Daca apoi acestui tabel ii corespunde un alt tabel cu informatii suplimentare, acela va fi denumit (X)_(nume secundar)_(nume secundar de nivel 2). Dumneavoastra veti putea denumi tabelele dupa orice conventie considerati potrivit, sau fara nici o conventie, dar retineti ca in phprel acolo unde folositi o denumire de tabel in cadrul unei descrieri de tabel, underscore-urile trebuie inlocuite cu puncte.

Denumirea de tabel in format phprel va putea fi folosita si in functiile sau tag-urile speciale phprel in care se cer denumiri de tabel, dar nu va putea fi folosita in parametrii de tip "where", "order", "group by", "having" ale query-urilor scrise direct sau prin intermediul unei functii sau tag special phprel.

Timpul castigat de dumneavoastra pe parcursul dezvoltarii aplicatiei ca urmare a descrierii cat mai complete a tabelelor mysql poate fi uneori incredibil, si constant veti castiga "mult" timp. Dar realizarea descrierilor poate da senzatia ca inceputul dezvoltarii este lent. De aceea, Web Assistant – interfata de management si dezvoltare inclusa in phprel – va ajuta in scrierea acestor descrieri, atat prin scrierea lor concreta in directorul web/rules/tables/ cat si prin intuirea valorilor pentru majoritatea variabilelor componente ale descrierilor. Incarcati intefata Web Assistant si urmati linkul "tabele (io)". Selectati tabelele dorite pentru a crea descrierile. Interfata va ajuta doar la crearea initiala a descrierilor, modificarile ulterioare fiind realizate editand fisierul php corespunzator din web/rules/tables. Modificarile de obicei sunt usor de realizat si pot fi efectuate rapid. La propunerea initiala de descriere facuta de Web Assistant, realizati imbunatatirile si/ sau corecturile necesare si sunteti gata in cateva clipe sa creati descrierea.

3.2.2. Reguli mysql (rule) – query-uri automate

In sectiunea curenta, subsectiunea 3.1.2. Pagini – componenta de prezentare (html) si de calcul (php), am definit "pagina" ca fiind conceptul central al aplicatiilor dezvoltate cu ajutorul phprel. Acestui concept ii corespunde un concept de o importanta similara, introdus inovativ de phprel, care interconecteaza o pagina cu baza de date: conceptul de "regula mysql", denumit in phprel "rule".

Intr-o aplicatie web, anumite pagini (in cazul aplicatiilor "backend", majoritatea) folosesc formulare pentru a introduce sau modifica datele dintr-o baza de date. Evident, respectivele date vor fi stocate in unul sau mai multe tabele, iar intre campurile formularului si campurile tabelelor va exista o corespondenta. In general, un formular modifica sau introduce o singura linie intr-un tabel la un moment dat, linie care poate fi indentificata precis printr-o conditie. Pentru acele pagini care au formulare de adaugare sau modificare de date, phprel introduce conceptul de "regula mysql" care sintetizeaza legatura logica intre formularul paginii si tabelele mysql implicate pentru a permite nucleului phprel sa realizeze query-urile necesare fara nici o interventie din partea dumneavoastra: selectarea datelor pentru a precompleta formularul de modificare, actualizarea datelor in urma trimiterii unui formular de modificare si inserarea datelor in urma trimiterii unui formular de adaugare. Extinzand conceptul, o pagina care listeaza o serie de linii dintr-un tabel, completate eventual cu date din alte tabele, va fi in mod evident

legata de tabelul principal, iar in cazul stergerii unei linii in baza acestei legaturi (care poate fi considerata o "regula mysql" simplificata, in care nu exista corespondenta intre campuri html si campuri mysql) nucleul phprel va realiza query-ul de stergere automat.

Concret, o regula a unei pagini va specifica:

- un tabel asupra careia pagina opereaza modificari
- o serie de campuri ale tabelului care vor fi folosite in pagina
- corespondentul lor in formularul html
- o conditie "where" care identifica in mod unic o linie din tabelul respectiv, linie cu care va opera pagina. In absenta conditiei, se va considera ca pagina insereaza o noua linie (neexistenta inca, deci fara conditie care sa o identifice). Daca exista conditie, se va considera ca pagina opereaza cu o linie deja existenta (altfel, conditia nu ar fi fost prezenta).

Regulile se vor descriere in web/rules/forms.mysql.rules.php. Pentru fiecare pagina se poate crea (conform exemplului din fisier) o regula folosind apeluri la functia phprel simpleRule:

- (array) $rule = simpleRule('(denumire tabel)', '(campuri mysql separate prin virgula)', '(campuri html separate prin virgula, corespondentele in ordine ale campurilor mysql)', '(conditie de identificare unica a liniei)');
    - denumirea de tabel poate fi si exacta, si in format phprel
    - campurile mysql pot lipsi in situatia in care tabelul are descriere activa in web/rules/tables/, fiind preluate campurile din descriere
    - campurile html pot lipsi, si vor fi preluate campurile mysql. Aceasta este varianta recomandata, in care campurile de formular si campurile din tabel au aceeasi denumire.
    - conditia trebuie sa fie o conditie "where" mysql valida care sa identifice in mod unic o linie. Se poate folosi conditia standard precalculata in variabila locala $where. Pentru a intelege modul in care valoarea acestei variabile este calculata, trebuie precizat mai intai ca variabila globala $edit va retine valoarea parametrului GET "edit" sau, daca acesta nu exista, a parametrului GET "view" (similar cu "edit", dar pentru vizualizare). Pentru o cerere http care nu contine nici "edit" nici "view", $edit va fi "false". Conventia regulilor mysql este ca daca o conditie este prezenta, ea se refera in mod necesar la o linie existenta, adaugarea unei linii noi se va face printr-o regula a carei conditie este vida. Varianta propusa si recomandata de phprel este ca parametrul GET "edit" sa contina id-ul liniei cu care se opereaza. Avand in vedere aceasta conventie si recomandarea phprel ca tabelele sa aiba cheia primara un camp "id", este usor de inteles semnificatia variabilei locale $where pusa la dispozitie pentru rapiditate in situatiile tipice.
    - fisierul php de reguli, web/rules/forms.mysql.rules.php, trebuie in mod necesar sa atribuie regula mysql stabilita variabilei globale $rule, in caz contrar subintelegandu-se ca paginii respective nu ii corespunde nici o regula.

O "regula simpla" este intalnita in peste 90% din cazurile in care veti defini reguli pentru pagini, si leaga pagina de un singur tabel mysql. Pentru a compune o "regula complexa", care leaga pagina de mai multe tabele mysql, se pot scrie mai multe apeluri succesive ale functiei simpleRule, rezultatele fiind retinute in variabile temporare de forma $rule1, $rule2, ..., din care se va genera la sfarsit, folosind functia php array_merge, variabila $rule = array_merge($rule1, $rule2, ...);

Un formular de editare sau de adaugare scris intr-o pagina pentru care s-a definit o regula corespunzatoare, va functiona de la sine selectand datele pentru precompletare, efectuand actualizarea sau

inserarea. Pe o pagina care are definita o regula, va functiona si actiunea de stergere a unei linii, indicata prin id in parametrul GET "del" – query-ul de stergere fiind scris si executat automat. O regula permite deci scrierea automata a celor patru query-uri de baza legate de o pagina care manipuleaza o linie de tabel. In plus, o pagina de editare poate fi convertita automat de phprel intr-o pagina de vizualizare: pentru detalii consultati sectiunea curenta, subsectiunea 3.4.11. Pagina de adaugare, editare, vizualizare.

Important: o regula mysql nu se va aplica si nici nu va fi calculata (fisierul web/rules/forms.mysql.rules.php nu va fi inclus) daca una din conditiile de mai jos nu este indeplinita:

- exista $_GET['edit'] zero sau numar pozitiv
- exista $_GET['view'] numar pozitiv
- exista $_GET['del'] numar pozitiv

Mai mult, actualizarea sau inserarea datelor nu va fi realizata decat daca, in plus fata de existenta $_GET['edit'], se trimite prin $_POST un parametru "goPost" cu valoare pozitiva. Este suficient sa includeti in formular un camp "goPost" folosind tag-ul <&frm.goPost> si phprel va inlocui automat campul cu un hidden initial de valoare zero, ulterior (in momentul validarii si trimiterii formularului) schimbata in 1.

Continuam exemplul inceput in sectiunea 1. phprel la prima vedere, adaugand o pagina "regula.mysql" (in practica, paginile vor avea un nume legat de functia pe care o indeplinesc):

- creem pages/regula.mysql.php, fara continut
- creem templates/regula.mysql.html cu urmatorul continut:

```
<&frm.form>
<table>
   <tr>
         <td class="text">User:</td>
         <td><&frm.user{text}></td>
   </tr>
   <tr>
         <td class="text">Parola:</td>
         <td><&frm.pass{pass}></td>
   </tr>
   <tr>
         <td class="text">Email:</td>
         <td><&frm.email{text}></td>
   </tr>
   <tr>
         <td class="text"> </td>
         <td><&frm.goPost><&frm.submit{value="Adauga"}></td>
   </tr>
</table>
<&endform>
```

- modificam templates/home.html adaugand un link:

```
<a href="regula.mysql/edit/0">Adauga utilizator</a>
```

- la un refresh al paginii, observam ca formularul functioneaza dar la trimitere baza de date nu este schimbata in nici un fel (valorile raman totusi precompletate in campuri, aceasta este o functionalitate a formularelor create cu phprel).

Definim o regula pentru pagina, iar formularul va functiona exact cum ne asteptam:

- modificam web/rules/forms.mysql.rules.php, adaugand o noua regula:

```
case 'regula.mysql':

    $rule = simpleRule('front.users', '', '', $where);

break;
```

- am completat in exemplul oferit in fisier denumirea paginii ("regula.mysql") si tabelul "front.users". Campurile nu este nevoie sa le completam deoarece tabelul are deja descriere, campurile html sunt identice campurilor mysql si deci nu necesita sa fie specificate in clar, iar conditia standard precalculata in $where se potriveste situatiei curente. Formularul este folosit deci pentru a adauga un nou utilizator in tabelul mysql "front_users".
- un refresh al paginii va demonstra rezultatul. Se observa ca parola a fost introdusa in baza de date in format "plain text", nu ca hash, aceasta aparenta problema va fi solutionata intr-o mod inovativ folosind functii de control, descrise in sectiunea curenta, subsectiunea 3.2.3. Functii de control, comportament standard, comportamente diferite pe pagina. Dupa ce ati adaugat un nou utilizator, se poate edita linkul adaugat in home.html, trecand o alta valoare pentru "edit", un id deja existent in tabel, de exemplu: "regula.mysql/edit/2". Se poate astfel edita respectivul utilizator (campurile de tip "password" nu vor fi insa precompletate).

Dar pentru fi intr-adevar "query-uri automate", phprel va pune la dispozitie o tehnologie inovativa care deduce regula mysql fara nici o interventie din partea dumneavoastra. "Assistant"-ul oferit in phprel va presupune pe paginile unde este necesar ce regula mysql trebuie declarata, si o va declara fara nici un efort din partea dumneavoastra. Formularele si listele vor functiona astfel de la sine, asa cum s-a putut vedea in mini exemplul parcurs in sectiunea 1. phprel la prima vedere. In general, in 90% din cazuri si 100% din cazurile des intalnite, veti folosi reguli definite automat de Assistant, rareori fiind necesar sa definiti manual regula mysql pentru o anumita pagina. Pentru mai multe detalii, consultati sectiunea curenta, subsectiunea 3.3.1. Predictii de reguli.

3.2.3. Functii de control, comportament standard, comportamente diferite pe pagina

Regulile mysql asigura o metoda automata sau asistata de a citi si scrie informatii in baza de date, si accelereaza dezvoltarea prin automatizarea functionarii formularelor si a altor elemente de pagina. Cu toate acestea, formatul in care datele sunt introduse de un utilizator nu corespunde formatului in care datele sunt retinute intr-o baza de date. Conversia din format "lizibil" in formatul intern al bazei de date este esentiala pentru folosirea practica a regulilor mysql. Inovatia esentiala adusa de phprel se refera tocmai la metoda prin care aceasta conversie este realizata: functiile de control ale operatiilor de citire/ scriere cu baza de date.

Functiile de control phprel sunt functii adaugate de dumneavoastra pentru a controla diferite procese automate sau asistate realizate de nucleul phprel. Aceste functii, daca exista, sunt folosite pentru transformarea unei valori intr-o alta valoare inainte de folosirea valorii intr-un anumit proces. In cazul procesarii formularelor folosind reguli mysql, presupunem ca avem o serie de campuri html (care corespund ca denumire campurilor mysql ale unui tabel "stiri"):

- titlu

- autor
- continut
- data

Daca folosim o regula mysql, datele asa cum au fost ele introduse de un utilizator in formular vor fi introduse in baza de date. Problema apare insa la campul "data" care in baza de date trebuie stocat in format unixtime. Solutia cea mai simpla este sa intervenim in procesul automat realizat de phprel, astfel:

- la salvarea datelor, in loc de procesul simplu prin care campului mysql ii corespunde valoarea campului html cu acelasi nume (camp[html] --- camp[mysql]), sa aiba loc o transformare a valorii printr-o functie de control: camp[html] --- functie control intrare --- camp[mysql]
- la citirea datelor pentru precompletarea formularului (in cazul modificarii unei linii deja existente), in locul procesului simplu prin care campului html ii corespunde valoarea campului mysql (camp[mysql] --- camp[html]), sa aiba loc o transformare a valorii printr-o functie de control: camp[mysql] --- functie control iesire --- camp[html]

Concret, in cazul de fata, functia de control "intrare" va converti din data in format lizibil "zi.luna.an" in unixtime, iar functia de control "iesire" va converti inapoi din unixtime in format lizibil "zi.luna.an".

Functiile de control au denumire variabila in functie de entitatea caruia i se aplica functia si pot sau nu sa existe, in functie de situatie. Daca exista, vor fi folosite, daca nu, procesul nu va fi influentat in nici un fel. "Denumirea variabila" contine denumirea exacta a entitatii asupra caruia se aplica functia. In cazul functiilor de control a citirii/ scrierii din baza de date, fiecare descriere de tabel poate contine functii de forma:

- std___input___(nume tabel in format mysql)___(nume camp) ($val)
- std___output___(nume tabel in format mysql)___(nume camp) ($val, $d)

Denumirea concreta a functiei va varia in functie de denumirea tabelului (scris in format mysql – deci cu underscore-uri, si nu cu puncte – in format phprel – deoarece o denumire de functie nu poate contine caracterul punct) si denumirea campului asupra caruia se aplica functiile.

Functia de control la scrierea datelor, std___input___(tabel)___(camp) (fiecare cuvant cheie este separat prin trei underscore-uri) primeste un singur parametru, valoarea initiala a campului html, si va intoarce valoarea finala a campului mysql.

Functia de control la citirea (si eventual afisarea) datelor, std___output___(tabel)___(camp) primeste doi parametrii, primul fiind valoarea initiala a campului mysql, al doilea fiind un array asociativ continand intreaga linie citita din tabelul mysql (cu valorile citite initial, acestea nu se vor modifica pe masura ce functiile "std___output" modifica valorile finale) si va intoarce valoarea finala folosita la afisare intr-un camp html (sau folosita la "afisare" in alte situatii).

In cazul de fata, pentru campul de "data" al presupusului tabel "stiri", adaugam in descrierea de tabel web/rules/tables/io.stiri.php doua functii:

```
function std___input___stiri___data ($val)
{
	return strtotime($val);
}
function std___output___stiri___data ($val)
{
	return date('d.m.Y', $val);
}
```

Important: functiile de control "input" si "output" sunt la origine destinate pentru a fi folosite cu formulare. In cazul listelor generate folosind modulul de liste inclus in phprel, functia de control care asigura flexibilitatea afisarii este cea de "transform": std___transform___(tabel)___(camp) similara functiei "output" dar aplicabila listelor. In cazul datelor selectate folosind functii rapide precum getData sau a combinatiei de functii alter(fetch(select())), functia de control aplicata este cea de "alter": std___alter___(tabel)___(camp) similara functiei "output" dar aplicabila query-urilor care nu se incadreaza in categoria "pentru formular" sau "pentru liste".

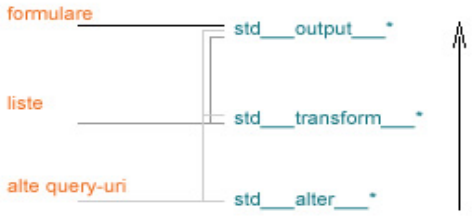


Fig. 3.2.3.a. Ordinea de aplicare a functiilor de control

Din motive de eficienta, functiile de control de importanta mai mare sunt folosite in absenta functiilor de control de importanta mai mica, dupa schema din figura. Astfel, formularele folosesc doar functia de "output", daca ea exista. Listele folosesc functia de "transform", dar daca aceasta nu exista, vor folosi functia de "output". Celelalte query-uri realizate prin functii phprel specializate folosesc functia de "alter", daca aceasta nu exista folosesc functia de "transform" iar daca nici aceasta nu exista, functia de "output".

De exemplu, pentru situatia descrisa, un apel getData('stiri.all'); va intoarce stirile cu data in format lizibil, functia getData incadrandu-se in categoria functiilor phprel pentru "alte query-uri decat formulare si liste", si cautand mai intai functia std___alter___stiri___data (inexistenta), apoi std___transform___stiri___data (inexistenta), si in final std___output___stiri___data (existenta). Daca in acest moment se scrie o functie std___alter___stiri___data care intoarce valoarea "8" indiferent de datele de intrare, formularul va functiona in continuare dar apelul la getData va avea ca valoare a campurilor "data" valoarea "8".

Eficienta descrierilor de tabel este acum explicata complet: comportamentul standard al campurilor va putea fi scris o singura data la crearea tabelului si se va aplica automat in toate paginile aplicatiei: codul va fi concis, usor de scris (o data functia definita, nu mai trebuie sa va faceti griji pentru manipularea corecta a datelor, prin simpla selectare a lor, datele vor fi in formatul corespunzator) si usor de intretinut (orice modificare ulterioara se va face simplu intr-un singur loc si se va propaga pe toate paginile).

Important: functia de control a scrierii (input) se aplica la procesarea formularelor (inserare sau actualizare) si la scrierea datelor folosind functia rapida writeData.

Functia de control a citirii/ afisarii (output) se aplica la precompletarea formularelor si, dupa caz, la generarea listelor folosind modulul de liste inclus in nucleul phprel precum si la selectarea datelor cu functia rapida getData sau folosind combinatia de functii alter(fetch(select())).

Functia de control a afisarii intr-o lista (transform) se aplica la generarea listelor folosind modulul de liste inclus in nucleul phprel precum si, dupa caz, la selectarea datelor cu functia rapida getData sau folosind combinatia de functii alter(fetch(select())).

Functia de control a afisarii "alter" se aplica la selectarea datelor cu functia rapida getData sau folosind combinatia de functii alter(fetch(select())).

In plus, sistemul de cos de cumparaturi foloseste functiile de control la scriere "input" pentru a scrie datele in tabelele interne phprel si metode rapide de selectare similare cu getData pentru citire/

afisare, aplicand regulile de prioritate a functiilor de control ale categoriei "alte query-uri/ query-uri realizate cu functii rapide phprel".

Functiile de control "std" definite in descrierile de tabele alcatuiesc "comportamentul standard" al diferitelor campuri ale tabelelor aplicatiei. In anumite situatii, pentru anumite pagini, comportamentul standard trebuie inlocuit cu un comportament specific paginii. Astfel, aveti posibilitatea supradefinirii oricarei functii de control cu o functie "locala" paginii:

Functiile de control destinate formularelor, "input" si "output" se pot supradefini in web/rules/io.functions.php, adaugand un nou "case" pentru pagina respectiva in interiorul caruia se pot scrie functii de forma:

- input___(nume camp) ($val)
- output___(nume camp) ($val, $d)

Tabelul nu se mai precizeaza, in rest parametrii si comportamentul functiei sunt similare functiilor omolog "std". **Important**: functiile vor fi folosite doar la procesarea formularelor (nu se vor aplica nici macar in cazul inserarii/ actualizarii datelor folosind writeData).

Functiile de control destinate listelor, de tip "transform" se pot supradefini in php-urile atasate paginii, construind noi functii de forma:

- transform___(denumire lista)___(denumire tabel in format mysql)___(denumire camp)
- transform___(denumire tabel in format mysql)___(denumire camp)

Parametrii si comportamentul functiilor sunt similare functiei omolog "std", a doua varianta de functie prezentata fiind folosita pentru a suprascrie comportamentul "std" pentru toate listele din pagina, in timp ce prima este folosita pentru a suprascrie comportamentul "std" sau comportamentul tuturor listelor din pagina pentru o singura lista specificata.

Functiile de control destinate functiilor phprel de scriere rapida a query-urilor se pot supradefini in php-urile atasate paginii, construind noi functii de forma:

- alter___(denumire camp)
- alter___(denumire tabel in format mysql)___(denumire camp)

Parametrii si comportamentul functiilor sunt similare functiei omolog "std".

In continuare sunt prezentate prioritatile de cautare a functiilor de control pentru fiecare categorie in parte:

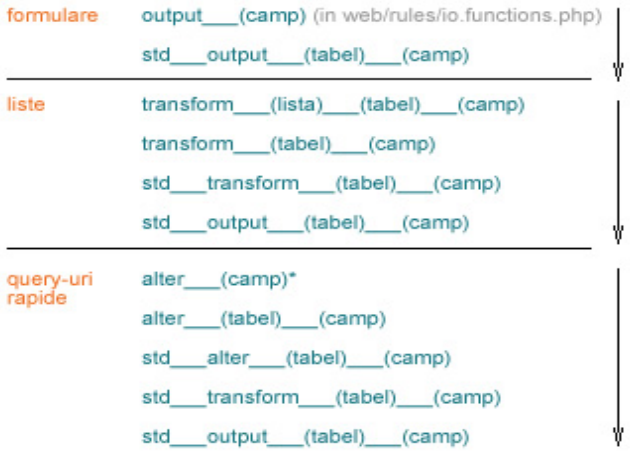


Fig 3.2.3.b. Ordinea de cautare a functiilor de control

In fiecare situatie o singura functie se va aplica, prima gasita de sus in jos. Daca nu poate fi gasita nici o functie de control, procesul nu va fi influentat si valoarea va fi pastrata identica cu cea initiala.

Important: functiile de control aplicate in cazul query-urilor care folosesc combinatia alter(fetch(select())) sau in situatii care folosesc indirect aceasta combinatie nu sunt 100% exacte deoarece functiile de control sunt aplicate intuitiv de phprel asupra unui array deja construit de rezultate (select realizeaza query-ul, fetch construieste array-ul de rezultate iar alter aplica transformarile pe baza functiilor de control). Nucleul phprel va reusi in majoritatea cazurilor sa presupuna corect care cheie din array a provenit din fiecare tabel, dar exista o situatie confuza care poate genera rezultate nedorite:

- query-ul se face pe doua sau mai multe tabele, din care exista doua tabele A si B care contin un camp cu denumire identica "c". Exista deci "A.c" si "B.c". Selectam campul "B.c" dar descrierea de tabel a lui A contine o functie std aplicata campului "c".
- apelul va modifica valoarea campului "c" prin functia std a tabelului A cu toate ca in query-ul initial am selectat "B.c" (dar nucleul este confuz si nu mai poate face diferenta)

Aceasta situatie nu apare in cazul folosirii functiei getData care previne automat confuzia denumirii campurilor si genereaza un array rezultat dezambiguizat. Combinatia alter(fetch(select())) este foarte folositoare in anumite situatii si este recomandata si incurajata folosirea ei, dar este bine sa aveti in vedere aceasta situatie pentru un eventual caz izolat in care confuzia s-ar putea produce.

Se recomanda folosirea functiilor de control specifice paginilor (acolo unde comportamentul se abate de la cel standard) atat datorita eficientei in scrierea codului (un cod scris cu astfel de functii va fi mult mai scurt si mult mai clar) cat si datorita posibilitatilor de extindere si mentenanta simplificata a paginii.

Important: functiile de control se pot aplica si asupra campurilor "virtuale"/ fictive ale unui query. De exemplu, urmatorul cod va functiona folosind functia de control:

```
function alter___statusplata ($val, $d)
{
	if ($d['achitat'] < $d['sumatotala'])
		return 'Plata incompleta';
	else
		return 'Plata completa';
}
$date = alter(fetch(select("id, produs, achitat, sumatotala, '1' AS statusplata", 'achizitii', '1'),
true)));
```

3.2.4. Functii de scriere rapida a query-urilor si prelucrarea rezultatelor

O descriere a functiilor phprel folosite pentru a construi rapid query-uri a fost realizata deja in sectiunea 2. Elemente de baza, subsectiunea 2.2. Accesul rapid la baza de date.

Functia select construieste un query de selectare a datelor:

- (result) select ($campuri, $tabele, $where, $order = '', $group = '', $limit = '', $having = '')
  - $campuri – campurile selectate, separate prin virgula. Se poate folosi doar sirul vid '' si phprel va selecta toate campurile tabelelor precizate, asa cum apar in descrierile de tabel. In cazul in care tabele au descrieri active si campurile nu au fost precizate iar campurile rezultate din descrieri s-ar suprapune, se vor selecta campurile in ordinea

prioritatii tabelelor (ordinea de precizare a tabelelor in parametrul $tabele), dupa cum urmeaza:
    - campurile cu denumire unica din toate tabelele vor fi selectate exact cu denumirea lor
    - campurile ale caror denumiri se suprapun vor fi selectate astfel:
        - primul camp intalnit cu denumirea respectiva (in ordinea tabelelor din $tabele) va fi selectat cu denumirea exacta
        - urmatoarele campuri vor fi selectate cu o denumire de forma "(tabel)___(camp)"
- $tabele – tabelele din care se selecteaza, separate prin virgula (se pot specifica si in format phprel)
- $where – conditia where, exact asa cum va apare in query (asadar, nu se pot folosi denumiri de tabele in format phprel), care va fi completata automat cu conditiile rezultate din $activeclause-urile tabelelor date
- $order – conditia order a query-ului mysql (exact asa cum va aparea in query), completata in situatia in care lipseste cu $defaultorder-ul descrierii tabelului principal (considerat a fi primul tabel specificat)
- $group – expresia de group a query-ului mysql (exact asa cum va aparea in query)
- $limit – conditia de limit a query-ului mysql, exact asa cum va aparea in query
- $having – conditia de having a query-ului mysql, exact asa cum va aparea in query
- toti parametrii sunt valorile efective ale conditiilor/ expresiilor pe care le reprezinta, fara a fi prefixate de identificatorii caracteristici (mai exact, fara a fi precedati de "LIMIT", "ORDER BY", etc.)
- functia va executa automat query-ul pe baza de date care trebuie, in baza specificatiilor din descrierile de tabel. In plus, pentru anumite query-uri (exista restrictii), va executa si query-uri pe mai multe baze de date simultan. Pentru detalii, consultati sectiunea 5. Elemente avansate, subsectiunea 5.1. Suport pentru multiple baze de date.

Functia fetch completeaza functia select fiind folosita pentru "citirea" rezultatelor unui query:

- (array de randuri / array asociativ / valoare) fetch ($result, $return_as_array = false)
    - fetch citeste rezultatele unui query (realizat printr-o functie care intoarce un rezultat mysql – "mysql result", de exemplu select(), run_query() sau mysql_query()) si intoarce rezultatul in functie de context:
        - daca s-a selectat un singur camp si query-ul a intors un singur rand, fetch intoarce unica valoare a campului
        - daca s-au selectat mai multe campuri si query-ul a intors un singur rand, fetch intoarce un array asociativ reprezentand randul (de forma camp => valoare)
        - daca respectivul query a intors mai multe randuri, fetch intoarce un array de randuri (de forma indice_rand => array (camp => valoare))
    - $result trebuie sa fie un rezultat mysql valid, in caz contrar intorcandu-se null (sau daca rezultatul este un array, se intoarce exact acel array – acest comportament permite folosirea transparenta a lui fetch cu select() chiar si in cazul query-urilor pe multiple baze de date simultan)

- $return_as_array – implicit pe false, comportamentul functiei fiind cel standard, descris mai sus. La setarea pe true, indiferent de numarul de randuri si/ sau de campuri intors de query, rezultatul va fi intors mereu sub forma un array de randuri.

Functia alter completeaza functia fetch fiind folosita pentru aplicarea functiilor de control asupra rezultatului unui query:

- (array de randuri) alter ($array_de_randuri)
  - alter intoarce un array de randuri asupra caruia s-au aplicat functiile de control (dupa metoda descrisa in sectiunea curenta, subsectiunea 3.2.3. Functii de control, comportament standard, comportamente diferite pe pagina)
  - array-ul dat ca parametru trebuie neaparat sa fie un array de randuri (deci de forma indice_rand => array (camp => valoare)), in caz contrar alter va intoarce un array vid.
  - rezultatul intors de alter poate fi direct folosit in inlocuirea unui tag de "loop"
  - mai multe detalii despre aplicarea functiilor de control au fost oferite in subsectiunea anterioara

Functia getData este cea mai rapida functie de construire a unui query de selectare a datelor, o serie de exemple fiind deja ilustrate in sectiunea 2. Elemente de baza:

- (array de randuri / false) getData ($instructiuni, $parametru = '')
  - $instructiuni – sir de caractere reprezentand o instructiune de selectare a datelor, de forma:
    - "(tabel).(functie)[(x, y)][.order(camp)][.group(camp)][.recursive[*nivel]]"
    - (tabel) este o denumire de tabel in format phprel (underscore-uri inlocuite cu caractere punct "."). Obligatoriu tabelul trebuie sa aiba descriere, in caz contrar intorcandu-se "false".
    - (functie) este una dintre functiile predefinite getData:
      - all
      - line
      - search
      - filter
      - toid
      - toname
      - leftsearch
      - options
      - searchoptions
      - filteroptions
    - optional, se poate adauga intre paranteze rotunde un parametru "limit" la functie. Denumirea functiei va fi urmata de un "limit" intre paranteze rotunde, dat prin doua numere x si y in mod similar limit-ului unui query mysql. De exemplu:

      all(0, 10) – va insemna selectarea primelor zece linii (mai exact, "tuturor liniilor" dar limitate de la linia cu indice 0, 10 linii.

      search(10,15)

      Evident, limit-ul va functia pe functiile care intorc mai mult de un rezultat.

- optional, se poate adauga ordonare adaugand la sirul instructiunii ".order(camp)" unde "camp" se inlocuieste cu campul de ordonare dorit, intre paranteze rotunde:

```
getData('produse.all(0, 10).order(denumire)');
getData('produse.all.order(denumire)');
```

  Ordonarea nu accepta expresii mysql continand functii, cu exceptia ordonarii aleatoare ".order(RAND())"
- optional, se poate grupa dupa un anumit camp, adaugand la sirul instructiunii ".group(camp)" unde "camp" se inlocuieste cu campul dupa care se grupeaza, intre paranteze rotunde:

```
getData('produse.all.group(categorie)');
```

- de asemenea, se poate adauga optional parametrul ".recursive" la sirul intructiunii, pentru a selecta datele recursiv si din tabelele relationate. Astfel, pentru fiecare rand rezultat din tabelul cerut se vor selecta si datele corespunzatoare din tabelele relationate. Pentru relatii de tip "nomen", valoarea denumita se va inlocui direct in campul indicator al relatiei. Pentru fiecare relatie se va crea o noua cheie la array-ul asociativ reprezentand randul, cu denumirea tabelului relationat (sau, daca aceasta denumire a fost folosita deja si relatia este relatie cu camp, cu denumirea de forma "(tabel).by.(camp)") si valoarea un array de randuri corespunzatoare liniei curente din tabelul relationat. Daca nu se specfica nivelul maxim, selectarea recursiva a datelor se face pana la nivelul standard specificat in cfgs/advanced/core.settings.inc.php prin $Tcfg_defaultrecursivelevel. Nivelul maxim se poate specifica adaugand "*" si un numar reprezentand nivelul la identificatorul "recusive". De exemplu:

```
getData('produse.all.recursive');
getData('produse.all.recursive*3');
```

  Nivelul reprezinta nivelul de adancime pana la care se merge recursive, tabelele relationate direct de tabelul cerut fiind considerate nivelul 1, tabelele relationate de tabelele relationate direct fiind considerate nivelul 2, etc. Pentru ca astfel de query-uri recursive pot genera rezultate imense, care sa consume resursele serverului http sau chiar mysql, exista o limitare de siguranta data in cfgs/advanced/core.settings.inc.php prin $Tcfg_safetyLimit, reprezentand numarul maxim de randuri selectate pentru query-urile recursive. La fiecare pas al recursivitatii, pentru fiecare tabel relationat se vor selecta maxim $Tcfg_safetyLimit linii. Cu toate acestea, se recomanda folosirea recursivitatii doar in situatiile in care este cu adevarat folositoare si se poate prezice cu o acuratete mare dimensiunea datelor rezultate.
- Ordinea de precizare a parametrilor ".order", ".group" si ".recursive" nu este relevanta.

- $parametru – este un parametru trimis functiei specifice getData solicitate, dupa cum urmeaza:
  - all – nu primeste parametru, parametrul este absent. Functia va intoarce toate liniile tabelului cerut (sau liniile specificate prin parametrul de "limit" atasat functiei)

        - line – primeste id-ul unei linii (valoarea cheii primare a liniei respective). Intoarce un array de randuri cu un singur rand (deci nu un array asociativ direct) corespunzator randului indentificat prin parametru.
        - search – primeste ca parametru o conditie "where" de cautare, exact asa cum va aparea ea in query-ul mysql. Intoarce un array de randuri corespunzator randurilor care valideaza conditia "where".
        - filter – primeste ca parametru o valoare sau un array simplu de valori corespunzatoare campului sau primelor N campuri (date ca N valori ale unui array simplu) specificate in $filterfield-ul descrierii de tabel. In cazul mai multor campuri (specificate in $filterfield separate prin virgula) se vor da valorile ca elementele unui array simplu (array(valoare1, valoare2, ...)) iar daca nu se doreste filtrarea dupa toate campurile, se pot da doar primele N valori, N fiind un numar arbitrar. Pentru o singura valoare, se poate da direct valoarea, fara a fi sub forma de array, corespunzatoare primului (sau singurului) camp din $filterfield. Intoarce un array de randuri corespunzatoare filtrarii campurilor dupa valorile date (folosind o relatie de forma: camp1='valoare1' AND camp2='valoare2' etc.). Necesita $filterfield, altfel va intoarce "false".
        - toid – primeste ca parametru valoarea expresiei date la $namefield in descrierea de tabel si intoarce ca rezultat valoarea cheii primara a primei linii care se potriveste $namefield-ului respectiv. Necesita specificarea unui $namefield in descrierea de tabel, altfel va intoarce "false".
        - toname – primeste ca parametru o valoare a cheii primare si intoarce valoarea expresiei data la $namefield corespunzatoare. Necesita specificarea unui $namefield, altfel va intoarce "false".
        - leftsearch – similar functiei "search", diferenta descrisa mai jos.
        - options – similar functiei "all", dar intoarce un array formatat de optiuni pentru un select-box construit folosind modulul de formulare inclus in nucleul phprel. Toate liniile tabelului (sau cele limitate prin conditia de "limit") vor fi convertite in optiuni cu value egal cu valoarea cheii primare si textul afisat propriu-zis valoarea expresiei specificate prin $namefield. Necesita $namefield altfel va intoarce "false".
        - searchoptions – similara functiei "search", dar construind optiunile pentru un select-box. Rezultatul este deci similar functiei "options". Necesita $namefield in descrierea de tabel, altfel va intoarce "false".
        - filteroptions – metoda de selectare similara functiei "filter" iar rezultatul similar functiei "options". Necesita $namefield si $filterfield, altfel va intoarce "false".
    - In plus, getData poate selecta date din mai multe tabele. Tabelele vor fi specificate in sirul instructiunii intre paranteze rotunde, in format phprel, in locul tabelului unic precizat standard:
        - "(tabel1, tabel2, tabel3).(functie)..." - functia se aplica in mod standard unei entitati, aceasta relatie de functie – entitate fiind desemnata prin caracterul

punct ("."): tabel.functie, iar acum functia se aplica unei entitati formate din mai multe tabele: (tabel1, tabel2, ...).functie.

Celelalte specificatii raman neschimbate, cu exceptia faptului ca singurele doua functii care pot fi folosite pe mai multe tabele sunt search si leftsearch. Diferenta intre cele doua consta in modul in care se realizeaza join-ul tabelelor: in primul caz, join-ul este de tip "inner join", realizat implicit prin conditii "where" de forma "tabel1.camp1=tabel2.camp2", in al doilea caz join-ul este de tip "left join". Conditiile de join si join-ul propriu-zis vor fi realizate automat de phprel, nu este necesar sa stabiliti conditiile de join in parametrul functiei.

Important: Assistant-ul phprel va descoperi singur legatura intre tabele chiar daca aceasta nu este directa. Atata timp cat exista un lant de legaturi care leaga tabelele specificate intre ele, si acele legaturi au fost precizate in descrierile de tabel, phprel va descoperi metoda de legare a tabelelor. De exemplu: pentru tabelele categorii, subcategorii, si produse (produse fiind legat doar de subcategorii, iar subcategorii de categorii), urmatorul exemplu va produce rezultatul dorit:

- getData('(produse, categorii).search', "pret>100");

- De asemenea, este important de retinut faptul ca getData construieste query-ul folosind functia select(), astfel ca primul tabel specificat va fi tabelul "principal" de la care se va completa ordonarea in cazul in care ea lipseste, campurile vor fi completate automat folosind regula de prioritate in cazul suprapunerii (descrisa la functia select), iar toate conditiile "where" date de dumneavoastra sau determinate implicit vor fi completate cu $activeclause-urile tabelelor solicitate.

Pana acum, am prezentat functiile predefinite pentru getData, dar dumneavoastra aveti posibilitatea definirii unor functii proprii pentru a le folosi cu getData daca este cazul. De exemplu, un query care se repeta des pe parcursul aplicatiei dar este prea complex pentru a fi scris folosind un getData simplu. In web/rules/data.access.php se pot crea functii cu denumire variabila de forma "get___(denumire)" care primesc un singur parametru si pot intoarce orice valoare. Scrieti o astfel de functie si dati-i o denumire oarecare, apoi ea poate fi apelata folosind getData astfel:

- $variabila = getData ('(denumire)', $parametru);
  - unde (denumire) este denumirea data functiei (retineti faptul ca orice caracter punct "." din denumirea data in sirul instructiunii getData va fi convertit in underscore pentru a fi compatibil cu denumirea de functie) iar $parametru va fi parametrul trimis functiei.

De exemplu, se poate crea o functie:

```
function get___produse_populare ( $getid )
{
        deturn alter(fetch(select('', 'produse', "rating > $getid"), true));
}
```

apelata apoi dupa cum urmeaza:

```
$foartepopulare = getData('produse.populare', 99);
```

Din convertirea caracterelor punct in underscore rezulta ca se pot supradefini si comportamentele standard ale functiilor predefinite dar numai pentru un tabel prestabilit:

```
function get___produse_search ( $getid )
{
        ...
```

```
}
```

Un apel la getData('produse.search', "denumire LIKE '%phprel%'") va trece acum prin functia creata in web/rules/data.access.php. Functiile definite de dumneavoastra au deci prioritate in fata functiilor predefinite.

Functia getfield permite obtinerea rapida (intr-o singura linie) a valorii unui camp dintr-un rezultat al unui query:

- $variabila = getfield ($camp, $array_asociativ_sau_de_randuri)
  - $camp – denumirea campului dorit
  - $array_asociativ_sau_de_randuri – array-ul care contine rezultatul query-ului (sau un array de randuri/ asociativ oarecare). Daca array-ul contine direct cheia specificata prin $camp, va fi intoarsa valoarea respectivulei chei, altfel, se va intoarce valoarea corespunzatoare primului rand si cheii specificate. Daca nici aceasta valoare nu exista, se va intoarce sirul vid.

Functia shift trunchiaza/ "deplaseaza" un array, dupa cum urmeaza:

- shift ($array)
  - daca $array este un array de randuri (chiar daca are un singur rand sau mai multe), shift intoarce array-ul asociativ corespunzator primului rand.
  - daca $array este un array asociativ, shift intoarce valoarea primei chei a array-ului.
  - daca $array nu este un array, intoarce $array.
  - De exemplu, in cazul folosirii functiei "line" predefinita in getData, daca avem nevoie de array-ul asociativ corespunzator randului indicat prin id, vom scrie:

```
$line = shift(getData('produse.line', $id));
```

Functia writeData permite scrierea rapida a query-urilor de inserare sau actualizare si functioneaza similar functiei getData:

- (id insert / array id-uri insert / true / false) writeData ($sir_instructiuni, $parametru)
  - $sir_instructiuni – este de forma: "(tabel).(functie)", unde (tabel) este denumirea tabelului in format phprel care obligatoriu are descriere de tabel iar (functie) este una din functiile predefinite:
    - insert – realizeaza inserarea uneia sau mai multor linii in tabel. Primeste ca parametru un array asociativ reprezentand un rand (din care se va ignora cheia primara, daca ea exista) care va fi inserat si al carui id de inserare va fi intors sau un array de randuri (din care se vor ignora valorile atribuite cheii primare, daca exista) care vor fi inserate si ale carui id-uri vor fi intoarse in ordine sub forma unui array simplu.
    - update – realizeaza actualizarea unei linii, sau mai multor linii. Similar functiei insert dar primeste pentru fiecare rand valoarea cheii primare folosita pentru a identifica randul care trebuie actualizat (se actualizeaza valorile celorlalte campuri specificate). Intoarce true pentru fiecare rand actualizat sau false daca nu s-a actualizat randul.
    - mixed – un hibrid intre "insert" si "update", pentru a insera sau actualiza mai multe linii simultan, acolo unde exista cheie primara se actualizeaza, acolo unde nu exista se insereaza.
    - delete – primeste ca parametru un id (valoarea unei chei primare numerice) sau o conditie "where" oarecare si sterge liniile din tabel corespunzatoare conditiei

sau cheii primare date. Intoarce true daca liniile au fost sterse sau false daca nu au fost sterse.
Important: writeData functioneaza cu tabele ale caror chei primare sunt numerice, auto-incrementate. Pentru inserarea valorilor cheilor primare care nu se incadreaza in acest model, folositi query-uri scrise obisnuit cu run_query. In plus, writeData nu valideaza faptul ca o cheie primara data exista in tabel, si va incerca sa actualizeze linia cu toate ca ea nu exista.

- $parametru – parametrul trimis functiei, dupa descrierea realizata fiecareia.

Nu in ultimul rand, se recomanda executia query-urilor complicate (care nu se pot realiza folosind una din funtiile descrise mai sus) cu functia run_query (echivalentul functiei mysql_query) si obtinerea id-ului de inserare cu get_insert_id() (echivalentul functiei mysql_insert_id()) pentru a va asigura ca tabelele virtuale (stabilite prin $virtual in descrierea de tabel), cache-ul si debugging-ul functioneaza corect:

- (mysql result) run_query ($query_string)
- (int / null) get_insert_id ()

3.2.5. Stergerea recursiva a liniilor pe baza relationarii tabelelor

In phprel, stergerea unei linii dintr-un tabel se poate face in trei moduri (exceptand, bineinteles, un query obisnuit de stergere):

- folosind actiunea de stergere dintr-o lista construita folosind modulul de liste inclus in nucleul phprel
- folosind regulile mysql pentru a sterge linia cu id-ul (valoarea cheii primare) trimis prin parametrul GET "del"
- folosind functia writeData

In toate trei cazurile, nucleul phprel va sterge mai intai linia sau liniile specificate, apoi pentru fiecare linie stearsa va cauta legaturile de tip "child" ale liniei respective si le va sterge si legaturile de tip "linked" ale liniei respective si le va "desface". Astfel, dupa stergerea liniilor specificate, se vor sterge recursiv liniile "child" (respectiv liniile din alte tabele relationate de tabelul curent "as.parent" printr-un camp care are valoarea egala cu id-ul liniei sterse in prezent) iar pentru fiecare linie stearsa (direct sau indirect) toate legaturile "linked" (fara camp) catre un tabel (care este legat inapoi de tabelul curent ca "linked" cu camp) vor fi parcurse si "desfacute" (setand campul respectiv din tabelul corespondent la valoarea zero).

In plus, inainte de stergerea fiecarei linii, daca functia de control de forma "delete___(denumire tabel in format mysql)" exista in descrierea de tabel se va apela acea functie pentru validarea stergerii. Acea functie poate indeplini orice sarcini conexe cu stergerea unei linii (al carei id il primeste ca parametru) si intoarce la final "true" daca linia poate fi stearsa sau "false" daca linia nu trebuie stearsa. Se asigura astfel controlul total asupra procesului de stergere.

Stergerea recursiva automata va functiona doar daca tabelul are descriere activa in web/rules/tables/ si va fi executata automat pornind de la linia sau liniile solicitate pentru stergere pana cand nu mai raman legaturi de tip "child" sau "linked" care sa fie sterse sau desfacute (printr-un procedeu recursiv).

3.3. Assistant

Cel mai important modul inclus in nucleul phprel este "Assistant"-ul, care foloseste toate celelalte module precum si algoritmi inovativi de analiza, corelare si interpretare a surselor scrise de dumneavoastra pentru a realiza automat sau a va asista in realizarea operatiilor sau proceselor necesare construirii aplicatiei. Modulul actioneaza ca raspuns automat la informatii sau precizari oferite direct sau indirect de dumneavoastra pe masura ce scrieti sursele si intervine in etapa finala de executie a unei pagini, dupa ce dumneavoastra ati avut ocazia sa realizati sau sa anulati orice proces de pe pagina respectiva. Actiunile realizate direct de dumneavoastra au deci prioritatea cea mai mare, si doar procesele netratate vor fi preluate de Assistant si rezolvate. In plus, orice actiune realizata automat sau asistat poate fi controlata prin anumite functii de control. Rapiditatea dezvoltarii aplicatiilor bazate pe phprel provine in primul rand din sprijinul activ acordat de modulul "Assistant".

Assistant-ul se ocupa in principal de inlocuirea tag-urilor speciale phprel dar are ca functii importante si predictia regulii mysql atasate paginii si accesul asistat la baza de date. Aceasta ultima componenta, desi deriva din algoritmii care alcatuiesc "nucleul" modulului de asistenta, a fost externalizata in modulul de acces la baza de date pentru a fi disponibila chiar si in cazul in care Assistant-ul a fost dezactivat si pentru a pastra specializarea fiecarui modul. Am descris activitatea acestei componente cand am discutat despre autocompletarea query-urilor, realizarea automata a join-urilor de tabele si selectarea automata a bazei de date in functie de tabelele specificate in query. In urma acestei externalizari a unei parti din serviciile oferite de Assistant catre modulul de acces la baza de date, modulul de asistenta se ocupa exclusiv de predictia de reguli si de inlocuirea tag-urilor.

3.3.1. Predictii de reguli

Intr-o aplicatie bazata pe phprel, unul din cele mai importante ajutoare acordate de modulul de asistenta este determinarea si definirea automata a regulii mysql. Regulile mysql au fost prezentate in sectiunea curenta, subsectiunea 3.2.2. Reguli mysql (rule) – query-uri automate. In cazul in care ar putea fi nevoie de o regula mysql pentru pagina curenta (mai exact, daca exista un parametru GET "edit", "view" sau "del" sau daca pentru a inlocui un tag special phprel este nevoie de determinarea unei legaturi intre pagina si baza de date) si o regula mysql nu a fost definita explicit de dumneavoastra, asistentul virtual inclus in phprel va analiza din punct de vedere "semantic" sursele paginii curente (mai exact, sursa html a elementului "content" care urmeaza a fi incarcat) si contextul in care se afla (cererea http, datele trimise prin POST) pentru a determina un tabel sau o serie de tabele care se leaga de contextul si/ sau pagina actuala. Acel tabel sau tabele vor fi folosite in functie de situatie pentru a genera o regula mysql de actualizare/ inserare date, de stergere date, de vizualizare date sau o regula mysql "formala" folosita pentru a inlocui anumite tag-uri phprel (de regula, liste sau formulare principale).

Asistentul realizeaza predictia regulii imediat dupa verificarea existentei unei reguli definite de dumneavoastra – in situatia in care o regula este explicit solicitata printr-unul din parametrii GET corespunzatori – sau in momentul inlocuirii unui tag special care necesita informatii suplimentare despre legatura intre pagina si baza de date (de exemplu, o lista principala sau un formular principal) si o regula nu a fost inca declarata fie de dumneavoastra fie de asistent.

Concret, se pot crea formulare sau liste care sa functioneze pur si simplu, prin insasi crearea lor, datorita faptului ca ele vor fi corelate automat cu baza de date. Asistentul va folosi diferite metode de determinare a tabelului sau tabelelor pe baza carora se va defini regula mysql, toate bazandu-se in primul rand pe descrierile de tabele iar in al doilea rand pe cererea http si eventual datele trimise prin POST, precum si sursa html a paginii (mai exact, a elementului "content"). In cazul in care algoritmii de predictie nu reusesc sa determine un tabel sau determina incorect tabelul necesar definirii regulii, se poate indica in clar tabelul care trebuie folosit (eliminand aproape complet necesitatea definirii manuale a unei reguli – cu exceptia cazurilor complicate in care tabelul indicat de regula difera cu toate ca pagina ramane constanta) adaugand un atribut "table" la unul din tag-urile phprel folosite care accepta atributul "table" (de exemplu, la formularul principal sau lista). Asistentul va considera prima valoare a unui atribut "table" gasit in sursa ca fiind tabelul folosit pentru regula mysql, deci in situatia in care aveti mai multe atribute "table" specificate pe mai multe elemente sau in cazul in care nu dispuneti de un element pentru care se poate specifica atributul "table", se poate crea un tag vid <&{table="(denumire tabel in format phprel)"}> continand atributul "table" pe prima linie a sursei html. Specificarea in clar a tabelului folosit pentru definirea regulii va determina si o definirii automata a regulii mai rapida, avand prioritate in fata algoritmilor de predictie care nu vor mai fi apelati. In foarte multe situatii, insa, tabelul va fi identificat corect foarte repede de catre asistentul phprel.

Se recomanda specificarea in clar a tabelului doar atunci cand aceasta specificare se face natural (adaugand un atribut "table" pe o lista de exemplu, pentru a evidentia faptul ca lista va afisa continut luat din respectivul tabel) sau cand este absolut necesara pentru functionarea corecta a paginii (asistentul nu a determinat deloc sau a determinat incorect tabelul).

Algoritmii de predictie vor lua in considerare denumirea paginii, denumirile campurilor html din sursa, denumirile campurilor unei liste din sursa precum si denumirile eventualilor parametrii POST.

3.3.2. Inlocuirea automata a tag-urilor phprel

Asistentul are ca principal scop inlocuirea automata a tag-urilor phprel din template-ul paginii. Aceasta inlocuire automata va permite, pentru tag-urile standard acceptate de motorul de template-uri, o dezvoltare mai rapida prin omiterea apelurilor catre functiile de inlocuire si permite in plus introducerea unor tag-uri speciale cu o semnificatie aparte pentru asistent, care vor tine locul unor procese complexe construite si executate de asistent fara ca dumneavoastra sa scrieti o linie de cod sau cu minimum de linii scrise. In final, scrierea unei pagini web este mult simplificata, numarul de linii de cod php este redus aproape de zero, reducand atat timpul de dezvoltare cat si numarul de bug-uri si deci timpul de verificare si corectare ulterioara, iar aplicatia este mult mai robusta, mult mai dinamica, mult mai ergonomica si mult mai avansata din punct de vedere al tehnologiilor folosite pentru ca asistentul va implementa de fiecare data un anumit proces sau operatie corect si complet printr-o tehnologie A.J.A.X. de exemplu, sau o alta tehnologie propice, in timp ce un programator ar putea, presat fiind de un termen limita, sa sara peste anumite etape din implementarea aceluiasi proces sau sa-l realizeze intr-o maniera simplificata. Selectarea unor optiuni din select-box-uri de tip "parent-child" va fi realizata mereu prin tehnologie de tip A.J.A.X., incarcarea detaliilor suplimentare intr-o lista va fi mereu realizata prin tehnologie de tip A.J.A.X., query-urile automate de filtrare a unei liste vor fi intotdeauna scrise pentru a preveni un atac de tip "sql injection", paginile de continut vor putea fi incarcate transparent prin A.J.A.X. de fiecare data, timpii de incarcare vor fi mereu optimizati folosind caching iar pagina va fi actualizata mereu prin adaptarea si

monitorizarea activa a cache-ului. Acestea sunt doar cateva exemple dintr-o lunga lista de actiuni realizate fara efort de asistentul inclus in phprel, de fiecare data cand este nevoie. Si fara nici o linie de cod din partea dumneavoastra. Veti petrece asadar mult mai putin timp dezvoltand o aplicatie care se va demonstra a fi mai fiabila, mai sigura si mai impresionanta pentru utilizatorii finali.

Inlocuirea tag-urilor standard are in vedere:

- inlocuirea tag-urilor simple de variabile
- inlocuirea tag-urilor de bucla
- includerea elementelor de pagina conform tag-urilor de incluziune

Inlocuirea tag-urilor speciale are in vedere:

- inlocuirea tag-urilor simple speciale:
    - tag-uri ce desemneaza functii
    - tag-uri ce desemneaza cereri asincrone javascript + xml (A.J.A.X. construit rapid)
    - tag-uri ce desemneaza directive pentru sistemul de cos de cumparaturi
    - tag-uri ce desemneaza parti de URL destinate navigatiei pe site (linkuri de intoarcere)
    - tag-uri ce desemneaza includerea codului javascript necesar folosirii unui calendar pentru un camp html
    - tag-uri conditionale
    - tag-uri de formular
    - tag-uri de liste
    - tag-ul special de dezactivare cache
- inlocuirea tag-urilor de bucla speciale:
    - continutul cosului de cumparaturi
    - comenzile efectuate de un utilizator
    - continutul unui tabel prin functii getData
    - continutul unui tabel prin denumire
- inlocuirea codurilor scurte javascript

3.3.3. Variabile globale

Asistentul phprel va corela denumirea tag-urilor cu variabile din scope-ul global, inlocuindu-le automat cu valorile corespunzatoare. Astfel, pentru a a inlocui o variabila in template, scrieti in locul dorit un tag de variabila cu exact aceeasi denumire. Tag-urile de variabile sunt tag-uri simple de forma <&(denumire)>. La intalnirea respectivului tag, asistentul va cauta in scope-ul global o variabila cu denumirea identica cu denumirea tag-ului, si in cazul in care ea exista, ii va inlocui valoarea in template.

Tag-urile de variabile pot fi imbricate, pe oricate nivele este nevoie, inclusiv cu alte tag-uri phprel. Asistentul le va inlocui in ordinea in care le gaseste si pe masura ce dependintele de alte tag-uri sunt rezolvate, si in cazul in care o corespondenta cu scope-ul global nu este gasita, va astepta sa vada in ce masura o variabila cu numele necesar este creata intr-un include secundar. Se asigura astfel o flexibilitate maxima a inlocuirii tag-urilor.

In plus, tag-urile de variabila pot contine referinte la chei de array-uri, indiferent de numarul de dimensiuni. Se pot inlocui deci tag-uri de forma <&(denumire)['(cheie1)'][(cheie2)]['(cheie3)']...> cu conditia existentei unui array global "(denumire)". Tag-ul va fi considerat ca "existent" daca array-ul

exista, iar o cheie inexistenta va fi inlocuita cu sirul vid. In cazul in care array-ul nu exista, se va astepta pentru o eventuala declarare intr-un include secundar, dupa cum am precizat mai sus.

Variabilele se declara implicit global din php-urile corespunzatoare header-ului, content-ului si footer-ului, iar din orice alta functie sau include secundar (cu exceptia celor incluse de dumneavoastra in scope-ul global) vor fi declarate ca globale, folosind constructia "global" sau variabila $GLOBALS.

Se considera prin extindere ca fiind variabile globale si se inlocuiesc in acelasi timp urmatoarele variabile speciale:

- referitoare la cosul de cumparaturi:
  - cart.add
  - cart.more
  - cart.less
  - cart.remove
  - cart.order
  - cart.total
  - cart.items
  - cart.*
- referitoare la cereri asincrone javascript + xml
  - async{...}
- referitoare la navigare:
  - nav.generate
  - nav.back
  - nav.here
  - nav.session.back
  - nav.session.jump
  - nav.session.jumpto{...}
- referitoare la functii:
  - apel catre una din functiile permise, cu nici un parametru sau pana la cinci parametrii, in functie de situatie (apelul este identificat prin prezenta caracterelor "(" si ")" semnificand o functie)
  - functiile permise sunt:
    - pentru siruri de caractere: ltrim, rtrim, trim, str_ireplace, str_replace, strlen, stristr, strstr, strpos, stripos, strtolower, strtoupper, substr, ucfirst, ucwords, md5
    - pentru array-uri: is_array, serialize, count
    - pentru date calendaristice: date, strtotime, mktime
    - pentru fisiere: file_exists
    - functii phprel: rewrite___output, getUser, getUserDetails, getSesid, cartTotal, cartItems, shift, getfield, getInputValue, locate, contentByAsync, optionsByAsync, fieldByAsync, a, pageByAsync, getBackUser, checkCredentials
- referitoare la cache:
  - <&cache off> – dezactiveaza salvarea in cache pentru pagina curenta

Variabilele referitoare la cosul de cumparaturi, cereri asincrone si navigare vor fi descrise in subsectiunile urmatoare (subsectiunile 3.7. Cereri asincrone javascript + xml, 3.8. Cos de cumparaturi si preluarea comenzilor, 3.9. Navigare). Variabilele apel la functii se vor scrie in forma: <&functie()> sau

<&functie(parametru)> sau <&functie(parametru1, parametru2, ...)> unde "functie" este denumirea functiei apelate, dintre cele permise. Un apel la o functie nepermisa va intoarce "false" si va fi inlocuit cu sirul vid. Parametrii functiei pot fi inlusiv tag-uri phprel, dar daca tag-ul intoarce un sir de caractere trebuie inclus intre apostroafe. De exemplu: <&str_replace('<&a>', '<&b>', '<&c>')> este o constructie valida. Similar, <&date('d.m.Y', <&timestamp>)> este o constructie valida. In plus, parametrii functiei pot fi variabile php, de exemplu: <&str_replace($a, $b, $c)> dar nu pot fi expresii php complexe, de exemplu apeluri de functii, operatii aritmetice, logice, de atribuire, sau orice expresie valida php care nu se rezuma la o simpla variabila. Variabila poate fi inclusiv o referinta la o cheie a unui array, de exemplu <&functie($a['cheie1']['cheie2'][cheie3])>, si va fi cautata in scope-ul global.

Functiile sunt restrictionate la lista specificata pentru a delimita precis componenta de calcul (php) de cea de prezentare (html). In cazul in care orice functie phprel ar fi fost permisa, s-ar fi putut face foarte multe calcule si operatii direct in html.

Inlocuirea variabilelor globale are loc in doua etape: o etapa dinamica, in care toate categoriile de tag-uri sunt inlocuite intr-o ordine optim calculata pentru a asigura o inlocuire completa si rapida in aproximativ orice situatie, si o etapa de "curatare" care are loc la final si care are ca scop finalizarea inlocuirii tag-urilor si inlocuirea cu siruri vide a tag-urilor care nu au fost inca inlocuite (indiferent de natura si provenienta lor). In etapa dinamica tag-urile care nu pot fi inca rezolvate, din cauza dependentei sau inexistentei datelor, vor fi intarziate pana cand pot fi inlocuite. In etapa de curatare, orice tag phprel de forma <&(sir de caractere)> cu exceptia tag-urilor de variabile post-incarcate si tag-urilor de resurse cache vor fi inlocuite cu valoarea corespunzatoare sau, in cazul in care acea valoare nu exista, cu sirul vid.

3.3.4. Bucle

Asistentul phprel va cauta si va inlocui tag-urile "loop" din template-ul paginii, cautand denumirea buclei printre array-urile definite in scope-ul global. In cazul in care un array corespunzator nu a fost gasit, phprel va verifica daca bucla este de fapt un apel la functia getData, sau denumirea unui tabel (in format phprel) sau o bucla speciala. Astfel, sunt inlocuite urmatoarele bucle:

- cele carora le corespunde un array cu aceeasi denumire in scope-ul global
- cele care contin in denumire constructia "::", fiind un apel catre o functie getData, de forma:
    - (tabel in format phprel).(functie)::(parametru), mai exact "<&loop "(tabel in format phprel).(functie)::(parametru)">" unde (functie) este una din functiile recunoscute de getData iar (parametru) este parametrul trimis functiei (nu este nevoie sa inludeti parametrul intre apostroafe chiar daca este sir de caractere). In cazul functiei "all" sau "options", chiar daca in mod normal nu primesc parametrii, va trebui totusi sa trimiteti parametrul "1", scriind "all::1" pentru ca asistentul sa recunoasca faptul ca este vorba despre o functie getData. Pentru mai multe detalii, consultati sectiunea curenta, subsectiunea 3.2.4. Functii de scriere rapida a query-urilor si prelucrarea rezultatelor, functia getData.
- cele ale caror denumiri incep cu "cart.contents" sau "cart.orders". Inlocuirea se va face cu continutul cosului de cumparaturi respectiv comenzile efectuate de utilizatorul curent, prin apel la functiile cartContents() si cartOrders(). Daca denumirea este urmata de caracterul "*" si un numar, selectarea datelor se va face recursiv pana la nivelul specificat prin numar, sau pana la nivelul standard (vezi sectiunea curenta, subsectiunea 3.2.4. Functii de scriere rapida a query-urilor si

prelucrarea rezultatelor, functia getData pentru detalii) in cazul in care nu este dat nici un numar. Pentru mai multe detalii, consultati sectiunea curenta, subsectiunea 3.8. Cos de cumparaturi si preluarea comenzilor.

- cele care au ca denumire denumirea unui tabel in format phprel. Tabelul va avea obligatoriu descriere activa in web/rules/tables/. Inlocuirea acestor bucle se face prin analizarea parametrilor GET si POST si construirea unei conditii "where" de filtrare, apoi selectarea liniilor corespunzatoare. Conditia se construieste pe baza identitatii unui parametru cu un camp din tabel si aplicarea operatorului "AND" intre legaturile astfel stabilite. De exemplu, pentru un tabel cu campurile "id, id_categorie, denumire", daca va exista un parametru GET "id_categorie" cu valoarea "5" va rezulta conditia "id_categorie='5'". In cazul in care nici o legatura nu poate fi stabilita intre denumirile parametrilor GET si POST si denumirile campurilor tabelului, toate liniile vor fi selectate si inlocuite in template. Pentru detalii privind descrierile de tabel, consultati sectiunea curenta subsectiunea 3.2.1. Descrierea tabelelor

Prioritatea de determinare va fi exact in ordinea data mai sus. De exemplu, daca exista si un array si un tabel care sa corespunda denumirii buclei, va fi inlocuit array-ul.

Inlocuirea unei bucle presupune parcurgerea unui array rand cu rand si, pentru fiecare rand, parcurgerea cheilor randului respectiv, cautarea lor in continutul buclei si, daca sunt gasite, inlocuirea tag-urilor respective cu valoarea cheii. Am vazut deja ca tag-urile pot contine denumiri simple de chei, de exemplu <&nume>, dar si referinte la (sub)array-uri ale randurilor, de exemplu <&produs['pret']>. Am vazut de asemenea ca o cheie trebuie neaparat sa existe in primul rand pentru a putea fi folosita in alte randuri. Este important de stiut ca in inlocuirea automata a tag-urilor, asistentul acorda prioritate tag-urilor din interiorul unui loop in fata tag-urilor de variabile globale. De exemplu, daca intr-un loop exista cheia "nume" dar exista in acelasi timp si variabila globala $nume, valoarea inlocuita va fi cea din loop. In exteriorul loop-ului insa, valoarea inlocuita va fi evident cea a variabilei globale.

De asemenea, asa cum am vazut deja, numele unei bucle trebuie sa fie unic in template-ul paginii, in cazul inlocuirii automate el trebuie sa fie unic indiferent de elementul paginii. Astfel, doua loop-uri cu denumiri identice vor fi inlocuite dupa cum urmeaza:

- prima intalnita va fi inlocuita corect
- urmatoarea bucla nu va mai fi inlocuita ca loop, dar va fi preluata la final de inlocuirea variabilelor globale, in etapa de curatare, si se vor sterge toate tag-urile, ramanand doar continutul html inclus printre tag-uri.

Important: nu folositi bucle cu denumiri identice pe aceeasi pagina. In cazul in care continutul aceleiasi variabile trebuie inlocuit in doua bucle, folositi doua variabile:

```
$a = (continut);
$b = $a;
<&loop "a"> ... <&loop "b">
```

De retinut faptul ca si buclele cu denumiri speciale trebuie sa respecte aceleasi reguli, denumirea ramane totusi o denumire, chiar daca asistentul o interpreteaza ca pe un apel la getData. De exemplu, pentru inlocuirea tuturor liniilor din tabelul de produse, bucla rezultata <&loop "produse.search::1"> va trebui sa fie unica. Pentru a inlocui in doua locuri acelasi continut, folositi in al doilea loc o expresie similara <&loop "produse.search::1 AND 1">.

3.3.5. Tag-uri conditionale

Tag-urile conditionale sunt folosite pentru afisarea sau ascunderea anumitor portiuni de template in functie de situatiile specifice in care se afla utilizatorul. Sunt interpretate si inlocuite doar de asistent, si pot fi asimilate instructiunilor conditionale dintr-un limbaj de programare. Structurile acceptate sunt:

```
<&if{(conditie)}>
        ...
<&endif>
```

si

```
<&if{(conditie)}>
        ...
<&else>
        ...
<&endif>
```

Ca si in cazul altor tag-uri inlocuite de asistentul phprel, sintaxa trebuie respectata intocmai, de exemplu caracterul "&" trebuie sa urmeze imediat caracterului "<" (nu se accepta, de exemplu, un spatiu intre ele), cuvantului "if" trebuie sa ii urmeze imediat "{", iar caracterului "}" trebuie sa-i urmeze imediat ">". Aceasta strictete permite o inlocuire mai rapida a tag-urilor, cu doar o foarte mica pierdere a flexibilitatii.

Conditia va fi exprimata in limbaj php, si poate contine orice constructie valida php, cu cateva restrictii impuse pentru respectarea delimitarii intre componenta de calcul (php) si componenta de prezentare (html):

- nu se poate folosi operatorul de atribuire (=)
- se pot apela doar functiile php permise, aceleasi cu cele permise in cazul tag-urilor de variabila globala. Pentru detalii, consultati sectiunea curenta, subsectiunea 3.3.3. Variabile globale.
- variabilele referite in conditie sunt cautate in scope-ul global
- nu se poate referi variabila $GLOBALS, fiind implicit folosite variabilele globale

In plus, o conditie inexistenta va evalua la "false" iar conditia "!" neurmata de nimic va evalua la "true".

O facilitate importanta care decurge din modul de lucru al asistentului este aceea ca se pot folosi alte tag-uri speciale phprel, de obicei variabile globale sau variabilele unei bucle, in conditia if-ului. In cadrul unui loop, de exemplu, se poate folosi `<&if{<&are_poza>}>` unde "are_poza" este cheie a randurilor buclei, cu valori true sau false. Asa cum am vazut deja, tag-urile phprel care sunt inlocuite cu siruri de caractere vor trebui incluse in apostroafe, de exemplu: `<&if{'<&nume>' == 'phprel'}>` unde "nume" este o cheie a randurilor unei bucle. Se poate evident extinde functionalitatea, `<&if{strtoupper('<&nume>') == 'PHPREL'}>` sau `<&strtoupper(<&if{<&prenume>}>'<&nume> <&prenume>'<&else>'<&nume>'<&endif>)>` unde "nume" si "prenume" sunt chei ale randurilor unei bucle. De asemenea, se pot folosi si variabile globale sau orice alt tag special phprel.

Tag-urile conditionale se pot imbrica in orice fel, si de asemenea se pot imbrica cu alte tag-uri phprel.

3.3.6. Tag-uri de includere a elementelor

Asistentul phprel va cauta tag-urile de includere a elementelor secundare si va include automat respectivele elemente. Astfel, pentru orice tag de forma <&include "(denumire)" /> sau <&include "(denumire)">, daca fisierul web/templates/(denumire).html exista, acesta va fi inclus automat de phprel. In plus, daca fisierul web/pages/(denumire).php exista, va fi inclus in momentul descoperirii tag-ului de incluziune. Inluderea php-ului atasat elementului se face intr-un scope local, deci orice variabile folosite in php care trebuie sa fie globale (de exemplu, pentru a fi inlocuite automat in elemente ale paginii) vor trebui declarate global. Asistentul va declara automat ca fiind globale acele variabile care par a fi folosite. Pentru detalii, consultati sectiunea curenta, subsectiunea 3.3.9. Auto-globalizarea variabilelor.

Astfel, pentru a include un element de pagina, dupa ce ati creat html-ul si eventual php-ul (doar daca este necesar), nu trebuie decat sa scrieti in locul in care doriti sa includeti elementul tag-ul de incluziune corespunzator.

Important: template-urile vor fi procesate intr-o anumita ordine, incepand cu "header", apoi "content", "footer", si "include"-urile secundare in ordinea in care sunt descoperite procesand (recursiv) template-urile. Variabilele pe care le creati intr-un "include" secundar si pe care le declarati a fi globale pentru a putea fi utilizate in template-uri, vor fi disponibile dupa cum urmeaza:

- daca variabila este folosita in template-ul corespunzator "include"-ului, variabila va putea fi inlocuita corect imediat
- daca variabila este folosita intr-un template inclus ulterior (avand rang mai mic, mai exact fiind descoperit mai tarziu decat elementul curent), variabila va putea fi inlocuita corect in momentul in care este descoperita
- daca variabila este folosita intr-un template care deja a fost inclus (de rang superior, de exemplu in "header", "content" sau "footer", dar si alte "include"-uri descoperite mai devreme) asistentul phprel a sesizat folosirea respectivei variabile inca de la procesarea acelui template, dar a intarziat inlocuirea ei din lipsa unei valori si in speranta ca va gasi valoarea intr-un "include" secundar. Valoarea va fi inlocuita corect intr-un ciclu/ etapa ulterioara de inlocuire, cu exceptia situatiei in care variabila apare intr-o expresie a unui tag "if", caz in care asistentul nu a intarziat procesarea tag-ului ci a evaluat respectiva conditie considerand valoarea variabilei ca fiind "null". Intr-o asemenea situatie, in care folositi o variabila declarata intr-un "include" de rang mai mic pentru o conditie dintr-un "if" al unui "include" de rang mai mare, folositi variabila ca tag de variabila globala imbricat in if, de exemplu: in loc de <&if{$variabila_a_unui_include_secundar}> scrieti <&if{<&variabila_a_unui_include_secundar>}> eventual intre apostroafe.

3.3.7. Coduri de prescurtare javascript

Asa cum am vazut, modulul Assistant va ofera un sprijin substantial in construirea paginilor, accesul la date, cereri asincrone (A.J.A.X.) si scrierea de html si php in general. Pentru a intregi imaginea, asistentul va ofera ajutor si in cazul scrierii anumitor secvente javascript.

In primul rand, scrierea tag-urilor <script type="text/javascript" language="javascript"> si </script> poate fi obositoare, si a fost inlocuita in phprel cu tag-urile <&js> si <&endjs>. In al doilea rand,

foarte des in codul javascript al paginilor web se folosesc elemente ale documentului identificate prin id, de tipul document.getElementById('(id element javascript)'). Aceasta scriere a fost prescurtata la &doc.(id element javascript) unde (id element javascript) este id-ul propriu-zis al elementului, inclus eventual intre paranteze rotunde in cazul in care contine caracterul punct ".". De exemplu:

```
&doc.nume
```

sau

```
&doc.(form.contents)
```

unde "nume" este id-ul unui camp de formular: <input type="text" name="nume" id="nume" value="" /> iar "form.contents" este id-ul unui div: .

Id-ul elementului este identificat prin subsirul care urmeaza dupa o constructie "&doc" pana la intalnirea celui mai apropiat caracter de separare (unul dintre caracterele punct ".", punct si virgula ";", spatiu " ", paranteza rotunda ")" sau enter "\n").

De asemenea, asa cum vom vedea in sectiunea curenta, subsectiunea 3.4. Formulare, la validarea automata a formularelor, mesajele stabilite pentru situatiile standard pot fi suprascrise cu alte mesaje, date sub forma unor functii de control scrise in javascript, de forma: function validate___(camp) () { return 'mesaj de eroare'; }. Aceste functii pot fi scrise prescurtat astfel:

```
&frm.validate: mesaj de eroare;
```

Nu in ultimul rand, asistentul phprel poate aplica un cod "tipar" mai multor campuri html din pagina prin gruparea lor folosind functia inovativa de grupare phprel (pentru detalii, consultati sectiunea curenta, subsectiunea 3.4.9. Grupuri si validare) si folosirea denumirii grupului pe post de id intr-o constructie de forma &doc.(id grup). De exemplu:

```
<&frm.camp1{checkbox; group="bife"}>
<&frm.camp2{checkbox; group="bife"}>
<&frm.camp3{checkbox; group="bife"}>
<a href="javascript: void(0);" onclick="javascript: &doc.bife.checked = 1-&this.checked;">toate</a>.
```

Acest cod va bifa sau debifa toate cele trei checkbox-uri la click pe link. O data grupate campurile, trebuie stiute urmatoarele:

- codul prescurtat &doc.(id) functioneaza si pentru id-uri de grupuri si are un comportament aparte in aceasta situatie
- daca id-ul specificat este al unui grup, asistentul phprel va genera in locul tiparului de instructiune dat, in acest caz "&doc.bife.checked = 1-&this.checked;", cate o instructiune similara pentru fiecare membru al grupului, in acest caz:
  - camp1
  - camp2
  - camp3
- instructiunile rezultate vor fi scrise una dupa alta in locul tiparului
- "&this" este un identificator special folosit doar in conjuctie cu &doc.(id) atunci cand (id) este id-ul unui grup, si insemna "membrul curent al grupului", fiind inlocuit cu o constructie de forma document.getElementById('(id membru curent)'). Oriunde este nevoie de o singura inlocuire referitoare la un membru al grupului, se va folosi &this.
- tiparul instructiunii este identificat dupa cum urmeaza:

- se porneste de la &doc si se identifica extremitatea stanga a instructiunii, delimitata printr-un din caracterele: punct si virgula ";", enter "\n", ghilimele ", sau sirul de caractere "javascript:".
- se identifica apoi extremitatea dreapta a instuctiunii tipar, delimitata initial prin caracterul punct si virgula ";". Daca instructiunea contine pana la prima punct si virgula un enter "\n", se considera ca este o instructiune complexa si trebuie delimitata conventional cu doua punct si virgula ";;".

Un alt exemplu care utilizeaza o instructiune tipar complexa ar putea fi:

```
<br />
<div id="help.div" style="height: 100px;" class="text_verde_2">
</div>
<&frm.camp1{checkbox; group="bife"; additional="onmouseover="javascript: indica()"
onmouseout="javascript: ascunde()" onclick="javascript: indica()""}>
<&frm.camp2{checkbox; group="bife"; additional="onmouseover="javascript: indica()"
onmouseout="javascript: ascunde()" onclick="javascript: indica()""}>
<&frm.camp3{checkbox; group="bife"; additional="onmouseover="javascript: indica()"
onmouseout="javascript: ascunde()" onclick="javascript: indica()""}>
<&js>
function indica()
{
        var sir = '';
        var i = 0;
        if (&doc.bife.checked)
        {
                i++;
                sir += 'Optiunea '+i+' este activa<br />';
        }
        else
        {
                i++;
        }
        ;;
        &doc.(help.div).innerHTML = sir;
}
function ascunde()
{
        &doc.(help.div).innerHTML = '';
}
<&endjs>
```

Sesizati folosirea conventionala a delimitatorului ";;". Instructiunea tipar este acum:

```
if (&doc.bife.checked)
{
    i++;
    sir += 'Optiunea '+i+' este activa<br />';
```

```
}
else
{
    i++;
}
```

delimitata la stanga de enterul dinainte de "if" si la dreapta de ";;".

Pentru a vedea exact functionarea acestui exemplu demonstrativ, se poate adauga continutul la o pagina phprel de test. La activarea unei bife sau la trecerea cu mouse-ul pe deasupra uneia dintre optiuni, o nota referitoare la care optiuni sunt active va aparea intr-un div explicativ.

3.3.8. Formulare si liste

Tag-urile de formulare si cele de liste sunt analizate si inlocuite automat de asistentul phprel in cadrul unor algoritmi complecsi de analiza, predictie si corelare care transforma modulul de formulare, in special, dar si cel de liste in cele mai puternice unelte de dezvoltare web aflate la indemana dumneavoastra atunci cand folositi phprel. Pentru detalii si utilizare, consultati sectiunea curenta, subsectiunea 3.4. Formulare si 3.5. Liste.

In plus, formularele scrise clasic (cu tag-uri html, nu phprel) pot fi preluate si transformate automat de asistentul phprel in formulare phprel functionale pentru situatia in care ati primit un design gata facut care continea si campurile unor formulare nefunctionale, folosite doar ca model. Trebuie stiut insa ca aceasta functie este pusa la dispozitie pentru accelerarea fazei de "pornire" a proiectului si nu este 100% exacta. Este recomandata intotdeauna scrierea formularelor sub forma tag-urilor speciale phprel. Nu in ultimul rand, un formular de adaugare/ modificare date va trebui completat cu campul hidden special "goPost" (pentru detalii, vezi sectiunea curenta, subsectiunea 3.2.2. Reguli mysql (rule) – query-uri automate). Pentru a activa aceasta optiune, initial dezactivata pentru a nu consuma resurse, modificati in cfgs/advanced/core.settings.inc.php valoarea variabilei $Tfrm_live_forms in "true".

3.3.9. Auto-globalizarea variabilelor

Asa cum am vazut deja in sectiunea curenta, subsectiunea 3.3.6. Tag-uri de includere a elementelor, asistentul phprel include automat elementele secundare ale paginii intr-un "scope" local. In general, in php-urile atasate acestor elemente veti construi variabile ale caror valori vor trebui inlocuite apoi in template-urile corespunzatoare sau in template-ul unui alt element al aceleiasi pagini. Pentru ca asistentul sa inlocuiasca automat aceste valori, asa cum am vazut deja, variabilele respective trebuie declarate in "scope"-ul global.

Rezulta o obligatie incomoda de a declara ca globale (folosind, de exemplu, constructia "global") variabilele folosite la afisare in elementele secundare, situatie care necesita un pic mai mult timp si poate fi sursa unor erori de afisare (atunci cand uitati pur si simplu sa declarati globala o variabila). Pentru a veni in ajutorul dumneavoastra, asistentul phprel analizeaza automat sursele php si html corespunzatoare unui element secundar in momentul includerii, si coreleaza denumirile tag-urilor cu denumirile variabilelor existente in php, globalizand automat variabilele folosite la afisare.

Astfel, pentru a afisa intr-un element secundar "stiri" o variabila $a, nu trebuie decat sa:

- atribuim valoarea corespunzatoare variabilei in stiri.php: $a = 'continut';
- scriem in template-ul stiri.html tag-ul corespunzator: <&a>

Asistentul va corela automat tag-ul cu variabila si va declara la includerea fisierului stiri.php variabila $a ca fiind globala. Auto-globalizarea functioneaza doar pentru variabilele afisate in template-ul corespunzator, de exemplu din "stiri.php" in "stiri.html", in cazul inlocuirii variabilei intr-un alt element de pagina, fie el principal sau secundar, variabila va trebui declarata manual globala.

Auto-globalizarea se face pentru toate variabilele folosite in tag-uri de variabile globale, bucle si tag-uri conditionale.

3.3.10. Ordinea de inlocuire

Algoritmul de inlocuire este echilibrat perfect intre performanta si flexibilitate, reusind sa inlocuiasca intr-un timp scurt aproximativ orice combinatie de tag-uri (cu mici exceptii in cazul carora se recomanda o alta strategie de imbricare a tag-urilor sau o balansare mai buna intre calculele efectuate in php si cele efectuate prin tag-uri speciale in html, de exemplu realizarea mai multor operatii in php si inlocuirea rezultatului sub forma unei simple variabile globale).

Inlocuirea are loc in sapte etape:

- inlocuirea dinamica a tag-urilor de variabile globale, bucle, conditionale si de incluziune in ordine optima. Ordinea de inlocuire este determinata in functie de situatie pentru a efectua cat mai putine calcule, cat mai putine incluziuni si cat mai putine inlocuiri de bucle, formulare si liste. Inlocuirea are loc in cicluri, la fiecare nou ciclu rezolvandu-se un nou nivel de adancime, ciclurile fiind intrerupte cand nu se mai pot efectua inlocuiri sau cand nu mai exista inlocuiri de efectuat. Se asigura astfel inlocuirea cu succes a oricator tag-uri si incluziuni imbricate.
- inlocuirea tag-urilor de traducere (daca este cazul, si are loc independent de asistent, in modulul multilanguage)
- inlocuirea tag-urilor de liste si inlocuirea dinamica a tag-urilor din detalierea listelor (similara primei etape dar aplicata elementelor de forma "(denumire lista).details")
- inlocuirea statica a tag-urilor de bucle, variabile globale si conditionale ramase nerezolvate in etapa I.
- inlocuirea codurilor scurte javascript si formularelor
- inlocuirea tuturor tag-urilor ramase, fie cu o valoare disponibila fie cu sirul vid, intr-o etapa finala de "curatare" (cu exceptia tag-urilor de forma <&sec{...}>, <&postload.(denumire)>, <&cache.(denumire)>)
- inlocuirea tag-urilor de adaugare manuala a codului de securitate si adaugarea automata a codului de securitate pentru linkuri cu risc ridicat (are loc independent de asistent, in modulul de validare a linkurilor)

Asa cum am vazut, o variabila apartinand unei bucle are prioritate de inlocuire in fata unei variabile globale. Ciclul de inlocuire dinamica va permite insa folosirea variabilelor globale in denumirea unei bucle pentru anumite situatii care sunt solutionate prin aceasta metoda. Astfel, tag-ul "prim" sau radacina nu mai exista, oricare tag poate contine oricare alt tag principal (recunoscut de motorul de template-uri), chiar oricare tag secundar (recunoscut de asistent) cu o singura exceptie: denumirea unei bucle nu poate contine un tag conditional.

Mai mult, elementele sunt procesate dupa cum urmeaza:

- index
- header
- content
- footer
- elemente secundare, in ordinea in care sunt descoperite procesand elementele in ordinea prezentata in aceasta lista

Un element de pagina care nu trebuie inclus nu va fi inclus de la bun inceput, de exemplu continutul urmator:

```
<&if{0}>
        <&include "stiri" />
<&endif>
```

va avea ca efect excluderea din start a tag-ului de incluziune din template. Ca urmare, fisierul "stiri.php" nu va fi inclus vreodata, codul din interiorul sau nefiind executat.

De asemenea, inlocuirea automata are loc intr-o etapa ulterioara includerii php-urilor corespunzatoare header-ului, continutului si footer-ului, asistentul va "percepe" template-urile asa cum sunt ele in urma eventualelor modificari realizate de dumneavoastra manual in respectivele php-uri, iar dumneavoastra nu veti putea "vedea" in acele php-uri modificarile realizate de asistent in template-uri sau accesa valori ale unor variabile din fisiere incluse de asistent din simplul motiv ca respectivele modificari/ incluziuni nu au avut inca loc.

3.4. Formulare

Formularele reprezinta unul din punctele cheie in acordarea sprijinului phprel pentru o dezvoltare mai rapida. Folosind phprel, veti crea mai rapid formulare complexe care folosesc standard tehnologii de tip A.J.A.X., validari la client (fara reincarcarea paginii), procesarea securizata a datelor trimise si sunt in acelasi timp mai ergonomice. In plus, formularele vor fi functionale fara nici o linie php scrisa de dumneavoastra, sau cu un minimum de linii adaugate si un numar mic de indicatii oferite asistentului. Totul de la completarea datelor in campuri, campuri cu valori persistente, validari, procesare si redirectionare va fi asigurat de phprel, in plus fata de crearea propriu-zisa a tag-urilor html necesare formularului.

3.4.1. Tag-ul de formular si atribute

Tag-ul phprel de creare a formularelor este de forma <&frm.(denumire camp)> sau <&frm.(denumire camp){(indicatii)}>, unde (denumire camp) este denumirea campului html care va aparea in atributul "name" iar (indicatii) sunt specificatiile date asistentului pentru construirea tag-ului html. Fiecare tag de forma <&frm.(sir de caractere)> va fi inlocuit cu unul din tag-urile html:

- <form>
- <input>
- <select>

- <textarea>

sau in anumite situatii cu un simplu sir de caractere reprezentand valoarea campului,

sau cu un sir vid in cazul in care campul nu a fost complet definit si asistentul phprel nu a putut intui atributele esentiale ale campului.

In primul rand, trebuie stiut ca un camp de formular construit de phprel va primi atributul "name" specificat si atributul "id" format din denumirea campului si un numar de ordine (corespunzator numarului de aparitii ale respectivei denumirii descoperite pana in momentul respectiv de phprel in pagina). De exemplu, un camp <&frm.nume> unic in pagina va avea atributele name="nume" id="nume1".

Un formular se incepe cu un tag <&frm.form> in cazul unui formular principal sau <&frm.form.(nume formular secundar)> in cazul unuia secundar, unde (nume formular secundar) este valoarea atributului "name" a tag-ului <form> rezultat, si se incheie cu tag-ul <&endform> care va fi inlocuit cu </form>. Tag-urile de formular vor avea atributele "name" si "id" identice si egale cu denumirea data prin (nume formular secundar) in cazul formularelor secundare sau cu "form" in cazul formularului principal. Numele si id-ul formularului principal se va putea schimba printr-o indicatie, daca este cazul.

Fiecare tag reprezentand un camp va trebui evident sa aiba un tip (atributul html "type" sau tipul concret al tag-ului: input, select, textarea). Tipul poate fi indicat de dumneavoastra sau poate fi intuit de asistentul phprel. In cazul in care nu specificati tipul unui camp, asistentul va determina sau presupune automat tipul campului daca formularul este corelat cu un tabel care are descriere activa in web/rules/tables/. Un formular poate fi corelat cu un tabel printr-o indicatie data de dumneavoastra sau, daca este tabelul principal si exista regula mysql pe pagina, formularul va fi corelat automat cu tabelul regulii mysql. Daca formularul a fost corelat automat sau de catre dumneavoastra cu un tabel, atunci pentru fiecare camp mysql al tabelului care se afla printre campurile formularului tipul va fi determinat dupa cum urmeaza:

- daca ati specificat un tip al campului folosind $type din descrierea tabelului, respectivul tip va fi folosit pentru a crea campul
- daca respectivul camp relationeaza tabelul de un alt tabel, tipul campului va fi "select" iar optiunile vor fi construite pe baza inregistrarilor tabelului relationat
- daca respectivul camp are o denumire care se potriveste tiparelor frecvente definite in cfgs/advanced/assistant.inc.php in array-ul asociativ $Tasi_guesstype, tipul determinat va fi folosit pentru a crea campul
- in orice alta situatie, campul va fi creat ca "text".

In plus, campurile altor tabele relationate care relationeaza la randul lor un tabel secundar de un altul si care ajuta la restrangerea optiunilor unui select-box vor fi inlocuite de asemenea cu select-box-uri care restrang automat prin cereri asincrone (A.J.A.X.) optiunile select-box-urilor "child". Pentru mai multe detalii, consultati sectiunea curenta, subsectiunea 3.4.6. Campuri select de tip parent-child.

Campurile al caror tip nu a fost determinat corect sau care apartin unor formulare care nu sunt sau nu pot fi corelate cu un tabel, vor fi completate cu indicatii privind tipul lor.

Important: Corelarea formularului cu un tabel foloseste doar la determinarea tipurilor campurilor, procesarea datelor realizandu-se pe baza regulii mysql atasate paginii in care datele sunt trimise de formular.

Nu in ultimul rand, un submit poate fi creat folosind tag-ul <&frm.submit> in cazul formularului principal sau <&frm.submit.(denumire)> in cazul unui formular secundar. Acest tag va fi folosit pentru

crearea oricarui submit, fie ca este un simplu buton fie ca este o imagine. Este important de stiut faptul ca un buton de submit va avea atributul "id" de forma "submitButton(numar de ordine al submit-ului)".

Oricarui camp, determinat automat sau specificat manual, i se pot adauga o serie de atribute care vor defini comportamentul si aspectul lor final, majoritatea fiind atribute de legatura cu atribute corespondente html (de exemplu "class", "style", "value"), altele fiind atribute speciale phprel care va ajuta la construirea cererilor asincrone ("async"), la gruparea campurilor ("group"), la stabilirea unei valori initiale ("defaultvalue"), si altele. Atributele posibile sunt urmatoarele:

- pentru tag-uri de <form>, create prin <&frm.form> sau <&frm.form.(denumire form secundar)>:
  - name – specifica numele (atributele "name" si "id") al formularului, de regula al formularului principal, deoarece unui formular secundar i se specifica numele in tag-ul principal
  - action – specifica atributul html "action" al formularului, respectiv url-ul caruia i se trimite formularul. Implicit vid.
  - method – specifica metoda de trimitere a datelor: fie una dintre metodele clasice "get" sau "post" fie "async" pentru trimiterea datelor printr-o cerere asincrona javascript + xml (A.J.A.X.) prin suportul oferit transparent de phprel. Implicit "post".
  - class – specifica atributul html "class" al formularului, respectiv clasa css a formularului
  - table – specifica tabelul cu care este corelat formularul. Tabelul trebuie sa aiba descriere in web/rules/tables/ si va fi dat in format phprel.
  - fromrule – fara valoare, specifica faptul ca formularul (de regula secundar) este corelat cu tabelul regulii mysql. Corelarea este o prescurtare a scrierii atributului "table" cu valoarea corespunzatoare denumirii tabelului din regula mysql si nu inseamna ca si datele vor fi incarcate pe baza regulii in formular. Doar un formular principal poate incarca date pe baza regulii mysql.
  - globaltemplate – specifica template-ul de tag folosit de phprel la construirea tag-urilor apartinand formularului care nu au specificate un "template" propriu. Pentru mai multe detalii, consultati sectiunea 3.4.3. Template-uri de tag.
  - additional – orice alte atribute html care trebuie atasate tag-ului rezultat. Valoarea va fi preluata ca sir de caractere si adaugata fara modificari in interiorul tag-ului rezultat. Orice caracter punct-si-virgula ";" va fi inlocuit cu [sc], orice caracter mai-mare ">" cu [gt] si orice simbol mai-mare-sau-egal ">=" cu [gtoe]. Alte caractere, inclusiv ghilimele, pot fi folosite fara restrictii.
- pentru tag-uri <input>, <select>, <textarea> create prin <&frm.(denumire)>:
  - type – specifica atributul html "type" al campului, respectiv tipul input-ului. Valorile pot fi:
    - text, pass (prescurtare pentru "password"), password, checkbox, file, hidden, radio, reset, button, submit (nerecomandat, necesar doar in situatia in care submit-ul creat folosind tag-ul special de submit nu este potrivit situatiei ceea ce este foarte putin probabil), image (nerecomandat, similar motivului enuntat pentru "submit")
    - select – pentru tag de tip <select>
    - textarea – pentru tag de tip <textarea>

    - editor – campul va fi construit sub forma unui <textarea> si va fi incarcat ca editor wyswyg (tinyMCE sau alt editor configurat de dumneavoastra)

  De observat faptul ca tipul poate fi specificat fara denumirea atributului "type", prin simpla valoare, de exemplu "text", daca este specificat ca prim atribut in sirul de indicatii.
- calendar – specifica faptul ca respectivul camp va fi folosit pentru a selecta o data calendaristica folosind o aplicatie third-party de tip calendar dhtml
- value – specifica atributul html "value", respectiv valoarea campului respectiv. In cazul specificarii explicite a unei valori, functiile de incarcare automata din POST sau MySQL nu vor fi folosite, valoarea data explicit avand prioritate. Folosita de obicei in conjunctie cu campuri "hidden".
- class – specifica atributul html "class", clasa css a campului.
- style – specifica atributul html "style", atributele css ale campului. Caracterele punct-si-virgula ";" vor fi inlocuite cu [sc].
- defaultvalue – stabileste valoarea initiala a campului, la prima afisare a formularului atunci cand formularul nu modifica informatii deja existente sau nu contine date, valoare care va fi suprascrisa de functiile de incarcare automata din POST sau MySQL. Foloseste variabila $_POST pentru stabilirea valorii.
- options – specifica optiunile unui "select-box". Tipul "select" este implicit subinteles in cazul folosirii atributului "options" si nu mai trebuie precizat. Optiunile se pot specifica in mai multe moduri:
    - in clar, separate prin virgula: optiune1, optiune2, optiune3, ...
      atributul "value" al fiecarei optiuni va fi un numar intreg de la 0 la N-1, corespunzator numarului optiunii in ordinea data in sir.
    - in clar, separate prin virgula precizand si atributul "value" al fiecarei optiuni: value1::optiune1, value2::optiune2, value3::optiune3, ...
    - prin referinta la o variabila php care contine optiunile intr-un format valid:
      options="$optiuni"
      unde variabila va contine optiuni ca sir de caractere (optiuni separate prin virgula similar primelor doua metode) sau ca array de optiuni in formatul asteptat de phprel, obtinut printr-una din functiile retOpt(), arrayToOpt(), getData() cu functie de optiuni, sau construit manual in formatul necesar (nerecomandat).
    - printr-un apel subinteles la getData, catre una din functiile options, searchoptions sau filteroptions aplicate unui tabel care are descriere activa in web/rules/tables/ si are specificat un $namefield (pe baza caruia se vor construi optiunile): (tabel).searchoptions::(conditie where)
- autopost – indica faptul ca la orice modificare a campului (onchange), formularul va fi trimis automat.
- rows – pentru textarea, specifica numarul de linii al campului.
- size – pentru select-box, specifica atributul html "size", respectiv dimensiunea campului in numar de linii daca acesta nu este un "drop-down" standard
- multiple – pentru select-box, specifica atributul html "multiple", care permite selectarea mai multor optiuni simultan.

- template – specifica template-ul de tag folosit de phprel la construirea tag-ului. Pentru mai multe detalii, consultati sectiunea curenta, subsectiunea 3.4.3. Template-uri de tag.
- async – specifica o cerere asincrona atasata campului. Valoarea atributului va fi de forma:
  (handler).(target)
  sau
  (handler).(caller).(target)[.(pagina)[.(procesor)]]
  cererea fiind atasata evenimentului "onchange", sau
  (eveniment): (handler).(target)
  sau
  (eveniment): (handler).(caller).(target)[.(pagina)[.(procesor)]]
  cererea fiind atasata evenimentului specificat prin (eveniment).
  Pentru mai multe detalii, consultati sectiunea curenta, subsectiunea 3.4.10. Formulare si cereri asincrone javascript + xml (A.J.A.X.)
- group – specifica grupul sau grupurile din care face parte tag-ul. Gruparea campurilor va fi folositoare la validare, la trimiterea unei cereri asincrone si la scrierea prescurtata a codului javascript. Numele grupului sau grupurilor nu trebuie sa coincida cu numele unui camp sau cu id-ul dat unui element html (folosind atributul "id"). Daca respectivul camp apartine mai multor grupuri, acestea vor fi precizate separate prin virgula.
- * - atributul asterix specifica validarea campului. Fara valoare, el indica validarea standard pentru valori nevide. Cu valoare poate indica o validare prin pattern de sir (regexp), prin cerere asincrona (A.J.A.X.) sau prin comparare cu un camp de validare. Pentru detalii, consultati sectiunea curenta, subsectiunea 3.4.9. Grupuri si validare.
- label – specifica denumirea campului asa cum va aparea intr-un mesaj de eroare la validare. Folosit impreuna cu validarea campului, in cazul in care denumirea reala a campului nu este stabilita corect de asistent, ca parte a validarii automate.
- additional – orice alte atribute html care trebuie adaugate tag-ului rezultat. Valoarea atributului va fi preluata si adaugata fara modificari in continutul tag-ului. Orice caracter punct-si-virgula ";" va fi inlocuit cu [sc], orice caracter mai-mare ">" cu [gt] si orice simbol mai-mare-sau-egal ">=" cu [gtoe]. Alte caractere, inclusiv ghilimele, pot fi folosite fara restrictii.

- pentru tag-uri <input type="submit"> si <input type="image">, create prin <&frm.submit> sau <&frm.submit.(denumire)>:
  - value – specifica valoarea atributului html "value", textul care apare pe butonul de "submit".
  - image – specifica valoarea atributului html "src", calea catre imaginea afisata ca "submit". Tipul campului va fi implicit "image".
  - class – specifica atributul html "class", clasa css a campului.
  - style – specifica atributul html "style", atributele css ale campului. Caracterele punct-si-virgula ";" vor fi inlocuite cu [sc].
  - template - specifica template-ul de tag folosit de phprel la construirea tag-ului. Pentru mai multe detalii, consultati sectiunea curenta, subsectiunea 3.4.3. Template-uri de tag.
  - additional - orice alte atribute html care trebuie adaugate tag-ului rezultat. Valoarea atributului va fi preluata si adaugata fara modificari in continutul tag-ului. Orice

caracter punct-si-virgula ";" va fi inlocuit cu [sc], orice caracter mai-mare ">" cu [gt] si orice simbol mai-mare-sau-egal ">=" cu [gtoe]. Alte caractere, inclusiv ghilimele, pot fi folosite fara restrictii.

Atributele se specifica sub forma unui sir de indicatii separate prin punct-si-virgula, iar valorile prin caracterul egal "=" urmate de valoare inclusa intre ghilimele(valoarea poate sa nu fie inclusa intre ghilimele si va fi interpretata corect, dar aceasta metoda nu este recomandata). Sirul de indicatii va urma imediat denumirii campului, inclus intre acolade. De exemplu:

```
<&frm.(denumire){(atribut1)="(valoare1)"; (atribut2)="(valoare2)"; (atribut3)="(valoare3)";
...}>
sau
<&frm.(denumire){(atribut1)="(valoare1)"}>
sau
<&frm.(denumire){(atribut1)="(valoare1)"; (atribut2)}>
```

Cateva exemple concrete ar fi:

```
<&frm.form{action="adauga.produs"}>
<&frm.form.login{globaltemplate="long"}>
<&frm.status{options="1::Activ, 0::Inactiv"}> sau <&frm.status{options="Inactiv, Activ"}>
<&frm.arediscount{checkbox}>
```

Alte exemple au fost deja prezentate intuitiv in sectiunile anterioare.

Denumirile campurilor de formular pot contine inclusiv caracterele paranteza-dreapta "[", "]" pentru construirea campurilor de tip "array".

3.4.2. Incarcare automata din POST si MySQL

Formularele create cu phprel au valori "persistente", in urma "post"-ului campurile pastrandu-si valorile. Acest lucru este util de exemplu in cazul formularelor de filtrare a unei liste, sau daca o validare sau supra-validare care are loc (din anumite motive) in php determina faptul ca formularul nu este inca valid pentru a fi introdus in baza de date. Fiecare camp preia la creare valoarea lui corespunzatoare din variabila superglobala php $_POST, pe baza denumirii campului (atributul "name" al campului, mai exact denumirea data in tag: <&frm.(denumire){...}>). Avand in vedere ordinea de succesiune a proceselor in nucleul phprel, se pot in plus stabili valori campurilor de formular prin atribuirea de chei si valori variabilei $_POST in php-ul atasat paginii.

De asemenea, un formular principal aflat pe o pagina care are atasata o regula mysql si care se ocupa de modificarea unei linii dintr-un tabel va prelua automat valorile campurilor respective in campurile de formular, fara cod suplimentar. Astfel, formularele de editare construite cu ajutorul regulilor mysql (fie ele definite manual sau, in majoritatea cazurilor, automat de asistentul phprel) vor functiona de la sine. Valorile sunt preluate din variabila phprel $_MySQL in care regula mysql incarca datele din tabel corespunzatoare unei linii indicate prin parametrul GET "edit" sau "view".

Important: Incarcarea valorilor din POST are prioritate in fata incarcarii din MySQL.

3.4.3. Template de tag

Campurile de formular vor avea un anumit aspect care se pastreaza pentru toate campurile aplicatiei, indiferent de pagina, si vor avea de regula dimensiuni standard in cadrul aceluiasi formular. Pentru a controla mai usor aspectul, prin "style"-ul si clasa css atribuite campurilor, phprel va propune crearea unor tipare care se potrivesc aplicatiei dumneavoastra si folosirea lor in conjunctie cu campurile de formular. Astfel, phprel va pune la dispozitie in web/settings/form.settings.inc.php o metoda simpla de definire a unor "template"-uri de tag, care se vor aplica implicit tag-urilor sau vor fi aplicate la cerere, formate astfel:

- o clasa css
- un style
- un "additional" reprezentand cod html care se adauga nemodificat la orice tag care apartine template-ului respectiv.

Un "template" de tag are o denumire care il identifica unic. In fisierul de setari veti gasi deja predefinite cateva template-uri:

- vshort
- short
- medium
- long
- vlong

Toate vor adauga tag-urilor clasa css implicita "input", fara nici o specificatie "aditionala", iar din style vor seta lungimi mai mici sau mai mari campurilor. Orice camp al aplicatiei care nu are setat un atribut "class", "style" sau "additional" propriu, va imprumuta respectivul camp din template-ul atribuit campului. Unui camp i se atribuie un template astfel:

- specificand direct template-ul prin atributul "template" al tag-ului phprel
- specificand indirect template-ul prin atributul "globaltemplate" al tag-ului phprel de formular din care face parte campul
- ca ultima varianta, campul este considerat ca apartinand template-ului standard setat prin ultimul apel la functia setTemplate, de regula apelul din web/settings/form.settings.inc.php de dupa definirea template-urilor.

Un template se creaza cu un apel la functia newTemplate:

- newTemplate ('(denumire template)', array('class' => '(clasa css)', 'style' => '(style css)', additional => '(cod aditional)'));

Se recomanda ca toate template-urile sa fie create in fisierul web/settings/form.settings.inc.php de unde vor putea fi folosite pe orice pagina a aplicatiei. Pentru a stabili template-ul standard pe baza caruia sunt create toate campurile pentru care nu se specifica un template, modificati apelul la setTemplate din fisierul de configurare.

Un exemplu ar putea fi, in situatia in care dispuneti inca de sursele create in sectiunile 1. phprel la prima vedere si 2. Elemente de baza, modificarea dimensiunii campurilor pentru template-ul "medium", din atributul "style" al respectivului tipar si reincarcarea paginii de "Adauga stire". Alternativ, se poate adauga la acelasi template "medium" atributul "additional" urmator:

```
'additional' => 'onfocus="javascript: this.style.backgroundColor=\'#F4FBFF\';"
onblur="javascript: this.style.backgroundColor=\'#FFFFFF\';"',
```

La un refresh al paginii de "Adauga stire" si click in campurile formularului veti observa rezultatul.

Important: template-ul "submit" pus la dispozitie standard in fisierul web/settings/form.settings.inc.php este inlus in setari ca punct de plecare pentru un template pe care il veti aplica butoanelor de submit. Acest template nu este aplicat automat de asistentul phprel la crearea tag-urilor html de "submit" deci pentru a schimba aspectul acestor tag-uri, nu este suficient sa modificati acest template, va trebui sa folositi atributele phprel "template", "class" sau "style" pentru a indica asistentului un alt aspect al tag-ului.

3.4.4. Formular principal si formulare secundare

Folosind phprel se pot crea oricate formulare pe o pagina. Acel formular care foloseste implicit regula mysql atasata paginii, pentru a insera sau modifica o linie dintr-un tabel, este formularul principal, restul formularelor fiind secundare. Formularul principal se defineste folosind tag-ul <&frm.form> si trebuie sa fie unic pe pagina. Acest formular va fi corelat automat cu tabelul regulii mysql, daca exista, si va fi precompletat automat cu datele din tabel in situatia in care se foloseste formularul pentru a modifica o linie din tabel. Toate celelalte formulare vor fi definite prin tag-uri de forma <&frm.form.(denumire)> si vor trebui corelate explicit cu un tabel folosind de regula atributul "table", sau "fromrule". Campurile unui formular secundar nu vor fi precompletate niciodata cu datele selectate pe baza regulii, pentru ca nu pot corespunde regulii mysql.

In principiu, toate formularele "mici" de login, abonare newsletter, etc. vor fi formulare secundare. In special formulare din elemente secundare de pagina care se includ pe mai multe paginii vor fi definite ca secundare (pentru a nu se suprapune la un moment dat cu un formular principal definit in continutul unei pagini). Formulare principale vor fi formularele definite de regula in "content" (sau in elemente secundare incluse pe un numar restrans de pagini) care au ca scop actualizarea datelor dintr-un tabel prin punerea la dispozitie atat a unei functii de inserare cat si a unei functii de modificare a liniilor. Pentru a prescurta scrierea, se poate folosi un formular principal si atunci cand nu e neaparat necesar, de exemplu la scrierea unui formular de filtrare a unei liste. Cu toate ca formularul respectiv nu este folosit pentru actualizarea bazei de date (nici macar nu va contine un camp hidden "goPost" necesar in asemenea situatie), daca pe pagina respectiva nu se afla un alt formular care sa aiba nevoie sa fie principal, se va castiga timp definind formularul ca principal (pentru ca nu va fi nevoie sa specificati si o denumire).

3.4.5. Campuri cu aceeasi denumire

Atunci cand creati multiple formulare pe o singura pagina, in anumite situatii campuri din diferite formulare vor avea denumiri identice. Asistentul phprel va construi tag-urile cu acelasi atribut html "name" (pentru ca o data trimise, datele formularului sa corespunda denumirilor necesare), dar cu atribute "id" diferite (pronind de la atributul "name" si adaugand un numar de ordine) pentru a putea fi identificate unic in pagina, in scopul realizarii validarilor, dar si a altor operatii automate phprel sau realizate de dumneavoastra. Campurile vor fi numerotate in ordinea in care sunt descoperite, care corespunde, in interiorul aceluiasi element de pagina, ordinii (de sus in jos) in care apar tag-urile in sursa iar intre diferitele elemente de pagina ordinii in care phprel include elementele (header, content, footer, include-uri secundare in ordinea descoperirii – definita evident dupa aceeasi schema recursiva: header, content, footer,

include-uri). Din punct de vedere functional, fiecare formular se va comporta ca si cum campurile lui sunt unic denumite pe pagina, toate functiile puse la dispozitie de formularele phprel fiind operationale. Se pastreaza cu ajutorul unor campuri "hidden" generate automat de asistentul phprel chiar si functia de "valori persistente prin POST" (asistentul va stii chiar si dupa trimiterea datelor din care formular a fost trimis un anumit camp).

Este important sa retineti ca id-urile difera de denumiri si pastreaza standardul "(denumire)(numar de ordine)" atunci cand scrieti cod javascript (de exemplu: cereri asincrone, sau schimbari de proprietati css, sau schimbari de valori etc.) care afecteaza sau folosesc un camp generat de asistentul phprel. Este important sa distingeti doua situatii: cand va referiti la un camp din "interiorul formularului" (insemnand din interiorul atributelor "async" sau "*" al oricarui camp al respectivului formular) se poate folosi doar denumirea – care va fi automat convertita de asistent in id-ul corect, cand va referiti la un camp din "exteriorul formularului" (orice cod javascript, cerere asincrona etc. realizata in afara tag-urilor de formular apartinand respectivului formular) va trebui sa verificati id-ul de generare al unui camp si sa-l folositi intocmai. De exemplu, pentru un tag <&frm.nume> veti folosi id-ul "nume1" sau daca sunt doua tag-uri <&frm.nume> veti folosi id-ul "nume1" sau "nume2".

3.4.6. Campuri select de tip parent-child

La construirea formularelor, asistentul va inlocui campurile care relationeaza tabelul formularului de alte tabele cu select-box-uri. Aceasta metoda este des intalnita si probabil cea mai preferata pentru a specifica legatura cu un alt tabel. De exemplu, un camp "id_categorie" va fi inlocuit cu un select-box in care vor aparea denumirile categoriilor (avand valori id-urile corespunzatoare).

Campurile pentru care aceasta inlocuire se realizeaza automat sunt campurile care nu au specificate un tip (prin atributul phprel "type" sau implicit, la folosirea atributului "options", "calendar" sau "editor"), apartin unui formular corelat cu un tabel si sunt specificate in array-ul asociativ $relate din descrierea tabelului corelat. In plus, tabelul destinatie (care este relationat cu tabelul formularului prin respectivul camp) va trebui sa aiba specificat un $namefield in descrierea de tabel (pentru a putea construi select-box-ul).

In anumite situatii, selectarea unei valori dintr-o astfel de lista poate fi dificila din cauza numarului mare de valori, si aria ar putea fi restransa prin folosirea inca unui camp. De exemplu, tabelul formularului este relationat cu un tabel (Y) care la randul lui este relationat cu un tabel (X). S-ar putea selecta mai intai o valoare pentru tabelul (X) care sa restranga valorile posibile pentru (Y), deci selectia pentru tabelul formularului. Sa concretizam exemplul, tabelul formularului, "produse", este relationat de un tabel "subcategorii" prin "id_subcategorie", care la randul sau este relationat de un tabel "categorii" prin "id_categorie". Am putea adauga formularului, inainte de campul "id_subcategorie", campul "id_categorie" si selecta mai intai o categorie care ar restrange subcategoriile posibile. Aceasta metoda este de asemenea des folosita iar asistentul phprel o va recunoaste si va inlocui automat campurile de acest tip ("parent" – "child") indiferent de numarul nivelelor pe care se intinde legatura. Pentru restrangerea valorilor corespunzatoare tabelului (X) se poate deci adauga in continuare un tabel (Z), lantul devenind (Z)-(X)-(Y). De asemenea se pot adauga mai multe astfel de "fire de selectie" intr-un singur formular. Premisa este ca descrierile de tabel specifica legaturile corect, iar campurile din formular sunt identice ca denumire cu cele din tabele, fiind folosite mereu campurile de legatura (specificate in $relate pentru fiecare tabel).

Restrangerea optiunilor unui camp "child" la selectarea unei valori pentru un camp "parent" va fi realizata printr-o cerere asincrona (A.J.A.X.) sau, daca browser-ul utilizatorului nu permite o astfel de cerere, prin "POST". Selectarea metodei se va face transparent.

3.4.7. Construirea optiunilor unui select

Unui select-box construit folosind phprel i se pot atasa optiunile dorite prin mai multe metode specificate deja in sectiunea curenta, subsectiunea 3.4.1. Tag-ul de formular si atribute. Construirea standard a optiunilor se realizeaza intr-un format special de array, de obicei folosind functiile addOpt(), addEmptyOpt(), retOpt() sau arrayToOpt() sau o functie specifica din cadrul getData().

Prima metoda foloseste functiile addOpt, retOpt:

- addOpt ($valoare, $text) - adauga o noua optiune intr-un "buffer" temporar, avand atributul "value" dat prin $valoare si textul propriu-zis care apare in select-box dat prin $text.
- (array de optiuni) retOpt() - intoarce toate optiunile din "buffer"-ul temporar, sub forma unui array care poate fi folosit direct intr-un select-box creat folosind phprel. Un exemplu de folosire ar fi, dupa construirea optiunilor temporare cu addOpt():

  in php:
  $opts = retOpt();
  in html:
  <&frm.selectbox{options="$opts"}>

  In plus, apelul la retOpt() va sterge "buffer"-ul temporar, cu exceptia cazului in care acest lucru este interzis prin trimiterea unui parametru "false": retOpt(false).

O a doua metoda foloseste functia arrayToOpt:

- (array de optiuni) arrayToOpt ($array_simplu [, $exclude]) - converteste un array simplu de forma cheie => valoare (de regula cu chei numerice consecutive, dar nu numai) intr-un array de optiuni. Se pot specifica suplimentar, printr-un array asociativ cheile care se exclud.

A treia metoda foloseste functia getData, subfunctiile options, searchoptions sau filteroptions aplicate unui tabel care are descriere activa in web/rules/tables si descrierea contine $namefield. Se aplica una din cele trei subfunctii unui tabel, rezultand un array de optiuni:

- (arrray de optiuni) getData ('(denumire tabel in format phprel).(functie: fie options, fie searchoptions, fie filteroptions)', (parametru functie: fie nici un parametru, fie o conditie where, fie valorile campurilor de filtrare)) - intoarce un array de optiuni construite selectand cheia primara pentru atibutul "value" si expresia data la $namefield pentru textul propriu-zis al optiunii.

Indiferent de metoda, se poate folosi addEmptyOpt pentru a adauga o prima optiune "vida" (pentru cazul initial, cand nu este inca nimic selectat):

- (array de optiuni) addEmptyOpt ((array de optiuni))

Optiunea adaugata va contine sirul vid ca atribut "value" si textul configurat in cfgs/advanced/core.settings.inc.php prin variabila $Tfrm_defaultEmptyOption, standard trecuta vida.

Ca metoda suplimentara, pentru select-box-urile construite cu ajutorul asistentului phprel, optiunile pot fi date direct printr-un sir de caractere sau printr-o referinta la o variabila continand un sir de caractere reprezentand optiuni. Un sir de caractere reprezentand optiuni are unul din urmatoarele formate:

- textele optiunilor separate prin virgula: optiune1, optiune2, optiune3, ... Optiunile specificate vor aparea propriu-zis in select-box, fiecaruia i se va asocia ca atribut "value" numarul de ordine al optiunii in sirul de caractere (incepand cu 0 pentru prima optiune, 1 pentru a doua, etc.)
- optiuni cu valoare si text separate prin virgula: valoare1::optiune1, valoare2::optiune2, ... Optiunile date vor avea ca atribut "value" exact valoarea data. Separarea valorii de textul propriu-zis se face prin operatorul "::".

Specificarea optiunilor prin sir de caractere se poate face direct in atributul phprel "options":

```
<&frm.selectbox{options="optiune1, optiune2, optiune3"}>
```

sau prin referinta la o variabila care contine sirul de caractere:

in php:

```
$opts = 'optiune1, optiune2, optiune3';
```

in html:

```
<&frm.selectbox{options="$opts"}>
```

Evident, se poate specifica si o prima optiune vida folosind aceasta metoda cu sir de caractere:

```
<&frm.selectbox{options=", optiune1, optiune2, optiune3"}>
```

3.4.8. Formular principal cu tabele auxiliare

Majoritatea formularelor adauga sau actualizeaza o singura linie dintr-un singur tabel la un moment dat. Exista insa situatii cand un formular adauga sau modifica o linie intr-un tabel principal si alte linii intr-unul sau mai multe tabele secundare. De exemplu, un formular de adaugat produse, cu categorie, subcategorie, furnizor, denumire, pret, si o serie de caracteristici dintr-un nomenclator de caracteristici (marcate printr-o bifa – un checkbox – daca respectivul produs are respectiva caracteristica sau nu). Din punct de vedere al bazei de date, ar fi necesar un tabel "produse" (tabelul principal), un tabel de caracteristici (nomenclatorul) si un tabel de "caracteristici produse" (cu legaturi la tabelul de produse si caracteristici) reprezentand caracteristicile asociate unui anumit produs. Trimiterea datelor ar trebui sa realizeze o inserare a unei linii in tabelul principal, continand datele principale referitoare la produs, si inserarea cate unei linii in tabelul de "caracteristici produse" pentru fiecare caracteristica bifata. Actualizarea datelor ar trebui sa cuprinda, la afisarea formularului: selectarea datelor din tabelul principal si selectarea liniilor din tabelul de "caracteristici produse", la trimiterea datelor: actualizarea datelor din tabelul principal, stergerea liniilor care au fost debifate, inserarea liniilor nou bifate.

Asistentul phprel sesizeaza automat astfel de situatii si coreleaza formularul cu toate tabelele necesare. Toate query-urile vor fi scrise si rulate de asistent, si datele incarcate in formular si salvate in baza de date automat. Premisa de recunoastere si functionare a formularului cu un tabel auxiliar este urmatoarea:

- pe langa campurile apartinand tabelului principal, exista campuri de forma:
  `<&frm.(denumire)[(id)]>`
- tabelul principal este relationat de un al doilea tabel ca "parent", respectiv are un $relate "as.child". Tabelul secundar este deci subordonat tabelului principal si contine la randul lui o legatura complementara "as.parent" catre tabelul principal.
- tabelul secundar contine suplimentar o a doua legatura "as.parent" catre un alt tabel. Tabelul secundar este, de fapt, un tabel de legatura N-la-M intre doua tabele care poate sa contina si

alte campuri ce descriu sau detaliaza intr-un anume fel legatura dintre cele doua tabele. De exemplu, in exemplul cu caracteristicile produselor, fara alte campuri, tabelul de "caracteristici produse" ar asocia unele caracteristici unui anume produs. Cu campuri suplimentare, o caracteristica de tipul "Inaltime" sau "Garantie" ar putea primi si o anume valoare specifica produsului (intr-un camp denumit, de exemplu "valoare", din tabelul de "caracteristici produse").

- campurile cu denumiri complexe de tip array "(denumire)[(id)]" apartin tabelului secundar – prin (denumire) – iar (id) va fi id-ul concret din cel de-al doilea tabel "parent", stabilind deci corelatia intre tabelul principal si cel de-al doilea tabel "parent" prin intermediul tabelului secundar. Campul specificat prin (denumire) poate fi:
  - campul ce relationeaza tabelul secundar de al doilea "parent", camp de tip checkbox insemnand ca linia editata din tabelul principal este legata de linia cu (id) dat din al doilea tabel "parent"
  - un camp din tabelul secundar, de orice tip reprezentand valoarea campului pentru "linia de legatura" din tabelul secundar stabilita intre linia editata din tabelul principal si linia cu (id) dat din al doilea tabel "parent"
- un acelasi camp mysql poate avea un numar oricat de mare de campuri in formular, pe baza diferitelor id-uri date prin (id). Campurile vor fi separate, de exemplu:

```
<&frm.valoare[3]>
<&frm.valoare[11]>
<&frm.valoare[12]>
...
```

Un exemplu simplificat ar fi:

- un tabel "produse", cu campurile "id, denumire, pret"
- un tabel "caracteristici", cu campurile "id, denumire", si $namefield = 'denumire'
- un tabel "produse_caracteristici" cu campurile "id, id_produs, id_caracteristica", cu:

```
$relate['produse'] = 'as.parent.by.id_produs';
$relate['caracteristici'] = 'as.parent.by.id_caracteristica';
```

- continutul formularului ar fi urmatorul:

```
<&frm.form>
<table>
    <tr>
        <td class="text_negru">Denumire:</td>
        <td><&frm.denumire></td>
    </tr>
    <tr>
        <td class="text_negru">Pret:</td>
        <td><&frm.pret></td>
    </tr>
    <tr>
        <td class="text_negru">Caracteristici:</td>
        <td>
            <table>
                <&loop "caracteristici.search::1">
```

```
                                        <tr>
                                                <td><&frm.id_caracteristica[<&id>]{checkbo
                                                x}></td>
                                                <td class="text_negru"><&denumire></td>
                                        </tr>
                                        <&endloop>
                                </table>
                        </td>
                </tr>
                <tr>
                        <td> </td>
                        <td><&frm.goPost><&frm.submit></td>
                </tr>
        </table>
        <&endform>
```

Formularul poate fi adaugat intr-o pagina "adauga.produs" pentru care s-au creat tabelele dupa specificatii si care se va incarca cu un link dintr-o alta pagina si parametru GET "edit" fie zero (0) fie o valoare corecta pentru a vedea concret rezultatul. In primul rand, campurile "denumire" si "pret" vor functiona similar unui formular simplu. Tag-ul de "loop" folosit va selecta toate caracteristicile din tabelul de "caracteristici" si va crea pentru fiecare un camp checkbox cu denumirea "de baza" "id_caracteristica" si cheia id-ul caracteristicii selectate, langa care se va afisa si denumirea respectivei caracteristici. La completarea campurilor "denumire", "pret" si bifarea anumitor caracteristici (in cazul unei inserari, respectiv parametrul GET "edit" cu valoare zero), o linie va fi inserata in tabelul de produse cu denumirea si pretul specificat, apoi id-ul corespunzator acestei linii va fi folosit ca valoare pentru "id_produs" la inserarea altor N linii (in functie de numarul de caracteristici bifate) in tabelul "produse_caracteristici".

De asemenea, se poate adauga inca un camp "valoare" pentru caracteristici, care va fi completat pentru fiecare caracteristica bifata:

```
<&loop "caracteristici.search::1">
    <tr>
            <td><&frm.id_caracteristica[<&id>]{checkbox}></td>
            <td class="text_negru"><&denumire></td>
            <td><&frm.valoare[<&id>]></td>
    </tr>
<&endloop>
```

Campurile pot fi de orice tip, inclusiv select-box-uri. Checkbox-ul cu campul de legatura poate lipsi, si se pot folosi direct campurile aditionale.

3.4.9. Grupuri si validare

Validarea unui formular nu necesita nici o linie de cod php, asistentul phprel se va ocupa de fiecare detaliu pentru dumneavoastra. Trebuie stiut faptul ca validarea standard a unui formular are loc la client, folosind javascript, fara nici un refresh al paginii. Campurile vor fi validate folosind functii javascript sau, acolo unde nu este posibil sau trebuie obtinute informatii suplimentare din baza de date, folosind cereri

asincrone javascript + xml (A.J.A.X.). Suplimentar, se poate activa optiunea de siguranta suplimentara la procesarea formularelor care, printre altele, va revalida formularul in php, inainte de introducerea datelor, fara nici o linie de cod din partea dumneavoastra.

Pentru a valida un camp al unui formular, adaugati atributul de validare "*" (caracterul asterix). La validare, sunt necesare urmatoarele:

- o conditie de validare, de exemplu: campul sa fie nevid, sau sa aiba un anumit tipar (de exemplu, sa respecte formatul unui email), sau sa fie identic cu un alt camp, sau sa fie validat printr-un alt algoritm (complex) de validare specific (folosind o cerere asincrona)
- o denumire "completa" sau o "eticheta" care sa denumeasca respectivul camp (denumirea campului ar putea fi o prescurtare sau neinteligibila, utilizatorul va trebui sa citeasca denumirea completa sau destinata lui)
- un mesaj de eroare in cazul in care campul nu este validat

Atributul de validare specifica metoda sau conditia de validare si suporta toate situatiile descrise in functie de valoarea specificata atributului:

- fara valoare sau cu valoare "std"– validarea standard: camp obligatoriu nevid.
- "confirm" – validarea de tip "confirmare", valoarea campului trebuie sa fie identica cu a unui alt camp (sau eventual trei sau mai multe campuri trebuie sa aiba valoare identica, functionalitate inerenta modului de implementare a "confimarii", neintalnita des in practica). "Confirmarea" se realizeaza intre campurile apartinand primului grup specificat pentru respectivul camp si pentru a functiona campul va avea in mod obligatoriu o functie de control javascript care specifica mesajul de eroare la validare (pentru detalii, mai jos).
- "async.(sir de caractere reprezentand o cerere asincrona)" – validarea campului printr-o cerere asincrona. Sirul de caractere specificat prin (sir de caractere reprezentand o cerere asincrona) poate fi de forma:
  - (handler) – se subintelege caller-ul ca fiind: primul grup din care face parte campul, daca atributul "group" este prezent sau campul in sine daca nu s-a specificat atributul "group". Target-ul va fi calculat si adaugat la cerere automat, acesta fiind un camp hidden adaugat automat de asistent la formular, folosit doar pentru validare, in care se va trimite raspunsul validarii prin cerere asincrona. Campul hidden folosit va fi de forma "Tfrm_Vchk(numar de ordine)".
  - (handler).(caller) – se specifica in clar caller-ul, de exemplu in situatia in care s-a specificat atributul "group" dar cererea trebuie realizata doar cu valoarea campului propriu-zis.
  - Exista si un handler special, "unique", care verifica pentru formularele principale corelate cu un tabel prin regula mysql ca valoarea respectivului camp sa fie unica in baza de date. Sursa handler-ului se gaseste in web/handlers/unique.php. Acest handler va folosi campul ca si caller, neputandu-se specifica un alt caller.
- "(sir de caractere reprezentand un pattern de validare)" – daca valoarea atributului "*" nu se incadreaza in nici una din situatiile de mai sus, se considera ca validarea se realizeaza cu un "pattern" (tipar) de tip "regular expression" specificat in web/settings/form.settings.inc.php prin array-ul asociativ $Tfrm_Vpatterns. Pattern-ul se va specifica prin denumire (cheia din array).

Exemple de validare pot fi:

```
<&frm.camp{*}>
<&frm.parola{pass; *; group="confirmare"}>
```

```
<&frm.confirmaparola{pass; *="confirm"; group="confirmare"}>
<&frm.camp{*="async.unique"}>
<&frm.camp{*="email"}>
<&frm.camp{*="async.disponibil"}>
<&frm.camp{*="async.disponibil.camp"; group="exemplu"}>
```

De notat faptul ca exemplele sunt date pentru a demonstra sintaxa atributului de validare, si unele dintre ele necesita conditii suplimentare pentru a functiona (de exemplu, campul cu "confirm" necesita o functie de control javascript, descrisa mai jos, campul cu "async.unique" trebuie sa apartina unui formular principal corelat printr-o regula mysql cu un tabel care sa contina un camp mysql denumit "camp" , campul cu "async.disponibil" si cel cu "async.disponibil.camp" presupun existenta unui handler de cereri asincrone denumit "disponibil").

De observat faptul ca validarea campului ca email va functiona, "email" fiind considerat un pattern, si care este scris standard in web/settings/form.settings.inc.php. Similar veti putea defini si utiliza alte tipare de siruri ("Regular Expressions").

In ceea ce priveste validarea prin cerere asincrona, trebuie stiut ca:

- se poate trimite valoarea campului sau a unui intreg grup pentru validare. De exemplu atunci cand validarea unui camp "data intoarcere" depinde de "data plecare", cele doua pot fi grupate si vor fi trimise in aceeasi cerere. Se poate specifica in clar numele grupului sau campului folosit.
- valoarea campului sau a grupului va trebui trimisa unui handler care se ocupa de validare. Acest handler respecta regulile oricarui handler a unei cereri asincrone si trebuie scris in formatul necesar in web/handlers/. Ca punct de pornire se poate copia si redenumi "samplevalidate.php". Handler-ul va valida datele si va solicita modificarea valorii unui camp hidden creat automat de phprel in scopul validarii, astfel: daca datele au fost validate, noua valoare a campului va trebui sa fie sirul vid, daca datele nu au fost validate, noua valoare a campului va trebui sa fie mesajul de eroare (eventual in format html, de exemplu continand la sfarsit un tag "
"). Denumirea exacta a campului hidden este primita in parametrul "target" al cererii.
- valoarea propriu-zisa de validat va fi primita in parametrul GET "call", respectiv variabila locala $call din handler. In cazul validarii valorii unui singur camp, valoarea va fi disponibila ca sir de caracter. In cazul validarii unui grup, valorile fiecarui camp din grup vor fi primite ca array asociativ in parametrul "call". Altfel spus, parametrul GET "call" si implicit variabila $call (daca se respecta formatul standard al handler-ului) vor contine un array asociativ de tip camp => valoare reprezentand intregul grup.

O data precizata conditia de validare prin atributul "*", urmatorul pas este stabilirea "etichetei" campului, folosita in mesajul de validare. Mai exact, denumirea completa a campului. Din nou, asistentul va este de ajutor si va incerca sa determine automat denumirea campului. Sa consideram mini exemplul construit in sectiunea 1. phprel la prima vedere si sectiunea 2. Elemente de baza. Pe pagina de "Adauga stire" am validat campul "titlu" pentru a fi siguri ca introducem un titlu nevid. Eroare furnizata de formular in cazul in care nu completam nimic si doar apasam butonul "Salveaza" va folosi denumirea "Titlu", diferita de denumirea campului ("titlu"). Asistentul a determinat "eticheta" in acea situatie nu prin simpla scriere a denumirii campului cu majuscula. Pentru a ne convinge, modificam tabelul html si pe respectivul <td> trecem in loc de "Titlu:", "Titlul complet al stirii:".

La un refresh al paginii si o incercare de a trimite formularul fara a completa campul obligatoriu vom constata ca asistentul a sesizat si folosit schimbarea. Eticheta sau denumirea completa a campului, folosita in mesajele de validare, este determinata automat de asistent astfel:

- prin cautarea unui tag <label> pentru camp (fie pentru denumirea concreta a campului, care coincide cu atributul html "name" al tag-ului rezultat – caz incorect din punct de vedere al tag-ului <label> dar care poate fi folosit din greseala, fie pentru id-ul campului – care va fi de forma "(denumire)(numar de ordine)", caz corect pentru ca tag-ul este de forma <label for="(id)">) si extragerea continutului tag-ului.
- prin cautarea unui tag <td> apropiat si preluarea valorii. In general, formularele vor fi construite intr-un tabel si este de presupus ca un tag <td> se afla in proximitatea campului si contine denumirea sa completa. Acesta a fost cazul de fata, corespunzator formularului de "Adauga stire".
- prin cautarea unui text apropiat (in acelasi tag "parent") cu tag-ul de formular.
  Din eticheta astfel determinata vor fi excluse caracterele finale daca sunt doua-puncte ":", asterix "*" sau ambele "*:"

Daca asistentul nu poate determina corect eticheta intuitiv, se poate preciza eticheta folosind atributul phprel "label" al tag-ului. De exemplu:

```
<&frm.field{*; label="Denumire"}>
```

Mesajele de eroare pot fi standard, configurate in cfgs/advanced/core.settings.inc.php, sau particularizate folosind functii de control javascript. Mesajele standard se configureaza folosind variabilele urmatoare:

- $Tfrm_Vempty
- $Tfrm_Vother
- $Tfrm_Vasync

De asemenea, erorile vor fi afisate in cadrul unui mesaj general de eroare, configurabil folosind $Tfrm_Verrors. Aceste variabile au fost explicate in sectiunea curenta, subsectiunea 3.1.6. Configurari de baza.

Mesajul complet de eroare va fi afisat intr-un div cu id de forma "frm.(denumire formular).errors". Precizarea denumirii unui formular a fost deja discutata in sectiunea curenta, subsectiunea 3.4.1. Tag-ul de formular si atribute. In cazul in care un astfel de div nu exista in pagina in momentul generarii codului de validare, asistentul va adauga imediat deasupra formularului (tag-ul <form>) un div de mesaje de eroare implicit construit pe baza continutului descris in variabila $Tfrm_Vdiv. Id-ul "frm.errors" nu trebuie modificat indiferent de modificarile aduse continutului variabilei, deoarece va fi automat inlocuit cu id-ul necesar.

Nu in ultimul rand, un formular phprel nu poate fi validat automat prin metodele descrise in aceasta subsectiune, decat daca butonul de submit atasat formularului este construit cu un tag de formular specific: <&frm.submit> in cazul formularului principal sau <&frm.submit.(denumire)> in cazul formularelor secundare. "Valoarea" afisata pe acest buton se va schimba pe parcursul procesului de validare, pana la aparitia mesajului de eroare sau trimiterea propriu-zisa a datelor, in mesajul specificat prin $Tfrm_Vloading.

Atunci cand se adauga fiecare camp la functia de validare a formularului, mesajul de eroare se selecteaza dupa cum urmeaza:

- se cauta in sursele html ale paginii curente o functie de control (scrisa in javascript) de forma "validate___(denumire camp)" sau "validate___(denumire formular)___(denumire camp)".

Daca o astfel de functie este gasita, ea va fi apelata si mesajul de eroare intors de functie va fi folosit ca mesaj de eroare la validarea campului respectiv.

- in cazul in care o astfel de functie nu exista, mesajul de eroare la validare pentru respectivul camp va fi construit pe baza mesajelor standard.

Functiile de control se pot scrie explicit:

```
function validate___titlu()
{
        return 'Titlul stirii trebuie completat obligatoriu.<br />';
}
```

sau folosind coduri scurte javascript:

```
&frm.validate___titlu: Titlul stirii trebuie completat obligatoriu.<br />;
```

De notat faptul ca in cazul scrierii explicite a functiei, nu se va lasa nici un spatiu suplimentar intre cuvantul "function" si denumirea functiei si absolut nici un spatiu intre denumirea functiei si paranteza rotunda "(".

In cazul unei validari de tip "confirm" aceasta functie este neaparat necesara.

Grupurile sunt introduse de asistentul phprel pentru a ajuta la validarea datelor, la trimiterea cererilor asincrone si uneori la scrierea rapida a codului javascript. Doar campurile de formular pot fi grupate, un grup continand mereu valorile tuturor campurilor membre. De fapt, un grup este un camp hidden adaugat automat de phprel care contine valorile tuturor campurilor din grup. Sincronizarea valorii grupului se face folosind functii javascript si evenimente "onchange". Un camp de formular poate apartine unuia sau mai multor grupuri. Aceasta apartenenta este specificata prin atributul "group" care poate avea ca valoare un nume de grup sau o lista de grupuri separate prin virgula. Un grup este creat automat de asistent (ca hidden si functii javascript) prin simpla mentionare a lui intr-un atribut "group".

Important: Denumirea grupului nu trebuie sa coincida cu denumirea unui camp de formular sau cu id-ul oricarui alt element din pagina.

Validarea de tip "confirm" necesita obligatoriu un grup, ale carui valori sunt verificate sa coincida. In plus, o cerere asincrona javascript + xml (A.J.A.X.) conform modelului descris contine in mod clasic un "caller" a calui valoare denumita "call" este trimisa handler-ului. Pentru a putea trimite mai multe valori handler-ului nu se pot specifica mai multi "caller"-i dar se poate specifica un caller care contine mai multe valori: acest caller fiind un grup. Valorile vor fi automat "despachetate" la destinatie sub forma unui array asociativ de forma camp => valoare pentru usurinta folosirii.

3.4.10. Formulare si cereri asincrone javascript + xml (A.J.A.X.)

Un camp apartinand unui formular poate initia cereri asincrone in urma anumitor evenimente standard (onchange, onclick, etc.). O astfel de cerere se construieste rapid folosind atributul "async", care va fi inlocuit cu codul javascript necesar realizarii respectivei cereri. Valoarea atributului poate fi de forma:

- (handler).(target) – se subintelege caller-ul ca fiind campul insusi
- (handler).(caller).(target)
- (handler).(caller).(target).(page) – cerere asincrona cu solicitare de pagina. Pentru detalii, consultati sectiunea curenta, subsectiunea 3.7. Cereri asincrone javascript + xml (A.J.A.X.).

- (handler).(caller).(target).(page).(processor) - pentru detalii, consultati sectiunea curenta, subsectiunea 3.7. Cereri asincrone javascript + xml (A.J.A.X.).

In plus, oricare dintre variantele specificate poate fi prefixata de eveniment:

(eveniment): (cerere)

unde (eveniment) este un eveniment standard precum "onchange", "onclick", etc. Cererea va fi atasata atributului html corespunzator evenimentului.

Cateva exemple au fost prezentate in mini exemplul construit in sectiunile 1. phprel la prima vedere si 2. phprel in detaliu. Pentru mai multe detalii, consultati sectiunea curenta, subsectiunea 3.7. Cereri asincrone javascript + xml (A.J.A.X.).

Nucleul phprel va pune la dispozitie si o metoda automata de trimitere a datelor unui formular prin intermediul unei cereri asincrone. Tot ce aveti de facut este sa specificati metoda de trimitere ca fiind "async", folosind atributul phprel "method" al tag-ului de formular.

Important: trimiterea datelor printr-o cerere asincrona (A.J.A.X.) nu implica nici o alta interventie din partea dumneavoastra si va fi realizata automat, dar exista o limita privind dimensiunea datelor din intreg formularul, limita dependenta de browser:

- daca formularul va fi trimis din Internet Explorer, limita este imediat sub 1KB, in functie si de numarul de campuri
- daca formularul va fi trimis din Opera sau Safari, limita este de aproximativ 3KB, in functie si de numarul de campuri
- daca formularul va fi trimis din Mozilla, limita este de aproximativ 7KB, in functie si de numarul de campuri

Se recomanda asadar trimiterea prin A.J.A.X. doar a formularelor mici, cu un numar mic de campuri (10-15, fara campuri de text de mai mult de 100-200 caractere) sau o cantitate de informatie mica.

La procesarea unui formular printr-o cerere asincrona se procedeaza in felul urmator:

- se grupeaza automat toate campurile
- la apasarea butonului submit, dupa validari, se trimite o cerere catre un handler special "formpost" (care este complementat de o secventa de cod din nucleul phprel ce realizeaza propriu-zis salvarea datelor)
- serverul salveaza datele in tabele, sau intoarce eventualele erori de (re)validare
- la final, in locul formularului se afiseaza un mesaj de succes, configurabil din cfgs/advanced/core.settings.inc.php, prin $Tfrm_form_post_async_ok

Formularul trebuie sa fie inglobat intr-un tag "div" cu id-ul de forma "(denumire formular).contents" pentru ca nucleul phprel sa stie ce sectiune de pagina sa inlocuiasca cu mesajul de succes. Alternativ, se poate plasa div-ul intr-o alta locatie, dar nu se recomanda retrimiterea formularului. Daca div-ul cu id-ul respectiv lipseste, mesajul va fi afisat ca mesaj de "alert" javascript.

3.4.11. Pagina de adaugare, editare, vizualizare

In phprel, o singura pagina indeplineste rolurile de pagina de adaugare, editare si vizualizare. O pagina care are atasata o regula mysql si contine un formular cu campurile tabelului reprezentat de regula va functiona ca o pagina de adaugare, daca parametrului GET "edit" i se atribuie valoarea zero (0), ca o pagina de editare daca i se atribuie valoarea cheii primare a unui rand din tabel, si ca o pagina de vizualizare daca parametrului GET "view" i se atribuie valoarea cheii primare a unui rand. Am vazut deja

ca un formular principal impreuna cu o regula mysql poate functiona ca formular de adaugare sau modificare, in functie de cum este construita regula mysql: cu sau fara conditie "where", ceea ce se reduce la valoarea nula sau pozitiva a parametrului GET "edit", daca se respecta conventia recomandata.

In plus, asistentul phprel poate converti o pagina de editare a unei linii intr-o pagina de vizualizare a respectivei linii, prin simpla inlocuire a parametrului GET "edit" cu parametrul GET "view". Formularul va fi convertit automat, in loc de campuri inlocuindu-se valorile corespunzatoare fiecarui camp al tabelului principal, fiecarui camp dintr-un lant de select-box-uri de tip "parent" – "child", fiecarui camp dintr-un tabel secundar. Butonul de submit nu va fi adaugat, si datele nu vor putea fi modificate in nici un fel.

In plus, variabila globala $edit va contine prin conventie sau valoarea parametrului GET "edit" sau valoarea parametrului GET "view", pentru a putea fi folosita in conditii "where" sau alte conditii si operatii fara a testa daca pagina este de editare sau vizualizare. Astfel, paginile de vizualizare sunt contruite automat de asistent.

3.4.12. Upload de fisiere si imagini, generarea thumbnail-urilor

Cu phprel, "upload"-ul de fisiere nu a fost niciodata mai simplu. De fapt, nu trebuie facut aproape nimic pentru a "upload"-a pe server fisierele trimise printr-un camp de tip "file". In primul rand, pentru a upload-a fisiere, trebuie sa existe in formular campuri de tip "file". Asistentul va recunoaste denumiri frecvente de astfel de campuri, date in cfgs/advanced/assistant.inc.php prin $Tasi_guesstype, si va construi automat respectivele campuri de tip "type". In al doilea rand, un formular capabil sa "upload"-eze fisiere pe server trebuie sa fie "encodat" ca "multipart/form-data", acest lucru sepecificandu-se de regula prin atributul "enctype" al tag-ului <form>. In cazul in care un formular contine campuri de tip "file", asistentul va adauga automat atributul "enctype" potrivit la tag-ul de formular, scutindu-va astfel de timp si prevenind o eventuala eroare in situatia in care ati omis sa adaugati respectivul atribut.

O data formularul construit, fisierul trebuie preluat de un script php atunci cand ajunge pe server si prelucrat si/ sau mutat in directorul destinatie. De obicei, fisierului ii corespunde si un camp in tabel, precizand ca respectivul fisier este "upload"-at.

Important: Nu se recomanda upload-ul de fisiere fara un corespondent in baza de date, in plus se recomanda ca respectivul camp din tabel sa contina numele fisierului asa cum apare el in directorul destinatie.

Avand in vedere aceasta conventie, fisierul va trebui prelucrat si/ sau mutat apoi denumirea inserata in campul corespunzator. Evident, dispunem in phprel de functiile de control la scrierea datelor in tabel, cele de forma "input___(camp)" sau "std___input___(tabel)___(camp)" si vom folosi o astfel de functie atasata campului pentru a prelucra si muta fisierul. Vom folosi un apel catre defaultUpload() sau defaultUploadImage(). Un apel tipic ar putea fi:

```
function std___input___(tabel)___(camp) ($val, $d)
{
        return defaultUpload();
}
```

unde (tabel) este denumirea tabelului iar (camp) este denumirea campului care va retine numele fisierului. In mod normal se specifica trei parametrii functiei defaultUpload, care pot sa lipseasca si vor fi completati de asistent pe baza unor presupuneri:

- (denumirea fisierului ca sir de caractere) defaultUpload ($numele_campului, $directorul, $tabelul)
    - $numele_campului – reprezinta denumirea campului prin care s-a realizat "upload"-ul. Acest camp va fi cheie in array-ul supergloal $_FILES. In absenta lui, nucleul phprel va considera numele campului ca fiind prima cheie din array-ul $_FILES care inca nu a fost procesat printr-una din functiile defaultUpload sau defaultUploadImage. Altfel spus, apelul la functiile de upload va functiona fara a specifica primul parametru daca exista un singur camp de tip "file" in formular sau daca ordinea campurile in formular este aceeasi cu ordinea campurilor specificate in $fields in descrierea tabelului (sau in regula mysql, dupa caz).
    - $directorul – reprezinta directorul in care va fi mutat fisierul. In absenta lui, nucleul phprel va considera directorul ca fiind locatia directorului "uploads/" data in configurarile smartlocate (de exemplu, pentru varianta "front" ar putea fi chiar directorul "uploads/" din radacina, in timp ce pentru varianta "back" ar putea fi "../uploads/" din radacina "frontend"-ului), la care se adauga un subdirector cu denumirea tabelului care apare in regula mysql a paginii. Upload-ul fara parametrii va functia astfel din start doar daca formularului ii corespunde o regula mysql in pagina destinatie.
    - $tabelul – reprezinta denumirea tabelului in care se va actualiza denumirea fisierului, tabelul este important pentru a selecta id-ul maxim din tabel, la care se va aduna 1 rezultand un nou "id" care va fi prefixat la denumirea fisierului pentru a construi denumirea finala. Astfel, denumirea finala a fisierului "upload"-at va fi de forma "(id maxim din tabel + 1)_(denumirea initiala)". In cazul in care lipseste, nucleul phprel va considera tabelul ca fiind tabelul specificat in regula mysql a paginii.
      Important: trebuie stiut ca functia de upload va executa un query pentru aflarea id-ului maxim in cazul unui query de inserare (o linie noua in tabel), altfel, va prefixa fisierul cu valoarea variabilei globale $edit.
      Se poate omite orice parametru prin lipsa lui din apelul functiei (in cazul in care lipseste un parametru de la final, de exemplu lipseste ultimul parametru, sau ultimii doi sau toti trei – aceasta este metoda standard php a parametrilor impliciti) sau prin trimiterea lui ca sir vid ".

Astfel, pentru ca upload-ul cu respectiva functie apelata fara parametrii sa functioneze, formularul principal trebuie sa trimita catre o pagina cu o regula mysql corespunzatoare (in mod normal, aceasta conditie este indeplinita, fiind metoda preferata de a procesa formulare) si in directorul corespunzator de "uploads" sa existe un subdirector cu denumirea tabelului (in format mysql). In cazul in care difera un parametru (de regula directorul sau poate campul), precizati parametrii functiei. Functia va intoarce denumirea finala a fisierului (fara directorul in care a fost mutat) sau va intoarce sirul vid in cazul in care a existat o eroare la upload sau fisierul se afla pe lista interzisa de extensii, definita in cfgs/advanced/core.settings.inc.php prin variabila $Tcfg_preventfileextensions. Orice fisier cu extensie aflata pe respectiva lista nu va fi "upload"-at (mai exact, mutat in directorul final).

Foarte similar cu upload-ul de fisiere general este upload-ul de imagini realizat cu functia defaultUploadImage care poate primi aceeasi parametrii ca si defaultUpload si se comporta la fel in cazul lipsei parametrilor. Se recomanda ca functia se fie apelata la upload-ul imaginilor de tip ".gif", ".jpg" si ".png" si se va comporta chiar identic cu functia defaultUpload in lipsa unor indicatii privind

"thumbnail"-uri. Atunci cand se solicita generarea thumbnail-urilor, functia defaultUploadImage va genera si imaginile redimensionate pe langa operatia de a muta fisierul upload-at in destinatie. Imaginile redimensionate se genereaza astfel:

- folosind functia addThumb, apelata inaintea functiei defaultUploadImage, se adauga specificatii intr-un "buffer" temporar, similar folosirii functiei addOpt.
- (true) addThumb ($denumire, $latime, $inaltime, $tip)
    - $denumire – specifica denumirea thumbnail-ului care va corespunde unui subdirector al directorului de upload principal (compus standard din locatia directorului "uploads/" si denumirea tabelului, sau specificat in functia defaultUploadImage). In cazul in care denumirea este sirul de caractere "standard", aceasta regula nu se pastreaza, thumbnail-ul fiind generat in locatia pozei principale, iar poza originala este mutata in subdirectorul "original" al directorului principal. Se poate astfel genera o poza "standard" de alte dimensiuni decat cea upload-ata.
    - $latime – latimea finala a pozei, exacta sau maxima in functie de $tip.
    - $inaltime – inaltimea finala a pozei, exacta sau maxima in functie de $tip.
    - $tip – poate fi "scale" (valoarea implicita daca parametrul este omis), sau "crop". Pentru "scale" se scaleaza poza la cea mai apropiata dimensiune care sa incapa in aria $latime x $inaltime. Astfel, doar una dintre dimensiuni se va pastra (poza se va scala dupa dimensiunea mai mare) cealalta rezultand mai mica decat valoarea data prin $inaltime/ $latime. Pentru "crop" poza se scaleaza mai intai pana la o arie mai mare decat $latime x $inaltime, potrivind cea mai mica dimensiune la valoarea corespunzatoare, cealalta depasind valoarea indicata, apoi poza este taiata pe acea dimensiune pentru a corespunde perfect ariei $latime x $inaltime. Se pierde fragmentul de poza care depaseste dimensiunea pe una dintre axe.
      Important: subdirectoarele necesare trebuie sa existe, nu vor fi create automat.

Se pot defini folosind addThumb oricate "thumbnail"-uri, si vor fi toate create in momentul apelului functiei defaultUploadImage, golindu-se totodata si "buffer"-ul temporar.

Toate directoarele trebuie sa aiba permisiune de scriere.

3.4.13. Procesare si redirect

Formularele procesate pe baza unei reguli mysql trec mai intai prin web/rules/forms.process.php care cuprinde doua "switch"-uri in functie de pagina, primul pentru procesarea propriu-zisa a formularului, al doilea pentru redirect. Procesarea standard ("default") realizata automat se face prin apel direct la functia de procesare processPost($rule). In cazul in care doriti sa realizati verificari si/ sau actiuni suplimentare la procesarea formularului, creati o noua ramura "case" pentru pagina in cauza si adaugati orice cod php necesar. Pentru a procesa automat formularul pe baza regulii, apelati functia de procesare processPost($rule). Trebuie stiut ca pentru validari suplimentare manuale sunt puse la dispozitie doua mijloace:

- $dontredirect – trecut pe "true", pagina nu va fi redirectionata, ramand deci in pagina formularului
- o variabila cu denumire de forma $Tfrm_(denumire formular)_errors – in care se pot specifica erorile afisate in div-ul de eroare.

De asemenea se pot face orice alte query-uri sau operatii inainte si dupa apelarea functiei processPost.

Al doilea "switch" specifica linkul catre care se redirectioneaza pagina dupa procesarea formularului. Creati un "case" pentru pagina curenta si completati valoarea variabilei $redir cu adresa url de redirect. Suplimentar, nucleul phprel va calcula automat o pagina de redirectionare, astfel:

- daca url-ul curent contine un parametru de navigare inapoi (link de intoarcere "nav:back") si nu contine un parametru GET care sa indice posibilitatea existentei unor subpagini a paginii curente, pagina va fi redirectionata inapoi catre pagina pe care utilizatorul a ajuns pe pagina curenta. Pentru detalii privind linkurile de intoarcere, consultati sectiunea curenta, subsectiunea 3.9.1. Url de intoarcere.
- daca pagina este incarcata intr-un popup, are link de intoarcere (nav:back) si nu are un parametru GET care sa indice posibilitatea existentei unei liste sau a mai multor subpagini in popup, popup-ul va fi inchis iar pagina principala reincarcata (se va incarca linkul de intoarcere).
- daca pagina este incarcata intr-un popup, are link de intoarcere (nav:back) si are un parametru GET care indica posibilitatea existentei unei liste sau a mai multor subpagini in popup, popup-ul va fi redirectionat catre el insusi (cu parametrul GET zero) si pagina principala va fi reincarcata (se va reincarca linkul de intoarcere)
- in orice alt caz, pagina va fi redirectionata catre ea insasi dar cu parametrul GET zero (0).
  Important: parametru GET care sa indice posibilitatea existentei unei liste sau unor subpagini a paginii curente inseamna existenta unui parametru GET cu denumire identica cu a unui camp din tabelul specificat in regula mysql. Rationamentul ar fi urmatorul:
  - o pagina simpla de editare, care contine doar un formular, va fi redirectionata inapoi catre pagina din care a provenit (de regula o lista cu liniile tabelului)
  - o pagina de editare care ea insasi contine o lista cu liniile tabelului va contine de obicei si un parametru GET de filtrare a respectivei liste, altfel nu ar contine link de intoarcere pentru ca nu ar fi "provenit" de nicaieri, ci ar fi o pagina principala ea insasi (si va intra in ultima categorie, se va redirectiona catre ea insasi). Daca ea contine lista cu liniile tabelului (si parametrul GET relevant), va fi redirectionata catre ea insasi, nu catre pagina de pe care a provenit, pentru ca utilizatorul ar putea dori sa faca si alte modificari. In cazul in care pagina este incarcata in popup, este oportuna reincarcarea paginii principale, pentru a reflecta modificarile realizate.

Astfel, procesarea si redirectionarea pot fi realizate automat dar se poate si interveni in procese folosind "regulile" de procesare specificate in web/rules/forms.process.php. Fiecare detaliu a fost pus la punct in phprel pentru ca dumneavoastra sa detineti controlul complet asupra tuturor proceselor.

3.4.14. Pagini virtuale

Asa cum am vazut, o pagina trebuie sa aiba atasat obligatoriu un php cu aceeasi denumire in web/pages/ si un template, de regula cu aceeasi denumire, in web/templates/. Un formular creat pe o pagina va trimite in mod evident datele spre procesare unei pagini phprel (aceeasi sau alta). Acea pagina va procesa datele si apoi redirectiona catre o alta pagina phprel (eventual aceeasi). Acest model este de

altfel similar oricarei aplicatii web: introducerea datelor intr-un formular presupune incarcarea a trei pagini (indiferent daca ultimele doua sunt identice cu prima, se vor realiza trei incarcari de pagina):

- o pagina care contine formularul
- o pagina care proceseaza continutul
- o pagina afisata ulterior, in urma procesarii formularului

In phprel, paginile virtuale sunt pagini care nu exista din punct de vedere al codului: nu au un corespondent in web/pages/ sau web/templates/, fiind folosite doar pentru a procesa un formular pe baza unei reguli mysql si apoi pentru a redirectiona catre pagina finala. Avantajul paginilor virtuale este dublu:

- se incarca mai repede si consuma mai putine resurse pentru ca anumite module phprel nu se includ iar anumite portiuni de cod din nucleu central nu se executa
- accelereaza dezvoltarea pentru ca nu necesita timp de creare a fisierelor template si php.

In plus, paginile virtuale pot fi folosite de asa maniera incat asistentul sa intuiasca exact regula mysql necesara (de exemplu, denuminindu-le sugestiv: "virtual.produse", cu referire la tabelul de produse).

Important: prin conventie, pentru a nu crea ambiguitati sau confuzii pentru alti programatori care lucreaza sau vor lucra la proiect, se recomanda sa denumiti paginil virtuale cu prefixul "virtual.".

O pagina poate fi declarata ca virtuala in web/settings/pages.settings.inc.php, prin array-ul asociativ $Tcfg_virtual_pages de forma (denumire pagina) => true. Alternativ, daca in cfgs/advanced/core.settings.inc.php, $Tcfg_auto_virtual_pages este true, paginile vor fi declarate automat virtuale de nucleul phprel daca sunt prefixate de sirul de caractere "virtual." (incheiat cu caracterul punct punct ".") sau "virtual-" (incheiat cu caracterul minus "-"). Se recomanda folosirea prefixului incheiat cu punct ".", cu exceptia cazurilor in care se urmeaza conventia de denumire a paginii pe baza denumirii tabelului si trebuie sa existe mai multe pagini care folosesc acelasi tabel dar cu comportamente diferite. In aceasta ultima situatie paginile se pot denumi dupa conventia: "virtual-(identificator pagina).(denumire tabel in format phprel)" sau o alta conventie similara. Paginile virtuale se vor incarca mai repede si vor putea procesa formularul trimis dar nu vor putea afisa continut propriu-zis si este indicat sa se specifice o locatie de redirect sau sa se creeze conditiile ca nucleul phprel sa determine automat o pagina spre care sa redirectioneze browser-ul utilizatorului.

In afara lipsei unui cod concret atasat paginii, paginile virtuale se comporta ca orice alta pagina: de exemplu, in ceea ce priveste reguli mysql, functii i/o, procesare si redirectionare, se pot crea "case"-uri in respectivele fisiere cu denumirea paginii virtuala pentru a detalia comportamentul paginii. Variabila globala $page va avea numele exact al paginii (inclusiv prefixul "virtual." daca el a fost specificat) si toate modulele phprel se vor comporta obisnuit (cu exceptia faptului ca modulul de liste si cel de template-uri nu vor fi incarcate si o mare parte de cod referitoare la procesarea template-urilor nu va fi rulata).

Nu in ultimul rand, paginile virtuale rezolva simplu o situatie care a fost omisa pana in prezent: daca intr-o pagina exista mai multe formulare care vor sa foloseasca o regula mysql pentru a actualiza date intr-un tabel, cum functioneaza conceptul phprel "o pagina – o regula mysql – un tabel"? Solutia este ca fiecare formular altul decat formularul principal sa trimita (folosind atributul "action") catre o alta pagina (evident, cu parametru GET "edit" stabilit si cu un camp goPost propriu – se vor folosi atatea tag-uri <&frm.goPost> cate sunt necesare). Dar trimiterea catre alte pagini atunci cand acest lucru nu este neaparat necesar inseamna crearea unor paginii suplimentare care doar sa proceseze datele si apoi sa redirectioneze catre alta pagina (eventual aceeasi din care s-a pornit). Solutia ca mai multe formulare sa functioneze in aceeasi pagina devine comoda folosind pagini virtuale – mai ales cand se folosesc cele declarate automat virtual – formularele trimitand catre ele printr-un simplu atribut "action" (in url, pe

langa denumirea paginii virtuale, care se recomanda sa fie denumirea tabelului, sau o conventie mai complicata de denumire dar care sa contina tabelul, se vor indica si un parametru GET "edit" corespunzator si eventual un url de intoarcere pentru redirectionare automata).

In cazul unui formular care trimite spre o pagina virtuala, se recomanda adaugarea la atributul "action" al formularului si a url-ului de intoarcere pentru ca nucleul phprel sa poata reveni in pagina in situatia unor validari suplimentare sau dupa inserarea datelor. Pentru detalii, consultati sectiunea curenta, subsectiunea 3.9.1. Url de intoarcere.

3.5. Liste

Majoritatea proiectelor, in special aplicatiile de tip "back office", afiseaza in peste 50% din pagini listari ale datelor din tabelele mysql. Construirea unei liste poate fi realizata rapid folosind phprel, in general intr-o linie html, uneori cu cateva indicatii date in php. Listele sunt perfect flexibile si pot fi adaptate oricarei situatii, tiparul de design va fi definit o singura data si apoi utilizat pentru a crea sute de liste in intreaga aplicatie. In plus, design-urile se pot porta de la un proiect la altul, se pot schimba usor si o aplicatie poate cuprinde oricate design-uri de lista, fiecare lista folosindu-l pe cel necesar. Datele pot fi transformate rapid in orice format de afisare si toate functionalitatile des intalnite pe web pentru o lista sunt implementate automat.

3.5.1. Tag-ul de lista si functionare

O lista se construieste folosind tag-ul special phprel <&lst.(denumire lista){(indicatii)}>. Orice lista trebuie sa aiba o denumire unica pe pagina (o pagina poate contine oricate liste). Un tag de lista va fi inlocuit cu un continut html construit pe baza unui template. Template-ul unei liste a fost denumit "layout" si poate varia de la o lista la alta. Template-ul standard este scris in web/templates/list-layout.html si web/templates/list-pagination.html (pentru paginare). In plus, template-ul respectiv va fi completat de un "cadru" (denumit "frame"), care va particulariza anumiti parametrii ai template-ului pentru un numar mare de liste, cum ar fi culori, dimensiuni, mesaje, etc. "Frame"-urile se definesc in web/settings/, cadrul standard fiind web/settings/default.frame.php.

Atributele minime necesare unui tag de lista depind de context, mai exact de cata informatie va putea deduce asistentul phprel. Unul dintre atributele de mai jos va fi necesar cu siguranta:

- columns – specifica denumirile coloanelor listei, separate prin virgula, asa cum vor aparea ele in tabel
- fields – precizeaza campurile atribuite coloanelor, separate prin virgula, in ordinea in care sunt corelate cu respectivele coloane

Atunci cand lipseste atributul "columns", coloanele vor avea aceeasi denumire ca si campurile. Atunci cand lipseste atributul "fields", asistentul phprel va deduce pe baza denumirilor coloanelor si a tabelului principal al listei (daca el a fost specificat sau a putut fi determinat) campurile potrivite. Daca asistentul nu determina corect campurile sau trebuie sa accelerati procesul de construire a listei, se pot specifica atat atributul "columns" cat si atributul "fields".

Evident o lista (mai ales ca am vorbit despre "campuri") va fi interconectata cu un tabel sau cu mai multe tabele, din care se vor extrage datele pentru afisare. Specificarea tabelelor se face cu urmatoarele atribute:

- table – precizeaza tabelul principal, care contine majoritatea datelor listate sau contine datele sau liniile care fac obiectul principal de listare.
- supportingtables – tabele aditionale folosite pentru selectarea sau afisarea completa a listei, separate prin virgula. Asistentul va realiza automat join-urile intre tabelele precizate, analizand relationarile dintre ele, directe sau indirecte.

Oricare dintre aceste doua atribute poate lipsi. In primul rand, daca lipseste tabelul principal, asistentul phprel va incerca sa determine tabelul pe baza campurilor date prin "fields", altfel va prelua tabelul fie din regula mysql (daca ea exista) fie va intui o posibila regula mysql pentru respectiva pagina si va prelua de acolo tabelul. In plus, daca nu este specificat atributul "supportingtables", in functie de campurile specificate (sau determinate), asistentul phprel va "explora" pentru a gasi toate tabelele necesare completand automat atributul de "supportingtables". Asistentul folosesti algoritmi complecsi de determinare a tabelelor participante, in functie de toate indiciile de care dispune intr-o anumita situatie.

Este deci usor de vazut ca nu exista o regula privind ce anume trebuie specificat si ce nu, toate depind de context iar asistentul va face tot posibilul sa determine corect si complet toate atributele omise de care este nevoie. Orice indiciu de care dispune va ajuta la construirea listei din cat mai putine atribute. Bineinteles, se pot si preciza cele in mod obisnuit necesare, dar atunci cand credeti ca asistentul ar putea sa deduca un atribut, el poate fi lasat necompletat.

Asadar, o lista preia date dintr-unul sau mai multe tabele (intre care exista un asa numit tabel principal, din care spre exemplu se vor prelua id-urile pentru actiunile de editare, stergere, vizualizare), construieste eventual un filtru si limiteaza in alte moduri numarul informatiilor afisate (prin conditii de tip "group by", "having", etc.), ordoneaza datele dupa anumite campuri si le afiseaza apoi sub forma unei liste construite pe baza unui "layout", a unui "frame" si a indicatiilor listei, atribuind unei coloane valorile unui camp selectat. Vom vedea cum anume se pot afisa valorile mai multor campuri intr-o singura coloana, vom folosi pentru astfel de situatii functiile de transformare a datelor listei.

Construirea listei presupune patru etape:

- definirea listei – consta in indicarea parametrilor si atributelor care particularizeaza sau descriu lista
- construirea automata a query-ului pe baza specificatiilor
- construirea listei propriu-zise folosind motorul de template-uri
- inlocuirea tag-ului de lista cu lista propriu-zisa

Lista contine un "header" – linia "zero" a tabelului cu denumirile fiecarei coloane, cu un "style" css separat si care poate contine ordonari la click pe denumirea coloanei sau prin alte metode create personalizat, si un continut: fiecare linie extrasa din query va deveni o linie in tabelul html construit, cu un "style" css diferit (eventual alternativ, in functie de paritatea numarului de ordine al liniei), cu posibilitatea de click pe o coloana pentru mai multe detalii (furnizate ca o zona suplimentara de continut imediat sub linie) si cu actiuni particularizate pentru fiecare linie (de regula: editare si stergere, optional vizualizare, si orice alta actiune creata de dumneavoastra). Fiecare linie poate avea propriul "style" css, propria clasa css si chiar fiecare celula a tabelului (o coloana a unei linii) poate avea propriile atribute css.

Definirea listei se face folosind atributele tag-ului special de lista (interpretate de asistentul phprel) sau folosind obiectul asociat listei din clasa "lstifc". La definirea listei se stabilesc unul sau mai multi dintre urmatorii parametrii:

- "layout"-ul folosit
- "frame"-ul folosit
- link-ul listei – majoritatea listelor vor contine actiuni sau ordonari dupa anumite coloane, sau detalii la click pe o anumita coloana, toate acestea vor trimite in aceeasi pagina cu anumiti parametrii GET completati, si este necesar un link "de baza" al listei de la care se porneste pentru a contrui aceste url-uri. Probabil in toate cazurile, link-ul listei este calculat automat de asistentul phprel in functie de url-ul curent, minus anumiti parametrii GET (presupusi a fi) irelevanti sau contextuali.
- ordonarea listei – dupa unul sau mai multe campuri, care au sau nu corespondent coloane din tabelul final.
- mesajul afisat cand nu exista randuri – stabilit de obicei standard in "frame".
- numarul de inregistrari pe pagina – lista este paginata automat pe baza indicatiilor
- numarul maxim de pagini afisate in paginare – daca o lista se intinde pe foarte multe pagini, sa spunem 1000, nu vor putea fi toate afisate in link-urile de navigare intre pagini. Sa presupunem ca se pot afisa maximum 11 pagini, se vor afisa paginile de la "pagina curenta -4" pana la "pagina curenta +4", plus prima si ultima pagina. Utilizatorul va putea selecta oricare din acestea pentru navigare, si apoi in functie de pagina noua se vor recalcula paginile afisate.
- template-ul zonei de detalii suplimentare per linie
- coloana care expandeaza sau restrange la click zona de detalii
- inaltimea minima a listei
- coloanele dupa care poate fi ordonata lista prin click. La un click va fi ordonata ascendent, la un al doilea click descendent (daca implicit la afisare lista este ordonata ascendent, la primul click va fi direct ordonata descendent). Coloana dupa care este ordonata lista va fi afisata in *italics*.

Construirea query-ului este realizata automat de modulul de liste pe baza indicatiilor puse la dispozitie de asistentul phprel (indicatii construite de asistent pe baza indicatiilor date de dumneavoastra). Query-ul poate cuprinde mai multe tabele legate fie prin "INNER JOIN" fie prin "LEFT JOIN". Campurile fiecarui tabel sunt selectate prefixate de denumirea tabelului, "campurile" finale fiind de forma "(denumire tabel)___(denumire camp)". Campurile pot fi referite oriunde este necesar atat cu denumirea completa cat si cu denumirea scurta (doar denumirea campului, dar respectivul camp va trebui sa fie primul care are respectiva denumire in ordinea data a tabelelor). De exemplu, se poate filtra pe baza unui camp POST "(tabel)___(camp)" sau preciza in atributul "fields" campuri de forma "(tabel)___(camp)" (in plus, asistentul va adauga automat respectivele tabele la atributul "supportingtables"). De asemenea, se pot introduce subquery-uri in query-ul principal, precum si conditii (aditionale fata de cele calculate automat) "where", "group by", "having". Lista poate fi filtrata automat pe baza datelor din POST.

Important: Filtrul automat pe baza datelor din POST functioneaza pe baza corelarii denumirilor parametrilor POST cu denumirile "campurilor" selectate in query (cheile array-ul rezultat in urma mysql_fetch_assoc, cuprinzand atat campuri concrete din tabele cat si alte rezultate ale unor expresii selectate, de exemplu "CONCAT(a, b, c) AS x" sau orice alta expresie/ subquery) astfel:

- se distinge intre camp "nativ" (existent in tabel exact cu denumirea respectiva) si camp "virtual", selectat in urma unei expresii sau subquery
- un filtru bazat pe un camp "nativ" va fi adaugat la conditia "where" a query-ului, in timp ce un filtru bazat pe un camp "virtual" va fi adaugat la conditia "having".

- un camp "nativ" sau "virtual" care exista cu exact aceeasi denumire in POST si are valoare nevida, va fi filtrat pe baza de egalitate: campul/ expresia respectiva sa fie egala cu valoarea data in POST
- un camp "nativ" sau "virtual" care apare in POST sufixat de "::like", de exemplu "nume::like", va fi filtrat pe baza unei conditii mysql LIKE: campul/ expresia respectiva sa fie LIKE '%(valoare POST)%' unde (valoare POST) este valoarea data in POST. Util pentru filtrari cu valori partiale.
- un camp "nativ" sau "virtual" care apare in POST prefixat de "s::" va fi filtrat pe baza unei conditii de forma:
    - daca exista in POST si campul prefixat de "e::", conditia va fi "(s::) <= (camp/ expresie) AND (camp/ expresie) <= (e::)", unde (s::) este valoarea din POST a campului prefixat de "s::", (camp/ expresie) este campul sau expresia in cauza, iar (e::) este valoarea din POST a campului prefixat de "e::". Util pentru restrangerea unui camp la un interval (de exemplu o data, sau o suma, sau un pret, etc.).
    - daca nu exista in POST si campul prefixat de "e::", conditia va fi similara cu cea precedenta dar doar "(s::) <= (camp/ expresie)". Util pentru selectarea valorilor unui camp peste o anumita valoare.
- un camp "nativ" sau "virtual" care apare in POST prefixat de "e::" va fi filtrat pe baza unei conditii similare cu cea precedenta, de forma "(camp/ expresie) <= (e::)". Util pentru selectarea valorilor unui camp pana la maxim o anumita valoare.
- un camp "nativ" care apare in POST prefixat de "sp::" va fi filtrat dupa urmatorul algoritm:
    - se cauta un alt camp, inca nefolosit intr-o astfel de filtrare, de care sa fie prefixat de "ep::" (altul decat campul curent gasit, si neaparat un camp "nativ").
    - daca un astfel de camp exista, deci exista o pereche "sp::(camp1)" – "ep::(camp2)", se adauga o conditie "where" de forma: "((sp::) <= (camp1) AND (camp1) <= (ep::)) OR ((sp::) <= (camp2) AND (camp2) <= (ep::)) OR ((camp1) <= (sp::) AND (ep::) <= (camp2))", unde (sp::) este valoarea din POST a campului (unu) prefixat de "sp::", (ep::) este valoarea din POST a campului (doi) prefixat de "ep::", (camp1) este primul camp gasit (cel care a fost gasit prefixat de "sp::") iar (camp2) este al doilea camp gasit (cel care a fost gasit prefixat de "ep::").
    - Aceasta conditie este de fapt o conditie de tip "perioada": in tabel exista doua campuri reprezentand o data de inceput si o data de sfarsit, iar cele doua campuri din POST reprezinta o perioada data printr-o data de inceput si o data de sfarsit. Filtrul va selecta liniile care "au loc" (au datele cuprinse) in perioada data. De exemplu, pentru un tabel de evenimente care contine (printre altele) doua campuri din tabelul mysql "startdate" si "enddate", se pot trimite prin POST doua date calendaristice in campurile html "sp::startdate" si "ep::enddate" si vor fi selectate evenimentele care sunt "active" in perioada data.
      Important: in cazul filtrelor de tip interval sau perioada valorile acceptate vor fi sau numerice sau date calendaristice care vor fi automat convertite la unixtime.

Filtrul automat pe baza datelor din POST nu va functiona in cazul formularelor care adauga informatii in baza de date (nu trebuie sa va faceti griji atunci cand in aceeasi pagina aveti un formular de adaugare si o lista, formularul nu va filtra lista).

Este important de stiut si faptul ca asistentul phprel va construi automat o conditie "where" pe baza parametrilor GET, in masura in care acestia sunt identici ca denumire cu campuri ale tabelului principal. De exemplu, daca tabelul principal are un camp "id_categorie", si exista un parametru GET "id_categorie" cu valoarea 7, asistentul va adauga automat conditia "id_categorie='5'".

Ordonarea listei poate fi rezultatul unei ordonari implicite specificate in denumirea listei sau unei cereri de a ordona lista dupa o anumita coloana. Trebuie stiut ca nucleul phprel va face toate eforturile sa coreleze cele doua ordonari, astfel:

- ordonarea implicita va fi analizata si coloana relevanta, daca exista, va fi trecuta in *italics* pentru a reflecta ordonarea. In plus, la primul click pe respectiva coloana, ordonarea se va realiza descendent, deoarece lista va fi deja ordonata ascendent.
- ordonarea solicitata pentru o anumita coloana va fi completa cu eventualele ordonari secundare date implicit. De exemplu, daca se ordoneaza implicit dupa doua campuri, si apoi se solicita ordonarea dupa o coloana, doar primul camp din ordonare va fi inlocuit, iar al doilea va fi pastrat, nu va dispare. Acest lucru este util, de exemplu, la ordonarea unei liste de persoane: ordonarea implicita ar fi dupa campurile "nume" si "prenume", iar la click din nou pe nume, lista ar fi reordonata descendent dupa "nume" dar s-ar pastra si ordonarea ascendenta dupa "prenume", ca ordonare secundara, in caz contrar, la ordonarea dupa o coloana, pentru aceleasi valori are respectivului camp lista nu ar mai fi ordonata in nici un fel.

De asemenea, trebuie stiut ca asistentul phprel va ordona implicit lista folosind parametrul $defaultorder al tabelului principal, sau ordonand ascendent dupa campul "id" al tabelului principal.

Limitarea query-ului la un anumit numar de randuri se realizeaza automat de nucleul phprel, in baza unui numar de randuri pe pagina si a unui parametru GET specificand pagina curenta. Sunt calculate automat si numarul total de pagini, prin executarea unui query suplimentar. Pe baza acestor informatii se genereaza automat sectiunea de paginare.

Construirea propriu-zisa a listei se realizeaza dupa ce query-ul corespunzator a fost realizat. Lista va contine zona de "header", zona de "continut" (liniile propriu-zise) si eventual zona de paginare. In plus, va putea contine si zona de detalii atasata unei anumite linii. Dupa adaugarea "header"-ului conform specificatiilor css, se adauga fiecare linie pe baza corespondentei "coloana" – "camp" si dupa aplicarea tuturor functiilor de transformare si de control pentru fiecare valoare a campurilor, fiecare caracteristica a liniei si fiecare caracteristica a celulei. Fiecare rand al query-ului va determina adaugarea unui rand la lista, sub fiecare coloana trecandu-se valoarea (eventual schimbata prin functiile de transformare) campului corespunzator.

Se construiesc astfel array-uri continand toate informatiile necesare iar lista este inlocuita in template ("layout") prin inlocuirea a unui tag de "loop" pentru "header", doua tag-uri "loop" pentru continut, un tag "loop" pentru paginare si un tag "loop" pentru detalii. Intregul continut, chiar si de mii de linii, va fi inlocuit prin doar doua bucle, intr-un timp foarte scurt. Lista se va incarca rezonabil ca timp chiar si pentru doua mii de inregistrari pe o singura pagina.

Daca ati solicitat o lista paginata, paginarea va fi adaugata automat de nucleul phprel imediat sub lista. Amplasarea paginarii poate fi schimbata scriind manual in locul dorit tag-ul de includere al paginarii: <&include "(denumire lista specificata in tag-ul de lista).pagination" />. Pentru ca inlocuirea sa fie realizata cu succes, tag-ul de lista (mai exact, denumirea listei) trebuie sa fie unic pe pagina.

3.5.2. Atribute

Asistentul phprel poate primi in tag-ul de lista si va interpreta urmatoarele atribute:

- columns – prezentat in sectiunea curenta, subsectiunea anterioara.
- fields
- table
- supportingtables
- layout – primeste ca valoare denumirea "layout"-ului folosit. Layout trebuie sa fie compus din doua template-uri, salvate in web/templates/, cu denumirile de forma: (denumire layout)-layout.html, (denumire layout)-pagination.html. Layout-ul standard are denumire "default" si nu trebuie specificat in lista.
- frame – primeste ca valoarea denumirea "frame"-ului folosit pentru a particulariza layout-ul. "Frame"-urile se salveaza in web/settings/, in fisiere cu denumiri de forma: (denumire frame).frame.php, iar cel implicit este "default.frame.php", incarcat pentru toate listele care nu au precizate un frame.
- noheader – fara valoare, indica faptul ca lista nu are zona de "header" (denumirile coloanelor nu exista si/ sau zona respectiva nu va aparea)
- minheight – indica inaltimea minima a listei, in pixeli
- rowsperpage – numarul de randuri afisate pe pagina.
- maximumpages – indica numarul maxim de pagini afisate in zona de paginare. In cazul in care lista are mai multe pagini decat specificate prin acest numar, vor fi afisate doar prima pagina, ultima pagina, si paginile din proximitatea paginii curente, in numar total de "maximumpages".
- order – primeste ca valoare o expresie mysql care va fi folosita la ordonarea datelor din lista (mai exact, la ordonarea query-ului).
- sortablecolumns – denumirile campurilor corespunzatoare coloanelor pentru care se poate solicita ordonare ascendenta sau descendenta prin click pe denumirea coloanei. Suplimentar, se poate specifica ordinea sortarii la primul click, adaugand la denumirea coloanei o valoare "DESC" precedata de doua-puncte ":":

  ```
  sortablecolumns="titlu, data: DESC"
  ```
- method – metoda de join a tabelelor, fie nespecificata sau "simple" pentru "INNER JOIN" fie "LEFT" pentru "LEFT JOIN".
- group – conditia mysql "group by", exact asa cum va aparea in query
- having – conditia aditionala mysql "having", exact asa cum va aparea in query
- noautofilter – fara valoare, indica faptul ca lista nu va fi filtrata automat pe baza parametrilor din POST.
- detailstemplate – denumirea template-ului din web/templates/ folosit pentru a construi zona de detalii. Nu trebuie sa contina extensia ".html". Daca acest atribut nu este specificat, asistentul phprel va cauta in directorul de template-uri un fisier cu denumirea de forma: "(page).(denumire lista).(sufix template de detalii).html", unde (page) este pagina curenta, preluata din variabila globala $page, (denumire lista) este denumirea listei specificata in tag-ul de lista, iar (sufix template de detalii) este o valoare setata implicit ca "details" si care poate fi schimbata din cfgs/advanced/assistant.inc.php in variabila $Tasi_detailstemplatepostfix. Daca nici acest fisier "standard" nu exista, se considera ca lista nu are zona de detalii atasata liniilor.

- detailscolumn – daca lista are zona de detalii, specifica prin denumirea campului atasat, coloana la click pe care se va expanda/ inchide zona de detalii. Valoarea atributului va fi asadar o denumire de camp, campul corespunzator coloanei pe care se expandeaza detaliile. In plus, asistentul phprel va intui automat care este coloana respectiva in cazul in care nu este specificata, astfel:
    - daca expresia de la $namefield din descrierea tabelului principal exista printre campurile atasate coloanelor listei, acea coloana va fi coloana "de detalii".
    - altfel, daca primul camp (atasat primei coloane) este denumit "id", se va alege a doua coloana, in caz contrar se va alege prima coloana.
- align – indica alinierea datelor corespunzatoare diverselor coloane. Valoarea atributului va avea una din formele:
    - (camp corespunzator coloanei): (tip align: left/ center/ right), (camp corespunzator coloanei): (tip align: left/ center/ right), etc. unde (camp corespunzator coloanei) este denumirea unui camp corelat cu coloana ale carei date vor fi aliniate intr-un anumit mod. Diferitele aliniamente vor fi specificate separate prin virgula.
    - (camp corespunzator coloanei): (camp corespunzator coloanei): (camp corespunzator coloanei): (tip align left/ center/ right), (camp): (tip align), etc. O metoda alternativa pentru acelasi tip de aliniament este sa specificati campurile corespunzatoare respectivelor coloane unul dupa altul separate prin doua-puncte ":", ultimul fiind tip-ul de "align". Aceasta metoda poate fi scrisa impreuna cu metoda clasica separate prin virgula.
- halign – similar cu "align", dar indica alinierea denumirilor de coloana.
- width – similar ca metoda de scriere cu "align", dar indica latimile fiecarei coloane, date in procente sau pixeli (sufixate de "%" sau, respectiv, "px")
- noactions – specifica faptul ca lista nu are actiunile clasice de "editare" sau "stergere" sau orice alte actiuni din coloana de actiuni (coloana de de actiuni va lipsi)
- setactionsanyway – fara valoare, indica asistentului phprel ca trebuie sa declare actiunile standard chiar daca dumneavoastra ati declarat deja alte actiuni pentru lista.
- editlink – link-ul catre pagina de "editare", folosit ca punct de plecare (la cere se adauga un parametru GET "edit" cu valoarea cheii primare a liniei pentru care se adauga actiunea) pentru a construi actiunea de editare dar si cea de vizualizare (in mod similar, dar adaugand parametrul GET "view")
- editinpopup – specifica faptul ca actiunile de editare si eventual vizualizare vor fi deschise in popup-ul principal.
- setviewaction – indica asistentului sa creeze, pe langa cele doua actiuni standard de "editare" si "stergere", si actiunea optionala de "vizualizare" (pagina de editare va fi convertita automat in pagina de vizualizare de catre nucleul phprel)

Trebuie stiut ca listei i se adauga automat o ultima coloana de "actiuni" cu "campul" (conventional) "lst.actions", care tine actiunile definite standard de asistent. Aceasta coloana nu va fi adaugata daca se adauga atributul "noactions" la indicatiile listei. Pentru a "muta" coloana de actiuni pe o alta pozitie in tabel, se poate corela o anume coloana de coloana de actiuni specificandu-i campul corespunzator ca fiind "lst.actions". De exemplu, se poate specifica atributul "columns" ca fiind "Id, Nume, Prenume, Actiuni, Data nastere, Status" si atributul "fields" ca fiind "id, nume, prenume, lst.actions, data_nastere, status". De

asemenea, "lst.actions" se poate folosi oriunde se poate specifica in mod normal o caracteristica a unei coloane, de exemplu in "align", "halign", "width", etc.

3.5.3. Functii de transformare

Datele afisate intr-o lista trec prin functiile de control la citirea datelor din mysql, in cazul de fata functiile de transformare specifice listelor, ale caror formate si prioritati au fost descrise in sectiunea curenta, subsectiunea 3.2.3. Functii de control, comportament standard, comportamente diferite pe pagina. Trebuie stiut suplimentar ca orice camp pentru care nu exista o functie de control proprie poate trece prin functia comuna "transform___(denumire lista asa cum apare in tag)___unknown", daca ea exista:

```
function transform___(lista)___unknown ($camp, $valoare, $linie)
{
        ...
}
```

unde $camp va fi denumirea campului, $valoare va fi valoarea acestuia iar $linie va fi un array asociativ continand intreaga linie selectata (sub forma camp => valoare initiala).

In plus, toate campurile, dupa ce au trecut sau nu prin diferitele functii de control, trec prin functia comuna "transform___(denumire lista asa cum apare in tag)___any", daca ea exista:

```
function transform___(lista)___any ($camp, $valoare_transformata, $linie)
{
        ...
}
```

unde $camp va fi denumirea campului, $valoare_transformata va fi valoarea campului dupa ce eventual au fost aplicata o functie de control, iar $linie este un array asociativ continand intreaga linie selectata (sub forma camp => valoare initiala).

Functiile de transformare pot avea diverse scopuri, inclusiv afisarea a doua campuri din baza de date intr-o singura coloana, de exemplu:

- pentru <&lst.lista{table="persoane"; columns="Id, Prenume si Nume, Data nastere"}>, specificam fields="id, prenume, data_nastere"
- adaugam in php-ul atasat paginii functia de transformare:

```
function transform___lista___persoane___prenume ( $val, $d )
{
        return $val.' '.$d['persoane___nume'];
}
```

Asa cum am vazut deja, array-ul liniei va contine campurile in formatul "(tabel)___(camp)", campurile fiind selectate in aceasta forma in query-ul mysql construit automat.

3.5.4. Functii de control a liniilor

In afara functiilor de control la citirea din baza de date, listele dispun de o serie de functii de control la nivel de linie care va permit sa schimbati dinamic clasa css sau atributele css ale unei linii sau

ale unei celule, sa adaugati continut inainte sau imediat dupa anumite linii, sa limitati actiunea de stergere sau de editare pentru anumite linii si sa stabiliti pentru care linii sunt disponibile detalii si pentru care nu. Functiile de control sunt urmatoarele:

- class___(denumire lista) ($row, $class, $field, $d) - va intoarce clasa css corespunzatoare unei celule sau linii. $row va fi numarul liniei (prima avand numarul 1), $class va fi clasa implicita care se va aplica, $field va fi denumirea campului corespunzator coloanei curente, $d va fi un array asociativ corespunzator intregului rand selectate
- style___(denumire lista) ($row, $style, $field $d) - similar functiei precedente dar va intoarce atributul "style" al celulei sau liniei.
- class___(denumire lista)___0 ($class, $field) – similar functiei class___* dar va intoarce clasa css corespunzatoare "header"-ului listei, pentru coloana specificata (sau intregul header).
- style___(denumire lista)___0 ($class, $field) - similar functiei precedente dar va intoarce atributul "style" corespunzator "header"-ului liste, pentru coloana specificata (sau intregul header).
- class___(denumire lista)___norecords () - similar functiei class___* dar va intoarce clasa css a mesajului afisat in cazul in care lista nu contine inregistrari.
- style___(denumire lista)___norecords () - similar functiei precedente dar va intoarce atributul "style" al mesajului afisat in cazul in care lista nu contine inregistrari.
- rowheader___(denumire lista) ($row, $d) - fiecare linie poate avea propriul "header", o zona html care va aparea inaintea linie. Functia va intoarce codul html adaugat inaintea unei anumite linii (linia este primita prin parametrul $row si functia va decide ce anume trebuie sa intoarca, sau daca trebuie sa intoarca ceva). Parametrul $d va fi un array asociativ continand randul selectat
- rowfooter___(denumire lista) ($row, $d) - similar functiei precedente, dar pentru "footer"-ul unei anumite linii, o zona html care va aparea imediat dupa linie.
- rowdetails___(denumire lista) ($row, $d) - va intoarce "true" daca linia data prin $row poate contine zona de detalii sau "false" daca linia data nu poate contine
- rowaction___(denumire lista)___(denumire parametru actiune) ($row, $name, $d) - pentru schimbarea denumirii/ imaginii/ continutului html corespunzator unei actiuni a unei linii, unde (denumire lista) este denumirea listei asa cum apare in tag-ul de lista, iar (denumire parametru actiune) este parametrul GET corespunzator actiunii respective, de exemplu "edit" pentru actiunea standard de editare sau "del" pentru actiunea standard de stergere. $row va if numarul randului curent (incepand cu 1), $name va fi denumirea/ continutul html standard al actiunii, $d va fi un array asociativ continand linia selectata.
- rowcheck___(denumire lista)___(denumire parametru actiune) ($row, $d) - pentru limitarea afisarii unei actiuni pe anumite linii: daca functia intoarce true pentru $row-ul dat, actiunea respectiva va fi afisata, daca functia intoarce false actiunea nu va fi afisata.
- rowconfirm___(denumire lista)___(denumire parametru actiune) ($row, $confirm, $d) - pentru schimbarea mesajului standard de confirmare a unei actiuni care cere confirmare (de exemplu, actiunea de stergere). $row va fi numarul randului curent (incepand cu 1), $confirm va fi mesajul standard de confirmare iar $d un array asociativ cu intreaga linie selectata.

Toate aceste functii vor fi folosite de nucleul phprel in masura in care ele exista, la fel ca orice alta functie de control. Astfel, daca functia exista (de exemplu este definita in php-ul atasat paginii) ea va fi

sesizata de nucleul phprel si folosita cu semnificatia data mai sus, altfel, textele/ continutul html sau css va fi cel standard, iar functionalitatile controlate vor fi implicit disponibile.

3.5.5. Configurare si feedback

Asa cum am vazut deja, configurarea listei poate fi lasata in sarcina asistentului phprel, specificand atributele de configurare direct in tag-ul de lista pe langa care asistentul va intui sau deduce el insusi alte atribute. Dar lista poate fi configurata si din php in masura in care anumite atribute sau specificatii sunt usor de construit in php. Fiecarei liste ii corespunde un obiect din clasa "lstifc" in care se pot specifica urmatoarele proprietati:

- link – linkul "de baza" al listei folosit pentru a genera toate linkurile (de ordonare, de paginare, de detalii)
- order – expresia mysql de ordonare a query-ului
- norecords – mesajul afisat in cazul in care nu exista inregistrari. Daca nu este precizat, mesajul standard este cel specificat in "frame"-ul listei
- rowsperpage – numarul de randuri pe pagina
- detailstemplate – denumirea (fara extensie) a template-ului html corespunzator zonei de detalii.
- detailscolumn – coloana (specificata prin denumirea campului corelat) la click pe care se expandeaza/ inchide zona de detalii corespunzatoare liniei
- minheight – inaltimea minima a listei in pixeli
- sortablecolumns – lista de coloane separate prin virgula si referite prin denumirea campului corelat, pentru care se poate solicita la click ordonare ascendenta sau descendenta dupa respectiva coloana

Aceste proprietati sunt interpretate direct de modulul de liste si nu necesita asistent phprel. Urmatoarele proprietati sunt interpretate de asistentul phprel:

- method – metoda de "join" a tabelelor: poate fi "simple" pentru "INNER JOIN" (metoda standard daca atributul sau proprietatea method lipseste) sau "LEFT" pentru "LEFT JOIN".
- subquery – o expresie mysql care va fi folosita fara modificari ca subquery in query-ul construit automat de nucleul phprel pentru lista. Se poate specifica orice subquery, de exemplu: "(SELECT camp FROM tabel WHERE tabel.id_camp_legatura = tabel_exterior.id LIMIT 0,1) AS camp", dar folosind pseudonim (identificatorul mysql "AS") pentru a denumi intr-un fel campul.
  Important: un camp al unui subquery sau orice alt camp pentru care se specifica pseudonim folosind identificatorul mysql "AS" nu va fi prefixat de o denumire de tabel sau de cele trei underscore-uri "(tabel)___". Campul va fi referit oriunde doar in forma sa scurta (forma completa neexistand), exact cum apare dupa "AS", inclusiv functiile de transformare vor fi direct de forma "transform___(camp asa cum apare dupa AS)".
- group – expresia mysql de grupare a query-ului
- where – o expresie mysql "where" adaugata la expresia "where" finala (care va contine si filtrul pe baza de POST dar nu va mai contine conditia where automata calculata de asistent pe baza parametrilor GET). Proprietatea va fi folosita frecvent pentru a adauga filtrari mai complicate.
- having – o conditie mysql "having" adaugata la conditia "having" finala (care va contine eventual si filtrul pe baza de POST)

- editlink – likul "de baza" folosit pentru a crea linkurile de editare si eventual vizualizare. Se adauga automat parametrul GET "edit" sau "view" impreuna cu valoarea cheii primare a tabelului principal corespunzatoare liniei respective
- setviewaction – indica asistentului sa adauge si actiunea de vizualizare (care nu este adaugata implicit alaturi de cea de "editare" si "stergere")
- setactionsanyway – indica asistentului sa adauge actiunile standard chiar daca dumneavoastra ati definit deja una sau mai multe actiuni proprii
- editinpopup – indica asistentului faptul ca editarea si vizualizarea se vor deschide in popup-ul principal (definit in web/settings/contants.inc.php, prin variabila globala $openPopup).

In plus, obiectul din clasa "lstifc" atasat listei va primi feedback din partea nucleului phprel privind anumite detalii concrete stabilite la construirea listei, feedback disponibil in anumite proprietati care sunt menite sa fie citite (spre deosebire de cele descrise pana acum care trebuie scrise):

- _count – numarul total de linii al listei
- _sLim – (linia) "limita stanga" de la care a inceput selectarea datelor pe pagina curenta.
- _rLim – (linia) "limita dreapta" pana la care s-au selectat datele pe pagina curenta.
- _page – numarul paginii curente afisate
- _pages – numarul total de pagini
- _query – va contine query-ul propriu-zis executat de nucleul phprel pentru a selecta datele paginii curente
- _isdefined – true sau false dupa cum lista a fost definita (configurata) sau nu
- _isparsed – true sau false dupa cum lista a fost inlocuita sau nu in template

Orice configurare realizata direct in php folosind obiectul atasat listei va avea prioritate si va suprascrie atributul similar indicat asistentului direct in tag.

Obiectul atasat fiecarei liste va trebui sa aiba acelasi nume ca si lista (denumirea listei data in tag) si sa fie de tip lstifc. Cu asistentul phprel activat, obiectul va fi declarat automat si dumneavoastra il veti folosi direct. De exemplu, pentru un tag <&lst.lista{...}> obiectul atasat va fi $lista disponibil direct in php -ul atasat paginii (se poate scrie, de exemplu, $lista->minheight='500';).

In cazul listelor construite automat de asistentul phprel, proprietatile de feedback din cadrul obiectului nu vor fi disponibile in php-ul atasat paginii din simplul motiv ca asistentul intervine si creaza lista mai tarziu, dupa ce toate operatiile efectuate de dumneavoastra in php au fost executate. Pentru a folosi o anumita proprietate din feedback mai tarziu, de exemplu pentru o inlocuire de tag, definiti o variabila globala in php ca referinta la proprietatea dorita:

$var = &$(denumire lista)->(proprietate);

iar in html, de exemplu, se poate inlocui direct valoarea variabilei:

Proprietatea este: <&var>

Astfel, valoarea variabilei se va modifica atunci cand valoarea proprietatii obiectului se va modifica, cele doua fiind legata prin referinta creata.

3.5.6. Cadru si template de lista

Cadrul implicit atribuit unei liste este cel denumit "default", disponibil in web/settings/default.frame.php. Alternativ, se poate specifica folosind atributul "frame" in tag-ul de lista orice alt cadru creat de dumneavoastra. La crearea unui nou cadru, se poate duplica si modifica cel

standard. De asemenea, exista predefinit cadrul "phprel" folosit in interfata de management a aplicatiei - Web Assistant. Acest cadru poate fi folosit doar impreuna cu "layout"-ul cu acelasi nume, "phprel" (specificat prin atributul "layout") si cu clasele css necesare predefinite.

Un cadru contine o variabila locala denumita $Tlst_frame, array asociativ cu urmatoarele proprietati:

- mainwidth – latimea listei (ca atribut html "width")
- mainheight – inaltimea listei (ca atribut html "height")
- mainstyle – atributul html "style" al tabelului exterior care contine intreaga lista
- mainclass – atributul html "class" al tabelului exterior care contine intreaga lista
- headercolor – culoarea de fundal a "header"-ului listei (continand denumirile coloanelor)
- headerstyle – atributul html "style" al fiecarei coloane din header
- headerclass – atributul html "class" al fiecarei coloane din header
- headerheight – inaltimea header-ului listei (continand denumirile fiecarei coloane)
- headeralign – alinierea standard a fiecarei coloane din header
- evencolor – culoarea de fundal a liniilor pare
- evenstyle – atributul html "style" al fiecarei coloane a liniilor pare
- evenclass – atributul html "class" al fiecarei coloane a liniilor pare
- oddcolor
- oddstyle
- oddclass – similar dar pentru linii impare
- rowheight – inaltimea (atributul html "height") fiecarei linii
- rowalign – alinierea standard a fiecarei coloane din continutul listei
- paginationpos – textul care apare in paginare explicand ce linii sunt afisate pe pagina curenta si cate inregistrari sunt in total. Se pot folosi simbolurile {sLim} – pentru "limita stanga" (numarul de ordine al primei linii afisate), {rLim} – pentru "limita dreapta" (numarul de ordine al ultimei linii afisare) si {total} – pentru numarul total de linii.
- paginationclass – clasa css a textului explicativ privind inregistrarile afisate
- paginationlinks – formatul textului prin care sunt afisate paginile (linkurile de navigare). Se pot folosi simbolurile {previous: (denumire link pagina precedenta)} pentru linkul de "pagina precedenta", {pages: (separator pagini)} pentru afisarea linkurilor spre alte pagini si {next: (denumire link pagina urmatoare)} pentru linkul de "pagina urmatoare".
- paginationlinksclass – clasa css a textului continand linkurile de navigare a paginilor
- paginationcurrentpageclass – clasa css a paginii curente in textul de navigare
- setupautofilter – true daca filtrarea automata pe baza de POST poate fi folosita, false altfel
- setupconfirmmessage – mesajul standard de confirmare pentru actiunile care necesita confirmare (de exemplu stergerea)
- setupshowlistifnodata – true daca lista va fi afisata si atunci cand nu are inregistrari, false altfel
- setupnorecords – mesajul standard afisat in cazul in care nu exista inregistrari (daca lista este vizibila conform setupshowlistifnodata)

Acestea vor constitui setarile standard ale listei dupa incarcarea layout-ului, orice schimbare efectuata printr-un atribut al tag-ului, o proprietate a obiectului de configurare sau o functie de control suprascriind setarile standard.

Un template de lista este format din doua template-uri cu denumiri de forma (denumire)-layout.html – continand lista propriu-zisa - si (denumire)-pagination.html – continand zona de paginare,

unde (denumire) este denumirea "layout"-ului. Layout-ul folosit implicit este disponibil in web/templates/list-layout.html si web/templates/list-pagination.html. In ceea ce priveste template-ul pe baza caruia se genereaza lista propriu-zisa, trebuie stiut ca lista este compusa dintr-un tabel cadru si un tabel cu continutul propriu-zis alcatuit la randul lui dintr-un <tr> corespunzator header-ului listei si un "loop" care multiplica un al doilea <tr> pentru fiecare linie. Noile template-uri vor fi construite prin duplicarea template-ului de baza si vor trebui sa contina tag-urile "loop" initiale (daca un loop nu exista, el nu va fi inlocuit dar va afecta functionarea corecta a listei). Un exemplu ar fi layout-ul "phprel", disponibil in web/templates/phprel-layout.html, care adauga un efect de mouseover pe linii. Se pot observa diferentele intre cele doua template-uri. Template-ul zonei de paginare este mult mai simplu, continand un singur tabel cu cateva variabile ale unei bucle cu un singur rand corespunzatoare in mare setarilor din cadru cu toate inlocuirile efectuate.

3.5.7. Construirea detaliilor suplimentare pentru o anumita linie

Daca ati specificat denumirea unui template pentru zona de detalii si coloana la click pe care respectiva zona va fi expandata/ inchisa, sau daca ati creat fisierul de template cu denumire standard pe care asistentul il va cauta implicit si pe baza caruia va specifica automat un "detailstemplate" si un "detailscolumn", respectivul template va fi folosit pentru a genera o zona de detalii pentru fiecare linie a listei. La click pe coloana "de detalii" corespunzatoare unei anumite liste, o zona de detalii se va expanda imediat sub linie cu continutul html construit folosind template-ul specificat si eventual php-ul atasat.

In primul rand, construirea propriu-zisa a zonei de detalii are la baza un template din web/templates/ care va cuprinde continutul html afisat. Zona de detalii, la fel ca zona de "header" sau "footer" a unei linii (a nu se confunda cu "header"-ul listei insasi – zona in care apar denumirile coloanelor), este afisata intr-un tag <td> care se intinde pe toate coloanele unei linii. Tag-ul care contine detaliile nu are nici un atribut sau clasa css proprie, de aceea toate formatarile de text, asezarile in pagina, culoarile si alte atribute vor fi integrate de dumneavoastra in continutul html propriu-zis. Continutul html poate contine orice tag special phprel cu exceptia urmatoarelor:

- un nou tag de lista – o lista nu poate contine in interiorul ei o noua lista
- un tag de formular care construieste un "form" sau un buton "submit" (nici un tag <&frm.form>, <&frm.form.(denumire)>, <&frm.submit>, <&frm.submit.(denumire)>)

Orice alte tag-uri speciale phprel pot fi continute si imbricate pe oricate nivele, similar oricarui alt "include" secundar. Template-ul de detalii poate contine chiar alte tag-uri de incluziune la randul sau si se va comporta similar unui "include" secundar, putand avea propriul php atasat in web/pages/ (cu aceeasi denumire de fisier ca si template-ul), care va fi inclus in acelasi timp si va beneficia de functia de auto-globalizare efectuata automat de asistentul phprel precum si va respecta toate celelalte reguli ale includerii unui element secundar. Trebuie retinut insa ca aceste elemente vor fi ultimele care vor fi incluse si nu admit tag-urile precizate mai sus.

In plus, intregul continut html din template va fi automat inclus intr-o bucla cu un singur rand care va fi inlocuita cu array-ul asociativ rezultat la selectarea datelor pentru lista: altfel spus, orice camp selectat in query-ul listei pentru linia curenta va fi disponibil si inlocuit prin simpla utilizare a unui tag de variabila cu denumirea sa (de exemplu, daca lista este construita folosind tabelul "persoane" cu campurile "id, nume, prenume, data_nastere", si are doar coloanele "Id, Nume, Prenume", se poate afisa in zona de detalii data nasterii cu urmatorul continut: Data nastere: <&data_nastere>). Aceasta bucla va fi inlocuita

automat iar continutul folosit la inlocuire va fi trecut prin toate functiile de transformare corespunzatoare campurilor care alcatuiesc array-ul asociativ. De asemenea, array-ul asociativ respectiv (care corespunde datelor liniei curente) va fi disponibil in php-ul atasat zonei de detalii in variabila locala $drow.

Zona de detalii poate avea un template indicat de dumneavoastra sau preluat automat de asistent daca are denumirea "(page).(denumire lista).(sufix zona detalii)" unde (page) este pagina curenta (preluata din variabila globala $page), (denumire lista) este denumirea listei asa cum apare in tag-ul de lista iar (sufix zona detalii) este un sir de caractere initial setat la "details" configurabil din cfgs/advanced/assistant.inc.php prin $Tasi_detailstemplatepostfix care diferentiaza template-urile de detalii de celelalte template-uri. Se recomanda folosirea standardului de denumire implicita atat pentru claritate cat si pentru viteza (template-ul va fi preluat automat de asistentul phprel, nu va trebui sa-l indicati). De asemenea, coloana "de detalii" va fi in multe cazuri determinata automat de asistent.

Incarcarea propriu-zisa se face standard printr-o cerere asincrona (A.J.A.X.) construita si aplicata automat de nucleul phprel pe coloana respectiva. Cererea asincrona va solicita continutul zonei de detalii corespunzatoare liniei respective printr-o cerere "cu pagina" (vezi sectiunea curenta, subsectiunea 3.7. Cereri asincrone javascript + xml). Oricate zone de detalii pot fi expandata simultan prin aceasta metoda. In cazul in care browser-ului clientului nu suporta cereri asincrone, zona de detalii va fi incarcata transparent prin redirectionare catre un link care va expanda respectiva zona. Doar o singura zona de detalii poate fi expandata simultan prin aceasta metoda.

Mesajul sau continutul html (eventual imaginea) care apare in timpul incarcarii detaliilor prin cererea asincrona poate fi configurat in cfgs/advanced/core.settings.inc.php prin $Tlst_loadingmessage.

3.6. URL rewrite

In phprel, url-urile "prietenoase" sau "SEO" sunt folosite natural. Cu ajutorul unei metode inovative de a rescrie url-urile, nu va fi nevoie nici de ".htaccess" nici de efort pentru a construi aplicatii bazate pe cele mai "umane" adrese uniforme de localizare a resurselor. Veti scrie url-uri mai usor, si le veti folosi ca si cum sunt in format clasic.

3.6.1. Setarea paginii si a parametrilor GET

La recrierea url-ului in format clasic, modulul de rescriere va considera url-ul ca fiind alcatuit din fragmente separate prin slash "/", fiecare fragment putand fi:

- denumirea sau un fragment din denumirea paginii
- denumirea (sau, mai rar, un fragment din denumirea) unei variabile
- valoarea (sau, mai rar, un fragment din valoarea) unei variabile

Scopul rescrierii este interpretarea url-ului pentru a obtine:

- pagina solicitata ($_GET['page'])
- arrray-ul asociativ superglobal GET, cu variabile si valori ca si cum url-ul ar fi fost in format clasic

Trebuie stiut ca folosind modulul de rescriere inclus in nucleul phprel, paginile vor functiona atat cu url-uri "prietenoase" cat si cu url-uri clasice simultan. Mai mult, din punct de vedere al programarii nu

veti intalni nici un inconvenient legat de preluarea datelor din url, nucleul phprel aranjand lucrurile sa para ca si cum url-ul a fost unul clasic.

Avand in vedere impartirea in fragmente a url-ului si scopul propus, se pot stabili cateva metode prin care fragmentele vor fi recompuse pentru a forma array-ul GET rezultat. Regulile dupa care url-ul va fi rescris vor fi date in web/settings/rewrite.rules.php, astfel:

- fragmentele de url vor fi primite in array-ul simplu $req, unde $req[0] va fi primul fragment, $req[1] va fi al doilea, etc.
- in functie de primul fragment se stabileste printr-un "switch" ce "regula" va recompune fragmentele in denumirea paginii, variabile si valori. Se adauga asadar un "case" pentru cazul curent (cu valoarea asteptata a primului fragment), sau se modifica algoritmul de pe ramura "default" pentru a include situatiile in care nici macar valoarea primului fragment nu este o constanta
- o data stabilita premisa privind primul fragment, se stabileste pagina in variabila locala $pag. De exemplu: $pag = $req[0]; (daca pagina este denumita de insusi primul fragment) sau $pag = $req[0].'.'.$req[1]; (daca e denumita de primele doua) sau $pag = 'login'; (daca este o constanta) etc.
- daca pagina a fost stabilita, "restul" de fragmente (sau eventual toate, daca este cazul) trebuie sa fie variabile GET si valori ale acestor variabile. Se precizeaza in primul rand incepand cu al catelea fragment incepe sirul de variabile si valori GET. Se seteaza deci o valoare pentru variabila locala $offset, de exemplu: $offset = 1; va insemna ca variabilele si valorile GET incep cu al doilea fragment (care are indicele 1 in array-ul local $req).
- de la indicele dat pana la finalul array-ului, nucleul phprel va considera alternativ: un fragment va fi denumirea unei variabile, urmatorul valoarea ei, urmatorul va fi denumirea altei variabile, urmatorul dupa acesta valoarea ei, etc. Aceasta insiruire de denumiri si valori nu este intotdeauna folosita, existand foarte multe exceptii dupa cum urmeaza:
    - un fragment poate fi direct valoare pentru o variabila subinteleasa, acest lucru fiind specificat prin array-ul local $vars: $vars[(indice)] = '(denumire variabila GET)';
    - un fragment poate fi valoare "de legatura" pentru o variabila, de exemplu atunci cand contine denumirea unui produs in loc de id-ul sau. De regula in url-urile clasice, parametrul GET "produs" contine un id (valoarea cheii primare din tabel), in timp ce in url-urile "lizibile" parametrul GET "produs" contine o denumire. Pentru ca transformarea sa fie completa, valoarea intermediara (denumirea) trebuie transformata in valoarea finala (id). Acest lucru este specificat prin array-ul local $substitute: $substitute[(indicele valorii)] = '(sir instructiune getData)'; unde (indicele valorii) este indicele in array-ul $req al valorii intermediare (neaparat un indice de valoare, nu de denumire de variabila), iar (sir instructiune getData) va fi un apel catre o subfunctie getData pentru un anume tabel caruia i se va trimite ca parametru valoarea intermediara iar rezultatul va fi considerat valoarea finala. Astfel, $substitute[2] = 'produse.toid'; va insemna ca valoarea de la indicele 2, fie aceasta o denumire de produs de exemplu: "phprel", va fi trimisa printr-un apel getData('produse.toid', 'phprel'); la subfunctia "toid" care va primi denumirea si o va converti in id, sa spunem 5. Valoarea finala a parametrului GET va fi "5". Evident, se poate folosi orice alta functie getData, dar de regula se foloseste "toid". In cazul de fata, functia "toid" a necesitat existenta unui $namefield in descrierea tabelului "produse".

- similar, o variabila rezultata poate contine o valoare intermediara ("de legatura") care trebuie transformata in id-ul corespunzator tot printr-un apel la o subfunctie getData. Acest lucru se realizeaza folosind array-ul asociativ local $assocsubstitute: $assocsubstitute['(nume variabila GET)'] = '(sir instructiune getData)'; De regula, veti scrie directivele $assocsubstitute dupa "switch"-ul implicit din rewrite.rules.php, pentru a fi disponibile indiferent de ramura/ regula pe care se merge: de exemplu, variabila "produs" va fi inlocuita cu id in loc de denumire indiferent de pagina pe care apare. Se trece deci sub switch $assocsubstitute['produs'] = 'produse.toid';

Functia "toid" va fi cel mai des folosita pentru a pastra coerenta cu url-urile de tip clasic, si folosirea altor functii nu este recomandata. In cazul rescrierii url-urilor, la sirul de instructiuni getData care contine un apel catre subfunctia "toid" se pot adauga urmatorii parametrii:

- ::globalize – de exemplu: 'produse.toid::globalize', care va indica nucleului phprel sa creeze o variabila globala cu denumirea variabilei din GET corespunzatoare ca array asociativ continand intreaga linie din tabel corespunzatoare id-ului extras prin subfunctia "toid"
- ::(denumire parent) – de exemplu: 'subcategorii.toid::categorie', care va indica nucleului phprel sa foloseasca in selectarea id-ului corespunzator denumirii si valoarea variabilei GET data prin (denumire parent) ca valoare a primului camp care relationeaza un alt tabel ca "parent" din descrierea tabelului dat in sirul instructiunii. In cazul de fata, sa presupunem ca in descrierea tabelului "subcategorii" exista o linie $relate['categorii'] = 'as.parent.by.id_categorie'; se va folosi il query-ul de selectare, pe langa conditia de tipul "(namefield subcategorii)='(valoare variabila GET curenta)'" si conditia "id_categorie='(valoare variabila GET "categorie")'". Aceasta conditie suplimentara se foloseste atunci cand denumirea nu identifica unic linia, fiind necesar si un camp suplimentar care trebuie sa fie intotdeauna primul camp care relationeaza un alt tabel ca "parent".
  Important: variabila GET specificata prin (denumire parent), in cazul de fata "categorie" trebuie sa se fi procesat si calculat in rewrite inaintea variabilei curente, altfel conditia nu va fi valida.
  La folosirea ambilor parametrii aditionali, "::globalize" va sta intotdeauna ultimul.

O data toate aceste reguli stabilite, controlul este preluat din nou de nucleul phprel care va interpreta toate variabilele de control date pentru a "citi" url-ul fragment cu fragment si a-l transforma in array-ul superglobal $_GET final, impreuna cu variabila $_GET['page'] finala.

Metoda prin care veti rescrie url-ul depinde in totalitate de dumneavoastra, existand foarte multe variante posibile in functie de situatie, bazate pe cele cateva directive puse la dispozitie de phprel. Indicarea regulii se realizeaza astfel intr-un mod inovativ si intuitiv, de regula in cateva randuri simple php, si fara nici o interventie in ".htaccess". Pentru cateva exemple, recititi sectiunea 2. Elemente de baza, subsectiunea 2.1. Url rewrite.

3.6.2. Caractere speciale

Caracterele speciale care nu trebuie sa apara intr-un fragment de url, de exemplu "/", "&", "+", "?", " ", dar si orice alt caracter special pot, bineinteles, fi inlocuite pentru a nu interveni (negativ) in procesul de rescriere a url-ului. Astfel, in cfgs/core.settings.inc.php prin array-ul asociativ $Tcfg_urlrewritechars se pot specifica aceste caractere si un corespondent al lor "url-friendly" (acceptat intr-un url). De exemplu, se

obisnuieste in practica sa inlocuim caracterul spatiu " " cu underscore "_". Dupa stabilirea acestor caractere speciale, un sir se poate "coda" pentru a fi "url-friendly" folosind functia rewrite___output (de exemplu, de cate ori trebuie scris un atribut "href" continand o denumire, acea denumire va fi trecuta prin functia rewrite___output) iar nucleul phprel va "decoda" automat toate fragmentele la rescrierea url-ului (folosind functia rewrite___input).

Important: decodarea automata a url-urilor are loc inainte de includerea fisierului web/settings/rewrite.rules.php, deci array-ul local $req va contine valori "decodate". In plus, decodarea se realizeaza invers decat "codarea" (pornind de la valorile array-ului $Tcfg_urlrewritechars carora le corespund cheile).

3.6.3. Operatii suplimentare la setarea paginii

Trebuie stiut ca in web/settings/rewrite.rules.php nu se pot apela anumite functii phprel (nucleul phprel nu este inca incarcat, cu exceptia unor module strict necesare) si nu este recomandat sa se faca alte operatii decat stabilirea diferitelor valori pentru variabilele locale ce definesc rescrierea url-ului. Toate operatiile suplimentare, apelurile de functii, sau alte calcule pot fi facute web/settings/extended.rewrite.rules.php, care dispune de un "switch" in functie de variabila locala $pag setata in web/settings/rewrite.rules.php.

3.6.4. Rewrite si validarea url-urilor

Asa cum am precizat, anumiti parametrii GET cu risc ridicat, de exemplu "edit", "view", "del" sau "(denumire lista):del" precum si alti parametrii configurati de dumneavoastra, vor fi validati folosind un cod, pentru a preveni accesul neautorizat la anumite resurse, sau stergerea (rau-)intentionata a unor date. Mai multe detalii despre validarea si securitatea url-urilor veti descoperi in sectiunea 4. Securitate si performanta, subsectiunea 4.1.1. Validarea url-urilor. Vor exista insa cazuri cand unui astfel de parametru cu risc ridicat, care necesita cod de validare, i se va atribui manual o valoare in timpul rescrierii url-ului, din diverse motive. Avand in vedere ca validarea are loc intr-o etapa ulterioara rescrierii url-ului, atribuirea va fi anulata, parametrul GET cu risc ridicat neavand un cod de validare. Pentru a atribui o valoare unui astfel de parametru veti folosi intotdeauna array-ul asociativ local $secureallow in web/settings/rewrite.rules.php. Astfel, $secureallow['(denumire parametru GET)'] = '(valoare)'; va stabili valoarea parametrului GET si va inregistra respectivul parametru ca "validat".

Suplimentar, valoarea data prin (valoare) poate fi de forma "GET (parametru)" semnificand faptul ca parametrul GET dat prin (denumire parametru GET) trebuie sa primeasca valoarea unui alt parametru GET dat prin (parametru) si sa fie inregistrat ca "validat".

De exemplu, pentru url-ul "produs/phprel" vom scrie urmatoarea regula de rewrite:

```
case 'produs':
        $pag = $req[0];
        $offset = 0;
        $secureallow['edit'] = "GET produs";
break;
```

```
...
$assocsubstitute['produs'] = 'produse.toid';
```

Insemnand urmatoarele:

- pagina destinatie va fi cea specificata in primul fragment de url, mai exact "produs"
- calculul variabilelor GET va incepe cu primul fragment (de indice 0), deci cu "produs"
- in lipsa unei directive $vars, nucleul phprel va interpreta dupa schema clasica: variabila/valoare/variabila/valoare/etc., deci "produs" va fi denumire de variabila, iar "phprel" va fi valoarea intermediara, schimbata printr-un apel la getData('produse.toid', 'phprel') conform directivei $assocsubstitute, si rezultand valoarea finala "5" (de exemplu). Mai exact, $_GET['produs'] va fi "5".
- in final, conform directivei $secureallow, $_GET['edit'] va fi egal cu $_GET['produs'] deci va avea valoarea 5, si va fi inregistrat ca "validat", ne mai fiind nevoie de codul de validare.

3.6.5. Completarea automata a url-urilor din template

Intr-un proiect care foloseste url rewrite, se creaza o situatie dificila in ceea ce priveste toate atributele "href" sau "src" (sau alte asemenea atribute indicand spre o locatie de pe server) care in mod normal sunt scrise relativ dar vor fi considerate relativ la un url incorect din cauza fragmentelor de url introduse strict pentru "lizibilitate" si care nu indica spre o locatie reala de pe server. In mod obisnuit, toate url-urile din template-urile site-ului vor fi prefixate sau de adresa de baza a site-ului sau de directorul in care se afla site-ul relativ la radacina ("/"). Acest lucru este realizat automat de nucleul phprel, care va adauga la toate url-urile din template-uri adresa data prin variabila globala $websiteURL reprezentand adresa de baza a site-ului (sau directorul relativ la radacina, in functie de configurare).

Nu trebuie deci sa va preocupati de url-uri, le veti scrie in mod natural relativ la radacina site-ului sau, eventual, relativ la radacina directorului web/ (de exemplu, veti scrie doar "images/" si se va subintelege ca v-ati referit la "(websiteURL)web/images/") si nucleul phprel va modifica template-urile pentru ca fiecare link sa corespunda locatiei reale.

3.7. Cereri asincrone javascript + xml (A.J.A.X.)

Cererile asincrone reprezinta trimiterea de solicitari/ informatii serverului http dupa incarcarea paginii (dupa prima cerere intiala, deci "asincron" cu prima cerere). Prima cerere solicita incarcarea unei anumite pagini, acea pagina va contine la randul ei cod javascript care va inainta noi cereri serverului http (in functie de anumite evenimente). Respectivele cereri vor fi solutionate de server, raspunsul fiind impachetat in format xml pentru facilitarea citirii si aplicarii lui la destinatie. Asadar, o cerere asincrona este realizata in urmatorii pasi:

- o functie javascript personalizata de dumneavoastra apeleaza o alta functie javascript de trimitere a unei solicitarii catre serverul http
- functia respectiva foloseste metode interne ale browser-ului pentru a lansa o noua cerere catre serverul http si atribuie unui eveniment (de "schimbare a starii cererii") un apel catre o functie de preluare a rezultatului

- cererea ajunsa la server este preluata de o aplicatie, rezultatul ei este calculat apoi impachetat sub forma unui xml si trimis inapoi ca raspuns la cerere
- la primirea rezultatului, evenimentul de schimbare a starii cererii este activat si functia de preluare a rezultatului primeste propriu-zis xml-ul rezultat
- xml-ul este despachetat de functia javascript de procesare a rezultatului si indicatiile sunt indeplinite in concordanta cu functionalitatile construite in functia de procesare

O serie de pasi din realizarea unei cereri asincrone pot fi complet automatizati si de aceia se va ocupa in intregime nucleul phprel. Pentru ceilalti doi pasi, phprel va pune la dispozitie mijloace si ajutor pentru a accelera procesul de realizare a codului. Astfel, asistentul phprel va ajuta sa construiti codul de apel al functiei javascript de trimitere a unei cereri, in timp ce nucleul phprel si inovatiile puse la dispozitie va vor ajuta in scrierea rapida a unui "handler" – un php care sa calculeze rezultatul cererii.

3.7.1. Handler

O cerere asincrona este preluata de nucleul phprel si inaintata unui "handler" – un php care se ocupa de calculul rezultatului. In primul rand, o cerere asincrona phprel contine urmatorii trei parametrii GET:

- async – denumirea handler-ului cerut
- call – o valoare propriu-zisa trimisa in cerere, de regula valoarea unui camp sau a unui grup de campuri
- target – elementul destinatie pentru care se formuleaza cererea, uneori folosit doar formal

Se cere astfel unui "handler" (specificat prin "async") sa intoarca un raspuns pe baza unei valori (data prin "call") pentru un element destinatie (dat prin "target"). Bineinteles, acesta este doar un model simplificat; cererea va putea contine multiple valori (intotdeauna grupate si disponibile in handler sub forma unui array asociativ de forma camp => valoare) si va putea modifica multiple elemente destinatie, de mai multe tipuri.

Din punct de vedere structural, un handler poate fi privit ca o pagina:

- in timp ce pagina se solicita prin parametrul GET "page", handler-ul se solicita prin parametrul GET async
- unei pagini ii corespunde un php cu aceeasi denumire in web/pages/, in timp ce unui handler ii corespunde un php cu aceeasi denumire in web/handlers/
- o pagina trebuie sa afiseze un continut html pe baza unor parametrii GET; un handler trebuie sa raspunda cu un continut xml generat pe baza a doi parametrii GET
- atat in scrierea unei pagini cat si in scrierea unui handler se pot folosi functiile phprel de acces la baza de date, functii ajutatoare phprel, sau orice alte functii disponibile public in nucleul phprel (mai putin, in cazul unui handler, functii legate strict de generarea de continut html, cum ar fi cele din modulul de formulare, de liste, motorul de template-uri, etc.)

Asadar, pentru construirea unui handler se creeza un nou php in web/handlers/. Pentru rapiditate, se poate duplica si redenumi fisierul web/handlers/samplehandler.php. Un handler contine trei zone, dintre care doar una va trebui practic construita:

- o zona de preluare a parametrilor "call" si "target" care de obicei nu se modifica
- o zona de procesare a cererii, care de obicei contine codul necesar care genereaza raspunsul sub forma unor array-uri asociative temporare

- o zona de inaintare a raspunsului catre nucleul phprel pentru impachetare, care de obicei nu se modifica din punct de vedere structural, alcatuita din variabile locale ce vor prelua valorile array-urilor temporare construite anterior

Astfel, la intrare se primesc doi parametrii:

- $call = $_GET['call']; - valoarea parametrului "call" a cererii, poate fi sau o valoare sau un array asociativ de forma camp => valoare in cazul trimiterii valorii unui grup
- $target = $_GET['target']; - valoarea parametrului "target" a cererii, un sir de caractere

La iesire se vor scrie una sau mai multe din urmatoarele patru variabile locale:

- $content – array de forma (id div) => (continut html), pentru schimbarea continutului (prin atributul "innerHTML") unui div din pagina
- $displays – array de forma (id div) => (true/ false), pentru schimbarea vizibilitatii unui div din pagina (prin atributul "style.display")
- $selects – array de forma (id select-box) => (array de optiuni construit cu una dintre metodele de optiuni), pentru schimbarea optiunilor unui select-box dat prin id
- $fields – array de forma (id camp) => (valoare), pentru schimbarea valorii (prin atributul "value") unui camp de formular. Nu se pot bifa checkbox-uri folosind aceasta metoda.
- $calls – array de forma (denumire functie) => (sir vid/ valoare parametru) pentru apelarea unor functii javascript dupa incheierea procesarii raspunsului la cerere

Pentru a pastra formatul handler-ului dar si pentru claritate, se recomanda ca fiecare variabila care va fi modificata sa fie calculata intr-o variabila temporara, eventual cu nume similar (de exemplu $flds pentru $fields) si atribuita variabilei de raspuns corespunzatoare in ultima parte standard a handler-ului.

Array-ul local $content poate contine ca valoare inclusiv continut html, iar $calls poate apela functii fara parametru setand valoarea presupusului parametru ca sir vid "". Se poate raspunde cu un numar oricat de mare de modificari, folosind una sau mai multe (eventual toate) formele de raspuns.

Array-urile sunt preluate de nucleul phprel si impachetate ca xml, apoi trimise ca raspuns. Raspunsul este preluat de o functie javascript scrisa intr-o extensie javascript a nucleului phprel, processRequest, despachetat si aplicat conform instructiunilor date. Trebuie stiut ca se pot construi cereri asincrone care sa fie procesate de o alta functie decat cea implicita, pentru maximum de flexibilitate in situatiile care ar necesita un alt "procesor de raspunsuri".

3.7.2. Contruirea unei cereri standard sau cu pagina

O cerere asincrona standard inseamna o solicitare trimisa catre un handler dupa modelul prezentat pana in prezent. Intr-o cerere standard, nu exista conceptul de "pagina" cu care ne-am familiarizat deja, modulele de template-uri, formulare, liste, etc. nefiind macar incarcate. O alta inovatie adusa de phprel, in spiritul scrierii unui cod cat mai clar si cat mai concis, este posibilitatea executatiei paralele a unui solicitari de pagina si de handler. Motivul este unul cat se poate de simplu: anumite cereri asincrone vor furniza ca raspuns continutul html al unui element de pagina. De exemplu, cererea asincrona realizata automat de phprel pentru a incarca zona de detalii corespunzatoare unei linii dintr-o lista, trebuie sa incarce de fapt elementul "(denumire lista).details" disponibil la acesarea linkului fara A.J.A.X. de pe coloana de detalii. Lista, cand primeste un anumit parametru GET, construieste pentru linia corespunzatoare zona de detalii, in elementul "(denumire lista).details" pe care apoi il include imediat dupa linia respectiva. La incarcarea printr-o cerere asincrona, ar fi foarte dificil sa reproducem exact

continutul elementului in absenta listei propriu-zise. Similar, in anumite situatii, ar fi foarte greu pentru dumneavoastra sa reproduceti continutul unui element de pagina de care aveti nevoie, in absenta respectivei pagini.

Asadar, dintre cei doi parametrii GET "page" sau "async", nu va fi unul care sa aiba prioritate, ambele solicitari vor fi incarcate simultan. Se va prelua cererea asincrona, apoi se va incarca si procesa pagina complet, iar la final cererea va fi inaintata handler-ului. Handler-ul poate decide sa foloseasca pagina sau nu, indiferent de decizie continutul paginii este apoi ignorat si raspunsul handler-ului este trimis inapoi. Nu este nevoie sa scrieti un handler special care sa "anunte" ca va folosi continut de pagina, daca handler-ul a fost scris sa foloseasca pagina si solicitarea s-a realizat impreuna cu un url de pagina, atunci totul va functiona de la sine. Un exemplu de handler care foloseste pagina este chiar cel folosit de modulul de liste, "listdetails.php" din web/handlers. Se foloseste variabila globala $parser si metoda "retrievefile" pentru a obtine continutul elementului dorit, care eventual se trimite inapoi browser-ului folosind variabila locala $content.

De notat faptul ca orice handler poate functiona cu o cerere de pagina in fundal si orice cerere asincrona va functiona corect cu o solicitare de pagina in fundal, chiar daca o foloseste, chiar daca nu. De aceea, se recomanda sa fiti atent pentru a nu genera astfel de cereri paralele atunci cand nu este cazul. O cerere care nu foloseste in nici un fel continutul unei pagini se recomanda sa fie inainta ca cerere standard, fara pagina, pentru a nu consuma inutil resurse.

Construirea unei cereri asincrone presupune generarea codului javascript de apel al functiei asyncrequest, functie definita in extensia javascript a nucleului phprel care se ocupa de trimiterea unei cereri catre server. Apelul catre respectiva functie poate fi construit in doua moduri:

- manual, din php, folosind un apel la functia php makeAsyncRequest sau makeAsyncPageRequest
- asistat, din html, folosind un tag interpretat si inlocuit automat de asistentul phprel, <&async{...}>

In plus, cererile asincrone vor putea fi realizate asistat si fara tag-ul respectiv in contextul unor elemente de pagina care pot accepta cereri asincrone, cum ar fi trimiterea formularelor folosind method="async", generarea unei cereri asincrone pentru un anumit eveniment al unui camp, folosind atributul "async", validarea unui camp prin A.J.AX. folosind atributul "*", etc. Aceste cereri implicite vor respecta formatul de scriere al tag-ului <&async{...}> eventual cu unele prescurtari, in timp ce tag-ul special va respecta formatul cerut de functiile makeAsyncRequest sau makeAsyncPageRequest.

La construirea manuala a unei cereri se va apela una din functiile indicate, rezultatul fiind preluat intr-o variabila care apoi va fi inlocuita folosind un tag de variabila in locul in care cererea ar trebui sa aiba loc (obligatoriu in cadrul unui eveniment/ functii/ sectiuni de cod javascript).

Functia makeAsyncRequest primeste urmatorii parametrii:

- (apel javascript de realizare a cererii ca sir de caractere) makeAsyncRequest ($handler, $caller, $target = false, $processor = $false)
- $handler va fi denumirea handler-ului solicitat
- $caller va fi denumirea elementului din pagina a carui valoare va deveni parametrul "call" al cererii. De exemplu, daca in $caller se va trece denumirea unui camp, valoarea din campul respectiv va deveni parametrul "call". Valoarea unui checkbox va fi "1" sau "0" dupa cum checkbox-ul este bifat sau nu. De notat ca daca nu exista un element cu un astfel de id in pagina, $caller va deveni direct parametrul "call" al cererii. Se pot asadar construi cereri cu parametru "call" constant.

- $target va fi parametrul "target" al cererii
- $processor, daca este specificat, va fi denumirea functiei de procesare a raspunsului, in situatia in care se foloseste alta functie decat cea implicita. Functia "procesor" va primi ca parametru un id al cererii, si va putea apela functia getResponse((id cerere)) pentru a obtine rezultatul xml

Rezultatul va fi un sir de caractere reprezentand apelul la "asyncrequest" particularizat pentru cererea curenta, care va putea fi preluat intr-o variabila si folosit acolo unde este nevoie.

Functia makeAsyncPageRequest are parametrii similari, cu exceptia faptului ca primul parametru va fi url-ul suplimentar cerut. Atentie, url-ul va trebui sa fie unul complet, de exemplu "index.php?page=produse" dar relativ la radacina site-ului.

De cele mai multe ori insa, veti folosi ajutorul oferit de asistent pentru a construi cererile asincrone, utilizand tag-ul special <&async{...}> sau atribute phprel similare. Structura tag-ului este urmatoarea:

- <&async{(handler).(caller).(target).(page url).(processor)}>
- parmetrii corespund parametrilor functiei makeAsyncRequest, iar (page url) este eventualul url pentru o cerere cu pagina. Daca lipseste sau este sir vid, cererea va fi considerata standard.

Asa cum se observa, fiecare parametru este separat prin punct ".". In cazul in care unul dintre parametrii contine punct ".", va fi inclus intre paranteze rotunde "(", ")" pentru a nu crea ambiguitati. De exemplu, daca handler-ul se numeste "produse.imagini", caller-ul "id_produs" iar target-ul "imagini" se va scrie tag-ul: <&async{(produse.imagini).id_produs.imagini}>. Aceasta regula se pastreaza si in cazul atributelor speciale phprel care primesc ca valoare solicitari de construire a cererii asincrone, cum ar fi atributul "async", "*", etc.

Tag-ul special phprel poate fi folosit oriunde intr-o secventa de cod javascript:

- in interiorul unui atribut de tip eveniment, de exemplu: <a href="javascript: void(0);" onclick="<&async{...}>">click
- in interiorul unei functii javascript, de exemplu:

```
function incarca()
{
        ...
        <&async{...}>
        ...
}
```

- in interiorul oricarui alt bloc de cod javascript

Trimiterea unei cereri asincrone este realizata criptat, folosind o cheie partiala de criptare configurata in cfgs/core.settings.inc.php prin $Tcfg_asyncTransmitKey (true sau false). Algoritmul de criptare se aplica valorii parametrului "call" al unei cereri asincrone, si nu se aplica in cazul in care "call" este 0 sau o valoare echivalenta cu zero (evaluata in javascript). Criptarea sau absenta criptarii unei cereri asincrone determina si daca cererea va fi salvata si rezolvata din cache sau nu. Ca strategie, se recomanda folosirea parametrului "call" pe zero atunci cand cererea nu presupune transferul de date de la client la server, pentru a activa cache-ul. Eventualele informatii aditionale pot fi trimise in parametrul "target" printr-un cod conventional stabilit de dumneavoastra.

3.7.3. Incarcarea paginilor folosind cereri asincrone

Cererile asincrone cu solicitarea unei pagini permit folosirea unei metode incredibil de simpla de incarcare a paginilor prin A.J.A.X. In phprel, aveti la dispozitie o functie de incarcare a continutului paginilor folosind cereri asincrone, functie gandita special pentru a fi folosita in conjuctie cu tag-uri de globale speciale ce permit apelul direct al unei functii.

Aceasta functie este functia a ($link, $element = 'content', $additional) unde parametrul $link va specifica linkul care se incarca prin A.J.A.X., $element va specifica elementul de pagina care se incarca (implicit, elementul "content") iar $additional poate contine orice alte atribute html aditionale care se vor adauga nemodificate tag-ului rezultat. Functiei ii corespunde un handler in web/handlers/, getpagecontent.php, care va intoarce continutul solicitat.

In primul rand, trebuie stiut ca parametrul $element poate contine denumirea oricarui element de pagina, chiar si multiple denumiri separate prin virgula (pentru a incarca mai multe elemente de pagina simultan). Functia va trimite o cerere asincrona cu pagina (url-ul paginii fiind url-ul specificat prin $link) catre server, va astepta rezultatul si va actualiza continutul elementului de pagina respectiv. Actualizarea se face pe baza unui div care poate fi scris manual de dumneavoastra in sursele html pentru a contine exact elementul de pagina dorit, sau poate fi adaugat automat de nucleul phprel sa cuprinda intreg elementul dat. Pentru a adauga div-ul manual, adaugati in locatia respectiva un div cu id-ul de forma "async.(denumire element)". Este recomandat sa lasati asistentul sa adauge div-ul, daca in functie de cum ati scris elementele este posibila adaugarea unui tag "div" pe primul rand al elementului.

Functia va fi folosita ca inlocuitor al tag-ului html "a", dupa cum urmeaza:

```
<&a((link))>text link</a>
```

sau

```
<&a((link), (element), "(additional)">text link</a>
```

si va fi inlocuita cu un tag html "a" care va incarca elementul specificat printr-o metoda de tip A.J.A.X. daca acest lucru este permis de browser-ul utilizatorului, sau clasic daca nu. In plus, functia va "simti" daca pagina destinatie poate sau nu sa fie incarcata prin A.J.A.X. (nu vor putea fi incarcate prin A.J.A.X. paginile care contin formulare phprel, sau cod javascript, mai exact tag-uri <&js> sau <script> din cauza ca functiile javascript nou incarcate nu vor fi recunoscute de browser) si daca pagina nu poate fi incarcata printr-o cerere asincrona, va fi incarcata clasic direct.

Configurarea functiei a() se face din cfgs/advanced/core.settings.inc.php, prin urmatoarele variabile:

- $Tcfg_allow_alink_function – pe true, functia a() va fi definita si va putea fi folosita, pe false functia nu va fi definita
- $Tcfg_check_if_target_page_is_compatible – true daca pagina destinatie va fi verificata sa fie compatibila cu incarcarea prin A.J.A.X., false daca nu va fi verificata si va fi incarcata oricum
- $Tcfg_alink_loading – folosit pentru a specifica mesajul sau imaginea afisata in timpul incarcarii continutului
- $Tcfg_alink_delay_before_showloading – indica un interval de timp de asteptare inainte ca mesajul/ imaginea de incarcare sa fie afisata. Util pentru a intarzia foarte putin afisarea mesajului de incarcare pentru cei care au o conexiune foarte rapida pentru ca navigarea sa li se para fluenta, ca si cum nimic nu s-a incarcat, totul era deja pregatit. Timpul specificat este in milisecunde.
- $Tcfg_auto_add_wrapper_div – true sau false dupa cum nucleul phprel va adauga sau nu automat div-ul folosit pentru incarcare (cu id de forma "async.(denumire element)") in cazul in care lipseste

Incarcarea paginilor de continut prin metode de tip A.J.A.X. este acum simpla si naturala, ca si cum ati scrie un link clasic catre o alta pagina.

3.8. Cos de cumparaturi si preluarea comenzilor

Obiectivul phprel este sa va ajute in fiecare operatie pe care o aveti de efectuat pentru a crea si intretine o aplicatie web. Astfel, fiecare sectiune sau capitol din "istoria" unui proiect web beneficiaza de un sprijin din partea nucleului phprel, pentru ca experienta dumneavoastra atunci cand realizati aplicatia sa fie una cat mai placuta si mai multumitoare. De aceea, am adaugat in phprel si un ajutor care ar putea fi considerat mai specializat decat celelalte: un sistem de cos de cumparaturi si preluare a comenzilor online, care poate fi integrat usor in orice situatie, folosind orice tabel de produse si orice sistem de plata.

Operatiile efectuate de sistemul de cos de cumparaturi includ:

- crearea automata a unei sesiuni de cumparaturi in baza de date
- adaugarea automata a produselor in cos, pe baza id-ului de produs
- modificarea automata a cantitatii unui produs din cos
- stergerea automata a unui produs din cos
- login automat al utilizatorilor pe baza unui formular cu user si parola
- logout automat
- recuperarea unei sesiuni de cumparaturi anterioara
- crearea automata a unei noi comenzi in baza de date la completarea formularului de comanda
- adaugarea automata a detaliilor de livrare
- pastrarea automata a oricaror campuri necesare referitoare la produs, discount-uri, etc. in sesiunea de cumparaturi

Cosul de cumparaturi este interconectat cu asistentul phprel pentru a va pune la dispozitie o metoda simpla si performanta de a construi un magazin online complet.

3.8.1. Sesiunea de cumparaturi

Cosul de cumparaturi este retinut in totalitate in baza de date, atat pentru referinte ulterioare cat si pentru statistici si flexibilitate. O sesiune de cumparaturi consta dintr-un cod generat aleator si retinut in sesiunea php, cod caruia ii corespund in tabelul de sesiunii (denumit standard "cart_sessions") produsele din cos la momentul actual. Sesiunea de cumparaturi este unica si corespunde unui cos de cumparaturi cu produse, cantitati, preturi, si orice alte informatii trebuie retinute.

Nucleul phprel genereaza automat o noua sesiune de cumparaturi la adaugarea primului produs in cos. Alternativ, sesiunea poate fi creata printr-un apel manual la functia de creare a sesiunii, newCart(). Dupa crearea sesiunii (generarea codului unic de sesiune) orice produs adaugat in cos se adauga in tabela de sesiunii impreuna cu toate detaliile necesare si codul de sesiune corespunzator.

Produsele pot fi adaugate, sterse, cantitatea fiecaruia poate fi modificata iar sesiunea impreuna cu toate produsele pot fi "inregistrate" ca apartinand unui utilizator. Daca sesiunea este generata intr-un moment in care un utilizator este deja autentificat, produsele vor apartine inca de la adaugare utilizatorului respectiv. Altfel, daca un utilizator se logheaza ulterior, produsele din cos vor fi actualizate sa apartina

respectivului utilizator. La logout, produsele nu vor mai fii vizibile, deoarece nu mai apartin unui cos de cumparaturi public. Sesiunea de cumparaturi functioneaza cu sau fara utilizator autentificat, fiind posibila inclusiv efectuarea comenzilor fara autentificare (daca aplicatia va fi construita de asa maniera incat sa permita acest lucru).

Comportamentul sistemului de cos de cumparaturi poate fi configurat din cfgs/advanced/core.settings.inc.php, prin urmatoarele variabile:

- $Tshp_cartTransportFields – campuri din tabelul de produse, separate prin virgula si existente si in tabelul de sesiunii (veti crea manual aceste campuri in tabelul de sesiuni) care vor fi copiate dintr-un tabel in altul o data cu adaugarea unui produs. Evident, functiile de control la scrierea in baza de date din tabelul de sesiuni referitoare la aceste campuri vor functiona.
- $Tshp_sessionTable – denumirea tabelului de sesiunii de cumparaturi, implicit "cart_sessions". Daca schimbati denumirea acestui tabel, desi nu este recomandat, schimbati si denumirea descrierii corespunzatoare din web/rules/tables/.
- $Tshp_ordersTable – denumirea tabelului de comenzi, implicit "cart_orders". Daca schimbati denumirea acestui tabel, schimbati si denumirea descrierii corespunzatoare din web/rules/tables/.
- $Tshp_shipmentTable – denumirea tabelului de adrese de livrare, implicit "cart_orders_shipment". Daca schimbati denumirea acestui tabel, schimbati si denumirea descrierii corespunzatoare din web/rules/tables/.
- $Tshp_autoClearCartAfterCheckout – true daca trebuie golit cosul de cumparaturi dupa efectuarea comenzi, false daca trebuie sa ramana neschimbat.
- $Tshp_redirectAfterLogin – o adresa url personalizata spre care browser-ul va fi redirectionat dupa operatia automata de login. "false" daca browser-ul va fi redirectionat catre aceeasi pagina.
- $Tshp_redirectAfterLogout – o adresa url personalizata spre care browser-ul va fi redirectionat dupa operatia automata de logout, "false" daca nu trebuie efectuata o redirectionare.

3.8.2. Interconectarea cu tabelul de produse

Sistemul de cos de cumparaturi poate fi interconectat cu orice tabel de produse care are o cheie primara denumita "id" si o descriere activa in web/rules/tables/. Configurarea legaturii cu tabelul de produse se face din cfgs/advanced/core.settings.inc.php, prin urmatoarele variabile:

- $Tshp_productsTable – denumirea tabelului de produse
- $Tshp_cartFields['price'] – denumirea campului de pret din tabela de produse.

O data interconectat cu un tabel de produse, veti putea adauga produse in cos prin simpla specificare a id-ului lor. Un produs trebuie sa existe in tabelul de produse pentru a fi adaugat in cos.

3.8.3. Adaugare, modificare, stergere, total si alte operatiuni

Sistemul de cos de cumparaturi este corelat cu asistentul phprel pentru a oferi o metoda rapida si performanta de construire a unui magazin online. Astfel, pentru adaugarea unui produs este suficienta atasarea la url-ul responsabil pentru adaugare a unui tag special phprel de forma:

```
<&cart.add{(id produs)}>
```

de exemplu, pentru a face functional urmatorul link:

```
<a href="produse/phprel">adauga in cos</a>
```

este suficient sa adaugam la link tag-ul cu id-ul produsului "phprel", sa spunem "5":

```
<a href="produse/phprel/<&cart.add{5}>">adauga in cos</a>
```

Evident, datorita posibilitatii imbricarii tag-urilor phprel, id-ul poate fi o variabila de bucla sau o variabila globala:

```
<a href="produse/<&denumire>/<&cart.add{<&id>}>">adauga in cos</a>
```

Tag-ul va fi inlocuit cu un fragment de url care va indica nucleului phprel sa adauge in cosul de cumparaturi produsul cu id specificat. Similar, cantitatea corespunzatoare unui produs din cos poate fi marita folosind tag-ul special <&cart.more{(id produs)}> sau scazuta folosind <&cart.less{(id produs)}>. Pentru ca tag-ul de "more" sa functioneze, este necesar ca produsul sa se afle deja in cos, iar tag-ul de "less" va sterge un produs din cos daca micsorand cantitatea aceasta devine zero. Pentru a sterge un produs se poate folosi tag-ul <&cart.remove{(id produs)}>.

De asemenea, aveti la dispozitie doua tag-uri speciale care se inlocuiesc cu suma totala a produselor din cos respectiv numarul de produse: <&cart.total> si <&cart.items>.

In plus, pentru afisarea rapida a continutului cosului de cumparaturi, se poate scrie o bucla cu denumire speciala "cart.contents" care va fi inlocuita cu produsele din cosul de cumparaturi (liniile selectate din tabelul de sesiuni) impreuna cu liniile corespondente din tabelul de produse (selectate in subarray-ul "product" al fiecarei linii). Rezultatul va avea deci chei toate campurile tabelului de sesiuni plus o cheie "product" care va contine un array asociativ cu produsul asociat liniei din tabelul de sesiuni. Campurile standard ale tabelului de sesiuni sunt:

- id_user – id-ul utilizatorului de care apartine linia respectiva
- id_session – codul de sesiune generat
- id_product – id-ul produsului adaugat in cos
- ordered – zero daca nu a fost inca comandat, 1 daca a fost comandat
- clickorder – numarul de ordine al produsului in cos
- price – pretul produsului (corespunzator pretului din tabela de produse la care eventual s-au aplicat alte taxe/ costuri si s-a scazut discount-ul)
- discount – discount-ul aplicat produsului in contextul in care a fost adaugat, implicit 0, poate fi modificat prin functia de control
- taxes – alte taxe sau costuri aplicate, implicit 0, poate fi modificat prin functia de control
- quantity – cantitatea din cos
- totalline – totalul pe linia respectiva (pret * cantitate)
- date – data si ora in format unixtime la care s-a adaugat produsul

Alternativ, denumirea buclei "cart.contents" poate fi completata cu o cerere de selectare recursiva a datelor similara unei cereri getData recursiva, adaugand la denumirea buclei un asterix "*" urmat de un numar de nivele de recursivitate. In aceasta situatie rezultatul buclei nu va mai contine cheia speciala "product" deoarece chiar si un nivel de recursivitate va prinde si tabelul de produse. Asadar, pentru afisarea continutului cosului se poate folosi:

```
<&loop "cart.contents">
```

sau
<&loop "cart.contents*(numar nivele)">

Daca un numar de nivele nu este specificat, se va folosi nivelul standard de recursivitate indicat in cfgs/advanced/core.settings.inc.php.

Similar, daca un utilizator este autentificat, se poate folosi o bucla cu denumire speciala "cart.orders" sau "cart.orders*(nivel recursivitate)" pentru a afisa toate comenzile efectuate de utilizatorul curent. Rezultatul va contine toate campurile tabelului de comenzi.

In phprel, aveti la dispozitie suplimentar metode simple de inserare a datelor in tabelul de comenzi sau in tabelul de adrese de livrare. Ordinea standard recomandata, care functioneaza automat, este inserarea intai in tabelul de adrese de livrare, apoi in tabelul de comenzi, sau in ambele simultan. Pentru inserarea datelor in tabelul de adrese de livrare, construiti un formular in care denumirile campurilor sa coincida cu denumirile campurilor tabelului de adrese livrare, si adaugati suplimentar un camp hidden folosind tag-ul special <&frm.goShipment> similar tag-ului <&frm.goPost> folosit in general cu reguli mysql. Nu este necesar sa adaugati si <&frm.goPost>, nucleul phprel nu va folosi o regula mysql clasica pentru a insera datele. La trimiterea formularului, nucleul phprel va sesiza prezenta campului "goShipment" si va insera datele in tabelul de adrese de livrare, retinand totodata id-ul de inserare in sesiunea de cumparaturi pentru a putea fi atasat apoi comenzii.

Pentru trimiterea unei comenzi, folositi un tag similar <&frm.goOrder> impreuna cu un formular. Cand formularul este trimis, o linie reprezentand comanda va fi adaugata automat la tabelul de comenzi impreuna cu cateva campuri calculate automat de phprel, iar toate campurile vor fi trecute prin eventualele functii de control definite pentru tabelul de comenzi. Campurile ale caror valori vor fi calculate automat sunt:

- id_user – id-ul utilizatorului care a lansat comanda
- id_session – codul de sesiune de cumparaturi, pe baza caruia se face legatura cu tabelul de sesiuni de unde se pot prelua produsele
- date – data si ora in format unixtime la care s-a lansat comanda
- totalunits – suma totala a produselor comandate, fara alte costuri adaugate
- othercost – orice alte costuri care se aplica comenzii, initial 0, valoarea poate fi modificata prin functia de control aplicata campului
- shipment – valoarea cheltuielilor de transport, initial 0, poate fi modificata prin functia de control aplicata campului.
- payed – daca a fost platita sau nu comanda, initial 0, va fi schimbata manual de dumneavoastra prin cod in momentul in care se realizeaza plata.
- total – este totalul rezultat dupa adaugarea la "totalunits" a costurilor de livrare "shipment" si a altor costuri "othercosts" precum si dupa aplicarea eventualei functii de control a campului

Alternativ, o comanda se poate efectua accesand un link GET care contine un parametru special generat folosind tag-ul <&cart.order> similar ca functionalitate tag-urilor <&cart.add{...}>, <&cart.more{...}> etc. Acest tag va fi inlocuit cu un fragment de url continand un cod de identificare pe care nucleul phprel il va recunoaste si in urma caruia comanda va fi inregistrata. Acest tag ar putea fi de exemplu adaugat url-ului de la atributul "action" al formularului continand datele de livrare din ultimul pas al comenzii.

In plus, tag-urile de adaugare, crestere a cantitatii sau scadere a cantitatii admit si un al doilea parametru reprezentand cantitatea adaugata, respectiv cu care se creste sau scade cantitatea actuala.

Trebuie stiut ca toate aceste metode asistate au si un corespondent php, apelabil manual:

- pentru adaugarea unui produs in cos se poate folosi functia addToCart ($id_produs, $cantitate = 1)
- pentru cresterea cantitatii, moreToCart ($id_produs, $cantitate = 1)
- pentru scaderea cantitatii, lessFromCart ($id_produs, $cantitate = 1)
- pentru stergerea unui produs din cos, removeFromCart ($id_produs)
- pentru suma totala a produselor din cos, cartTotal()
- pentru numarul total de produse din cos, cartItems()
- pentru continutul cosului de cumparaturi sub forma de array de randuri, cartContents() sau cartContents ($nivel_recursivitate)
- pentru comenzile efectuate de utilizatorul curent sub forma de array de randuri, cartOrders() sau cartOrders ($nivel_recursivitate)

Pentru un exemplu de folosire a sistemului de cos de cumparaturi, recititi sectiunea 2. Elemente de baza, subsectiunea 2.5. Cos de cumparaturi si preluarea comenzilor.

3.8.4. Functii de control

Tabelele folosite pentru sistemul de cos de cumparaturi au si ele descrieri in web/rules/tables/ si fiecare camp al fiecarui tabel, fie ca este tabelul de utilizatori, de sesiuni, de comenzi sau de adrese de livrare, poate avea functii de control la scriere sau citire din baza de date.

Suplimentar fata de functiile de control clasice, sistemul de cos de cumparaturi dispune de urmatoarele functii, de obicei scrise in web/rules/tables/io.cart.sessions.php (descrierea de tabel a tabelului de sesiuni de cumparaturi):

- cart___add ($product, $fields, $values) - apelata la adaugarea unui produs in cos, $product fiind un array asociativ continand linia din tabelul de produse corespunzatoare produsului adaugat, $fields fiind un array simplu (chei numerice consecutive) continand campurile adaugate in tabelul de sesiuni cumparaturi, iar $values fiind un array simplu cu valorile campurilor respective. Functia va intoarce un array de valori, neschimbat sau modificat in functie de situatie, care vor fi valorile finale inserate in tabelul de sesiuni.
- cart___modify ($product, $fields, $values) - apelata la modificarea cantitatii unui produs din cos, parametrii fiind similari functiei "cart___add" dar $fields si $values vor contine doar campurile modificate din tabelul de sesiuni, mai exact "quantity" si "totalline"
- cart___remove ($product) - apelata la stergerea unui produs din cos (fie prin operatia de stergere directa, fie prin scaderea cantitatii la zero). $product va contine un array asociativ cu linia din tabelul de produse corespunzatoare produsului sters, iar functia va intoarce true daca operatia de stergere din cos este permisa, false altfel.
- cart___total ($total) - apelata pentru a rotunji totalul in momentul solicitarii lui fie prin tag-ul de <&cart.total> fie prin apel la cartTotal(). Totalul trebuie de obicei formatat intr-un anumit fel sau cel putin rotunjit la un anumit numar de zecimale, iar aceasta functie este pusa la dispozitie tocmai in acest scop. Intoarce un sir de caractere reprezentand totalul (dat prin parametrul $total) formatat corect pentru afisare.
- cart___validate ($row, $d) - apelata pentru a valida afisarea fiecarei linii la apelul functiei cartContents() sau la afisarea continutului folosind o bucla cu denumire speciala. Functia este utila pentru orice validare suplimentara inainte de afisarea produsului in cos (de exemplu,

atunci cand se revine la o sesiune de cumparaturi mai veche, anumite produse ar putea sa nu mai fie disponibile si trebuie scoase din cos). Intoarce true daca linia (data prin $row ca numar de la 0 la N-1 si $d ca array asociativ reprezentand concret linia) va fi afisata sau false altfel.

- cart___refresh ($contents) - apelata la autentificarea unui utilizator, primeste ca parametru continutul cosului de cumparaturi si va intoarce noul continut (sub forma unui array de randuri). Utila atunci cand anumiti utilizatori beneficiaza de discount-uri, de exemplu sau in orice situatie in care continutul/ preturile afisate in cos trebuie modificate in momentul autentificarii unui utilizator.
- cart___checkout ($fields, $values) - apelata in momentul in care se inregistreaza o noua comanda, primeste ca parametrii doua array-uri simple, $fields cu denumirile campurilor si $values cu valorile care se insereaza in tabelul de comenzi. Intoarce valorile finale dupa ce eventual au fost modificate (daca este cazul) sub forma unui array simplu.
- cart___done () - apelata in momentul in care comanda a fost inregistrata. Valoarea intoarsa nu conteaza. Se pot realiza actiuni suplimentare la inregistrarea unei comenzi.
- cart___login ($userid) - apelata imediat inainte de autentificarea unui utilizator (cu id-ul trimis ca parametru), intoarce true daca se permite operatia de autentificare sau false altfel.
- cart___logout () - apelata imediat inainte de operatia de logout a utilizatorului curent. Intoarce true daca operatia este permisa, false altfel.

Toate functiile de control (clasice sau proprii sistemului de cos de cumparaturi) pot accesa variabila globala $Tshp_currentCartOperation care va detalia in fiecare moment ce operatie asupra cosului de cumparaturi este in curs de desfasurare. Variabila este un array asociativ cu unele dintre cheile urmatoare:

- type – tipul operatiei curente, poate fi:
  - "add" – adaugare produs in cos
  - "modify" – modificarea cantitatii unui produs din cos
  - "remove" – stergerea unui produs din cos
  - "getPreviousShipment" – solicitarea ultimei adrese de livrare a utilizatorului curent
  - "cartContents" – solicitarea continutului cosului de cumparaturi
  - "cartOrders" – solicitarea comenzilor efectuate de utilizatorul curent
  - "addShipmentDetails" – adaugarea unei adrese de livrare
  - "checkout" – finalizarea unei comenzi
  - "processDirectives" – procesarea operatiei solicitate de catre nucleul phprel
- quantity – cantitatea produsului in cos
- product – array asociativ continand datele produsului (din tabelul de produse)
- addquantity – cantitatea adaugata printr-o operatie de modificare (valoare negativa pentru micsorarea cantitatii)
- previousquantity – cantitatea precedenta a produsului in cos, pentru operatia de stergere
- user – utilizatorul curent, in cazul solicitarii adresei de livrare precedente sau adaugarii unei adrese de livrare
- recursive – nivelul de recursivitate in cazul solicitarii continutului sau comenzilor precendente, false daca nu este o cerere recursiva.
- shipment – array asociativ continand adresa de livrare, in momentul finalizarii comenzii
- products – array de randuri continand produsele atasate comenzii, in momentul finalizarii comenzii

Pentru a citi continutul variabilei intr-o functie de control trebuie, bineinteles, sa declarati variabila ca globala.

3.8.5. Sistemul de login

Sistemul de cos de cumparaturi poate fi interconectat cu un tabel de utilizatori care in mod obligatoriu va contine campurile urmatoare:

- id – cheia primara numerica
- user – denumirea utilizatorului
- pass – parola ca hash md5
- confirmed – 1 sau 0 dupa cum utilizatorul a fost confirmat (de exemplu prin email) sau nu

Implicit, tabelul este denumit "front_users" si creat automat de interfata de management a aplicatiei – Web Assistant. O data tabelul stabilit, un utilizator va putea fi autentificat foarte usor folosind un formular cu doua campuri, "user" si "pass", impreuna cu un hidden special creat prin tag-ul <&frm.goLogin> similar hidden-ului <&frm.goPost> folosit in conjuctie cu regulile mysql. La trimiterea unui astfel de formular, nucleul phprel va sesiza prezenta parametrului "goLogin" in POST, va prelua parametrii "user" si "pass" si va verifica existenta utilizatorului in tabel.

In cazul in care utilizatorul sau parola sunt incorecte, nucleul phprel va crea o variabila globala $Tshp_cartLoginError cu valoare "1" (codul de eroare conventional pentru "utilizator / parola incorecte") si nu va autentifica utilizatorul. Altfel, daca valoarea campului "confirmed" corespunzatoare utilzatorului este echivalenta cu zero, nucleul phprel va crea variabila globala $Tshp_cartLoginError cu valoarea "2" (codul de eroare pentru "utilizator neconfirmat") si nu va autentifica utilizatorul. In final, daca atat combinatia utilizator / parola cat si "confirmed" au valori corespunzatoare, utilizatorul va fi autentificat iar id-ul sau va fi disponibil apeland functia getUser().

In plus, detaliile corespunzatoare utilizatorului autentificat (mai exact, linia corespunzatoare din tabelul de utilizatori) vor fi disponibile printr-un apel la getUserDetails(), sau un anumit camp al liniei de detalii va fi disponibil printr-un apel de forma getUserDetails ('(camp)').

Inainte de autentificarea utilizatorului se apeleaza suplimentar functia cart___login care decide daca autentificarea este permisa sau nu. Dupa autentificare, au loc urmatoarele operatiuni:

- toate produsele din cosul de cumparaturi sunt marcate ca apartinand utilzatorului curent
- este apelata functia cart___refresh pentru eventuala actualizare a cosului de cumparaturi in functie de utilizator
- daca exista date in sesiunea persistenta (pentru mai multe detalii, consultati sectiunea 5. Elemente avansate, subsectiunea 5.3. Memorarea variabilelor de la o sesiune la alta), acestea vor fi inregistrate ca apartinand utilizatorului curent
- daca nu exista date in sesiunea persistenta, vechea sesiune persistenta a utilizatorului va fi incarcata

Logout-ul se realizeaza printr-un parametru GET "logout" cu valoarea pozitiva. Inainte de logout se va apela functia cart___logout care va decide daca logout-ul este unul valid sau nu.

3.8.6. Adrese de livrare, costuri de transport, si adaugarea oricarui alt discount/ taxa/ precizare/ informatie

Asa cum am vazut deja, sistemul de cos de cumparaturi se ocupa si de adresele de livrare. Acestea sunt stocate intr-un tabel separat, denumit implicit "cart_orders_shipment" care are implicit campurile:

- id – cheia primara
- id_user – id-ul utilizatorului cu respectiva adresa de livrare
- id_order – id-ul comenzii corelate cu respectiva adresa de livrare

Celelalte campuri necesare, care sa contina concret adresa, vor fi adaugate de dumneavoastra in functie de particularitatile proiectului.

Pentru a adauga o adresa de livare am vazut ca trebuie doar sa construiti un formular care sa contina toate campurile necesare (respectiv acele campuri pe care dumneavoastra le adaugati si in tabel, cele doua campuri implicite referitoare la utilizator si comanda fiind completate automat de nucleul phprel) impreuna cu un camp "hidden" special <&frm.goShipment>. Adresa de livrare se va adauga inainte sau simultan cu solicitarea creerii comenzii.

Aveti de asemenea la dispozitie functia previousShipmentDetails() care va intoarce un array asociativ cu adresa de livrare anterioara a utilizatorului curent (pe care il veti putea folosi pentru a precompleta formularul, de exemplu).

Costurile de transport le veti putea adauga folosind functia de control la scrierea in baza de date atasata campului implicit "shipment" al tabelului de comenzi. Initial, campul va avea valoarea zero, dar aceasta valoare poate fi modificata. In functie veti dispune de toate informatiile necesare stabilirii costului de transport prin variabila $Tshp_currentCartOperation dar si prin functiile phprel referitoare la cosul de cumparaturi. De notat faptul ca nucleul phprel aduna automat valoarea costurilor de transport (intoarsa de functia de control) la totalul comenzii.

Similar, se poate adauga un discount produselor prin functia de control a pretului produsului din tabelul de sesiuni de cumparaturi, discount care poate fi precizat apoi in campul implicit "discount" al aceluiasi tabel. Pretul produsului insa va fi in mod obligatoriu modificat sa indice pretul final, nici o alta adunare sau scadere nefiind efectuata asupra pretului in mod direct de nucleul phprel.

Trebuie de asemenea stiut faptul ca orice alta informatie referitoare la produs sau comanda poate fi retinuta prin crearea unui camp destinat respectivei informatii in tabel si adaugarea unei functii de control. Suplimentar, campurile care influenteaza comanda din tabelul de produse si care ar putea fi modificate in timp pentru aceleasi produs, pot fi copiate in tabelul de sesiuni folosind variabila $Tshp_cartTransportFields.

Dispuneti asadar de un sistem de cos de cumparaturi flexibil, dinamic si usor de folosit, cu ajutorul caruia veti construi magazine online foarte rapid.

3.8.7. Recuperarea sesiunii si vizualizarea produselor sterse

O sesiune de cumparaturi precedenta poate fi recuperata si continuata printr-un apel la newCart ('(cod sesiune anterioara)'). Apelul ar putea fi realizat de exemplu in functia cart___login (la autentificarea unui utilizator) pentru a recupera sesiunea de cumparaturi anterioara a utlizatorului, sau prin specificarea de catre utilizator a unui cod pe pagina (corelat cu codul sesiunii de cumparaturi). Un apel de

recuperare a sesiunii va functiona doar daca sesiunea respectiva este publica (nu a apartinut unui utilizator autentificat) sau daca utilizatorul caruia apartine corespunde utilizatorului curent.

La stergerea unui produs din cos, produsul nu este fapt sters din tabel, doar din sesiunea curenta, fiind pastrat pentru eventuale statistici, strategii de marketing, etc. Pentru ca produsul sa fie scos din cos, codul sau de sesiune este prefixat de "REMOVED-". Astfel, pentru o sesiune cu codul "X" (fictiv, codul de sesiune este un cod alfa-numeric din 10 caractere) produsele sterse din cos vor fi vizibile prin selectarea produselor cu cod de sesiune "REMOVED-X".

3.8.8. Plata printr-un sistem de plata

Nucleul phprel nu va pune la dispozitie cod preconstruit pentru integrarea unui anume sistem de plata in magazinul dumneavoastra online. Cu toate acestea, folosind phprel va fi foarte usor sa redirectionati utilizatorul catre pagina sistemului de plata dupa incheierea comenzii, si sa preluati cu o pagina de "feedback" rezultatul platii pentru a marca ulterior campul "payed" din tabelul de comenzi cu valoarea "1" (cod conventional simbolizand faptul ca respectiva comanda a fost platita).

Ordinea naturala recomandata este: finalizarea comenzii prin sistemul de cos de cumparaturi, apoi redirectionarea catre sistemul de plata, in final procesarea raspunsului sistemului de plata (intr-o pagina de "feedback" separata solicitata de majoritatea sistemelor de plata).

3.9. Navigare

Nucleul phprel va pune la dispozitie doua metode simple de navigare pe site: linkuri de intoarcere utile in cazul redirectionarii formularelor sau butoanelor de "Inapoi" si un sistem de navigare pe baza de sesiune care va retine intreaga istorie a navigarii pe site.

3.9.1. Url de intoarcere

Un url de intoarcere reprezinta, dupa cum spune si denumirea, url-ul precedent la care va fi redirectionata pagina in cazul in care este necesara sau solicitata "intoarcerea" la pagina anterioara. Url-ului destinatie i se adauga din pagina curenta un parametru GET special care va contine url-ul curent ca url de intoarcere. Parametrul GET este denumit "nav:back" si va fi folosit automat pentru redirectionarea formularului, in cazul in care un formular trimis catre o pagina virtuala nu intruneste conditiile php de validare din rules/forms.process.php, sau va fi folosit pentru a construi linkuri sau butoane de "Inapoi".

Pentru a atasa acest parametru GET la un url, in template-ul care contine respectivul link se va adauga la atributul "href" corespunzator variabila globala speciala <&nav.generate>. Aceasta va fi inlocuita cu un sir de caractere de forma "&nav:back=(url de intoarcere)" unde (url de intoarcere) va fi url-ul curent codificat ca parametru GET.

Trebuie stiut ca un url de intoarcere va pastra atat linkul anterior exact cat si parametrii POST din momentul generarii, iar in urma redirectionarii catre un url de intoarcere, pagina va fi intoarsa la exact stadiul precedent atat din punct de vedere al solicitarii http cat si din punct de vedere al formularelor

(POST). Dupa urmarea linkului, nucleul phprel va sesiza prezenta linkului de intoarcere si il va folosi automat in anumite situatii.

Pentru folosirea manuala a url-ului de intoarcere, se poate folosi tag-ul special <&nav.back>. De exemplu, un link de "Inapoi" pentru o pagina care a fost incarcata cu url de intoarcere se va scrie:

```
<a href="<&nav.back>">Inapoi</a>
```

Se recomanda folosirea url-urilor de intoarcere atunci cand:

- pe pagina destinatie se creaza un link de "Inapoi" catre trebuie sa duca pe pagina anterioara exact in stadiul in care era la directionarea catre pagina destinatie (trebuie pastrate valorile formularelor, parametrii GET aditionali, etc.)
- pagina destinatie este un popup care contine o lista si un formular de editare sau doar un formular de editare
- pagina destinatie este o pagina virtuala
- pagina destinatie contine un formular de editare pe care se ajunge dintr-o lista sau pentru care nu se va scrie un link de redirectionare personalizat

In php, url-ul de intoarcere curent este disponibil in variabila globala $Tsys_navback.

Trebuie avuta grija in folosirea linkurilor de intoarcere impreuna cu cache-ul phprel.

3.9.2. Sesiune de navigare

Sesiunea de navigare permite retinerea fiecarui url vizitat de utilizator impreuna cu parametrii POST in sesiunea php, pentru crearea unui "navigator" pe site cu ajutorul caruia utilizatorul sa se poata intoarce dintr-o pagina in cea precedenta sau dintr-o pagina de detalii inapoi la lista de pe care a ajuns pe respectiva pagina, etc. In primul rand, navigarea cu sesiune este dezactivata standard, si poate fi activata manual la nevoie. Trebuie stiut ca folosirea sesiunii de navigare va invalida cache-ul asa cum este configurat el in mod obisnuit, si va necesita un anumit stil de programare a linkurilor de intoarcere folosind resurse de cache precum si o configurare aditionala a modului de cache. Asadar, se recomanda activarea sesiunii de navigare doar in cazul in care aceasta optiune este cu adevarat necesara si folositoare.

Sesiunea de navigare se poate activa din cfgs/advanced/optimizers.inc.php, prin setarea variabilei $Topt_sessionnavs pe "true". O data activata, sesiunea de navigare va inregistra pasii facuti de utilizator sub forma unui array de url-uri de intoarcere folosit ulterior pentru a genera linkuri de intoarcere la pagina precedenta sau la o pagina anterioara cu o anume semnificatie. Numarul maxim de url-uri retinute in sesiunea de navigare poate fi configurat din cfgs/advanced/core.settings.inc.php prin variabila $Tcfg_maxsessionnaventries.

Url-ul precedent poate fi obtinut printr-un apel la sessionNavBack ($jump = false, $returnpage = false). Apelata fara parametrii, functia va intoarce url-ul vizitat anterior url-ului curent. Apelata cu parametrul $jump "true", functia va intoarce url-ul precedent pentru care s-a incarcat o pagina php diferita de cea curenta. $jump este util pentru intoarcerea la pagina precedenta "reala", de exemplu in situatia in care pe pagina curenta se pot configura diverse filtre etc. care adauga noi parametrii GET sau POST la url modificand usor continutul paginii (dar ramanand totusi in aceeasi pagina). Parametrul $returnpage poate contine o denumire de pagina phprel si functia va intoarce cel mai "apropiat" url pe lantul de url-uri de intoarcere care a incarcat pagina solicitata. Util pentru intoarcerea la o lista sau o pagina centrala de pe care s-a plecat.

Pentru detalii privind folosirea sesiunii de navigare direct din template-uri cu ajutorul asistentului phprel, consultati subsectiunea urmatoare.

3.9.3. Variabile de navigare

Exista o serie de tag-uri speciale phprel care pot fi folosite in conjuctie cu metodele de navigare:

- <&nav.generate> - genereaza url-ul de intoarcere de forma "&nav:back=(url de intoarcere)"
- <&nav.back> - este inlocuit cu url-ul de intoarcere
- <&nav.here> - url-ul curent continand inclusiv datele din POST
- <&nav.session.back> - url-ul de intoarcere din sesiunea de navigare
- <&nav.session.jump> - url-ul de intoarcere din sesiunea de navigare, cu parametrul $jump = true
- <&nav.session.jumpto{(pagina)}> - url-ul de intoarcere din sesiunea de navigare cu parametrul $returnpage solicitat prin (pagina).

Aceste tag-uri sunt considerate tag-uri de variabila globala speciale si vor fi inlocuite cu valorile corespunzatoare de asistentul phprel.

3.10. Evenimente preincarcare si postincarcare a paginii

Am construit phprel avand in vedere trei concepte legate de scrierea codului: rapiditate, concizie si flexibilitate. De aceea, nu s-a facut nici un compromis privind posibilitatile pe care vi le oferim atunci cand construiti o aplicatie folosind phprel. Orice proces, orice structurare sau orice metoda introdusa de phprel care ar fi putut sa va dea senzatia ca vi se ingradeste libertatea a fost construit sa permita orice modificare, efectuata rapid si eficient. Nici succesiunea etapelor in incarcarea, procesarea si afisarea paginii nu a fost omisa. Pentru a avea un control complet asupra paginii, phprel va pune la dispozitie doua evenimente suplimentare care au loc inainte de incarcarea paginii (cu mult inainte, cand nu au fost incarcate inca majoritatea modulelor phprel) si dupa incarcarea paginii (la final, cand toate procesele automate sau asistate au avut loc si aveti posibilitatea verificarii si completarii tuturor operatiilor).

Evenimentul de preincarcare a paginii poate fi specificat prin crearea unui fisier cu denumirea de forma "preload.(denumire pagina).php" in web/pages/. In respectivul php se pot realiza toate operatiile suplimentare necesare inainte de incarcarea paginii, inclusiv dezactivarea anumitor module phprel care se incarca in general in aplicatia dezvoltata de dumneavoastra dar care nu sunt necesare pentru respectiva pagina.

Evenimentul "postincarcare" a paginii poate fi specificat prin crearea unui fisier cu denumirea de forma "postload.(denumire pagina).php" in web/pages/. Pentru rapiditate, se poate duplica si redenumi fisierul "postload.samplepage.php". Acest fisier, spre deosebire de cel de "preincarcare", va contine o functie cu denumirea "postload___transform___content" care primeste ca parametru intregul continut html al paginii si intoarce continutul final al paginii. Suplimentar, contine un array asociativ $Tcfg_postload_vars de forma variabila => valoare care contine valorile variabilelor "postincarcare" prezente in template-urile paginii. Mai exact, se pot scrie in template-urile html tag-uri de forma

<&postload.(denumire variabila "postincarcare")> care vor fi inlocuite automat in etapa finala conform specificatiilor array-ului asociativ.

Important: o variabila "postincarcare" nu va fi inlocuita cu un sir vid daca nu are corespondent in array-ul $Tcfg_postload_vars din fisierul php "postincarcare" atasat paginii ci va ramane in continutul final al paginii.

Se poate astfel modifica sau completa continutul paginii sau se pot realiza operatii suplimentare inainte sau dupa incarcarea paginii, in functie de context. Aceste metode suplimentare vin sa completeze flexibilitatea nucleului phprel si vor fi folosite in doar cateva cazuri.

3.11. Popup si template comun

Intr-o aplicatie web, anumite pagini vor fi incarcate in popup. De altfel, aveti la dispozitie o constanta $openPopup care se poate configura din web/settings/constants.inc.php si care poate fi folosita ca tag <&openPopup> pentru a deschide un popup cu javascript, precum si un atribut special "editinpopup" pentru liste care indica deschiderea linkului de editare in popup. Pentru paginile deschise in popup va trebui evident sa incarcati un alt "layout" (un alt header si un alt footer, cuprinse intr-un alt index). Aveti in web/templates/ fisierele necesare: index.popup.html, header.popup.html, footer.popup.html precum si php-urile atasate in web/pages/.

Pentru a specifica faptul ca o anumita pagina trebuie incarcata cu layout-ul de popup, editati in web/settings/pages.settings.inc.php array-ul asociativ $Tcfg_popup_pages (de forma pagina => true).

De asemenea, doua sau mai multe pagini pot avea acelasi template html (cu php diferit). O pagina care imprumuta template-ul altei pagini poate fi indicata prin array-ul asociativ $Tcfg_switchtemplate de forma (pagina) => (denumire template fara extensia ".html") din settings/pages.settings.inc.php.

3.12. Aplicatii third-party

Pentru comfortul dumneavoastra, cateva aplicatii third-party au fost integrate in phprel si pot fi de asemenea inlocuite cu orice alte aplicatii de acelasi tip (printr-o metoda de tip "plugin").

3.12.1. Calendare

Formularele includ un calendar dhtml util pentru selectarea datelor. Includerea calendarului pentru un anumit camp se realizeaza cu atributul phprel "calendar" atasat campului, fara valoare sau eventual cu valoarea "time" pentru afisarea unui calendar pe care se poate selecta si ora. Includerea propriu-zisa a calendarului se realizeaza din header.html, iar configurarile din web/templates/calendar.without.time.html respectiv calendar.with.time.html. Calendarul este "open-source", disponibil gratuit la www.dynarch.com in momentul publicarii versiunii curente de phprel.

De asemenea, calendarul poate fi activat prin click pe un "buton", un link sau o imagine cu id de forma "button_(id camp)". La click pe respectivul link sau imagine, calendarul va fi deschis pentru campul specificat. Atentie, id-ul linkului trebuie sa contina id-ului campului, nu denumirea sa.

Linia de includere a sursei javascript apartinand calendarului este inclusa intr-un loop cu ajutorul caruia nucleul phprel va exclude automat din pagina respectiva linie in cazul in care calendarul nu este folosit. Acelasi comportament il vor avea javascript-ul de cereri asincrone si javascript-ul de Jquery. Nucleul phprel va sesiza automat daca respectivele componente sunt folosite sau nu si excluse din pagina pentru a nu fi transferate la client fara motiv, ingreunand astfel inutil incarcarea paginii.

3.12.2. Editoare html

De asemenea, formularele includ si un editor html "wysiwyg" (What you see is what you get) folositor pentru aplicatii de tip CMS. Un camp este transformat in editor html prin atributul phprel "editor" sau "type" cu valoarea "editor". Nucleul phprel va include automat sursele javascript necesare si va configura automat editorul sa functioneze pentru respectivul camp. Editorul poate fi schimbat modificand web/templates/editor.include.html. De asemenea, clasa css necesara incarcarii editorului (care se aplica automat campurilor de tip "editor") poate fi schimbata daca e necesar din cfgs/advanced/core.settings.inc.php, prin $Tfrm_editorClass. Editorul html este "open-source", disponibil gratuit la http://tinymce.moxiecode.com/ in momentul publicarii versiunii curente de phprel.

3.12.3. Jquery

Pentru a facilita suplimentar scrierea codului javascript, pe langa ajutorul oferit de asistent, phprel include si jquery, o biblioteca des folosita pe web si disponibila gratuit la http://jquery.com/ in momentul publicarii versiunii curente de phprel. Nucleul phprel va sesiza automat daca ati folosit sau nu jquery in sursele html ale paginii curente si va exclude linia de includere a sursei in cazul in care nu ati folosit libraria. Alternativ, daca ati inclus un js care foloseste jquery (si nucleul nu va detecta automat folosirea librariei) se poate seta pe "true" variabila $Tsys_jqueryWasUsed in php-ul atasat paginii.

3.12.4. Mail

Nu in ultimul rand, pentru trimiterea email-urilor, nucleul phprel foloseste phpmailer, disponibil gratuit la http://phpmailer.codeworxtech.com/. Biblioteca este inclusa si folosita automat in momentul trimiterii unui email folosind functiile phprel de mail.

4. Securitate si performanta

In sectiunea precedenta, am explorat posibilitatile oferite de phprel pentru dezvoltarea rapida a aplicatiilor web. Am descoperit ca phprel ofera ajutor in majoritatea situatiilor tipice intalnite pe web prin metode intuitive si flexibile. In continuare, vom vedea cum procedeaza phprel pentru a pastra in siguranta aplicatia dumneavoastra si datele administrate prin intermediul aplicatiei precum si cum reuseste un

framework sa ramana performant cu toate ca trebuie sa faca foarte multe calcule pentru a oferi un sprijin real (masurabil prin scaderea timpului de dezvoltare).

4.1. Securitate

Cu phprel, nu primiti doar un framework si o serie de inovatii extraordinare care va vor ajuta sa dezvoltati rapid aplicatii web. Am fost atenti si la fiecare detaliu privind securitatea pentru a intari aplicatia dumneavoastra si pentru a va putea baza pe o aplicatie scrisa cu phprel. Vom descoperi in continuare principalele metode create pentru a ne asigura ca sunteti in siguranta.

4.1.1. Validarea url-urilor

Anumite url-uri folosite in aplicatiile construite cu phprel sunt cu risc ridicat, fie din motive ce tin de phprel (de exemplu, url-urile care folosesc parametrii GET "edit", "view" sau "del" sunt legati automat de o actiune si un tabel prin intermediul asistentului) fie din motive ce tin de aplicatie in sine (de exemplu, pe o anumita pagina, un parametru GET limiteaza accesul la anumite resurse care apartin unui utilizator sau unei situatii, dar utilizatorii rau-intentionati ar putea schimba valoarea parametrului pentru a accesa resurse sau informatii care nu le sunt destinate, eventual chiar ar putea crea pagube daca respectivii parametrii sunt folositi pentru stergere sau editare). In phprel, anumiti parametrii GET pot fi declarati cu "risc ridicat" si nu vor fi acceptati decat in prezenta unui cod de validare adaugat la url. Acest cod va fi adaugat automat de nucleul phprel, sau in cele cateva situatii in care nu este clar ce parametrii GET contine de fapt un url poate fi adaugat asistat.

Respectivii parametrii GET vor fi declarati in cfgs/advanced/core.settings.inc.php, prin array-ul asociativ $Tcfg_secure. Cheia va fi denumirea unui parametru, iar valoarea "true". Implicit, sunt trecuti parametrii GET "edit", "view", "del" dar se pot adauga oricati parametrii in functie de nevoile aplicatiei dumneavoastra.

De validarea url-urilor se ocupa un modul special din nucleul phprel, care dispune de asemenea de o serie de alte setari realizate in cfgs/advanced/core.settings.inc.php:

- $Tcfg_nodirectivesforfront – pe "true", indica nucleului phprel ca in varianta "front" parametrii standard "edit", "view", "del" nu se pot folosi. Daca nucleul va sesiza prezenta acestor parametrii la incarcarea paginii, ii va deseta imediat. Implicit "false".
- $Tcfg_useSession_withSecure – pe "true", foloseste si codul sesiunii in generarea codului de validare. Linkurile care au cod de validare atasat pot fi schimbate intre mai multi utilizatori sau intre mai multe browsere. Cu aceasta setare activata, linkurile vor avea un cod de validare generat special pentru sesiunea php curenta, si nu vor putea fi incarcate in alte browsere. Implicit "false".
- $Tcfg_autosecure – pe "true", linkurile din pagina ce urmeaza a fi afisata sunt cautate si acolo unde este necesar sunt adaugate automat codurile de validare.
- $Tcfg_manualsecure – pe "true", se permite folosirea tag-ului explicit de solicitare a adaugarii codului de validare, tag de forma <&sec{(link)}>. Implicit "false".

- $Tcfg_keysTable – numele tabelului intern folosit pentru gestionarea unor chei partiale care intervin in generarea codului de validare.
- $Tcfg_keyTimeout – intervalul de timp pentru care o cheie partiala pe baza caruia se genereaza codul de validare este valabila.
- $Tcfg_uniquekey – pe "true", cheia partiala generata va fi unica.

Generarea codului de validare a url-urilor tine cont de urmatorii factori:

- o cheie partiala generata aleator de nucleul phprel la un interval de timp prestabilit; codurile de validare vor fi valabile maxim respectivul interval de timp
- parametrii GET cu risc ridicat care intra in componenta url-ului cat si valorile exacte ale acestora; codurile de validare vor fi valabile pentru exact valorile date, nu se va putea schimba valoarea unui parametru GET
- link-ul in sine; codurile de validare vor fi valabile pentru exact url-ul dat, orice adaugare sau modificare adusa url-ului il va invalida
- optional, de codul de sesiune php; codurile de validare vor fi valabile doar pentru sesiunea curenta

Este imposibil de spus care va fi codul de validare pentru un url dat, chiar daca se cunosc parametrii care trebuie validati: codul va depinde de o cheie partiala generata aleator de nucleul phprel. Cheia aleatoare va fi unica si va expira la un interval fix de timp, un utilizator care candva a avut acces la o resursa dar i s-au revocat drepturile nu va mai putea vizualiza resursa accesand acelasi link.

Se pot inclusiv valida cererile asincrone, adaugand parametrul "async" la lista parametrilor cu risc, si specificand ca generarea codului de validare sa tina cont de sesiune.

Url-urile clasice (fara rewrite) vor fi descompuse si li se va adauga automat codul de validare in toate cazurile. Url-urile rescrise vor fi descompuse corect in masura in care parametrii vizati sunt dati explicit in url, sub forma "variabila/valoare", de exemplu "edit/5". Pentru url-uri rescrise care contin parametrii cu risc subintelesi (desi aceasta practica nu este recomandata) va trebui sa generati codul de validare printr-un apel la functia secureLink ($link, $directives, $values) unde $link este url-ul propriu-zis (care poate fi unicul parametru, caz in care parametrii GET cu risc vor fi descoperiti automat), $directives este un array simplu continand parametrii de risc inclusi in url, $values este un array simplu continand valorile respectivilor parametrii.

Ca alternativa la descoperirea automata a url-urilor in template-uri, acestea se pot indica folosind tag-ul special phprel <&sec{(link)}>, unde (link) este url-ul respectiv. De exemplu, pentru a indica manual solicitarea de validare a url-ului urmator:

```
<a href="produs/edit/5">editeaza</a>
```

se poate scrie:

```
<a href="<&sec{produs/edit/5}>">editeaza</a>
```

Nu se justifica indicarea manuala a url-urilor decat in cazul in care descoperirea automata dureaza mai mult timp decat este convenabil pentru standardul de performanta al aplicatiei dezvoltate.

In cazul in care url-ul curent contine parametrii cu risc ridicat si codul de validare lipseste sau nu este corect, parametrii cu risc vor fi desetati din GET. In plus, se va apela functia de control "securelinks___failed" cu parametru un array simplu continand parametrii cu risc implicati (daca functia exista). Respectiva functie de control poate fi folosita pentru a inregistra tentativele de acces fraudulos.

4.1.2. Revalidarea formularelor

Asa cum am vazut, validarea formularelor se realizeaza prin simpla adaugare a atributului "*" pe campurile necesare. Formularele se pot verifica complet in browser-ul clientului, fara reincarcarea paginii, folosind validari javascript sau prin cereri asincrone. La trimiterea datelor, se presupune ca acestea sunt corecte si sunt inserate direct in tabele. Pentru majoritatea aplicatiilor aceasta validare javascript este suficienta si nu ridica probleme.

In schimb, daca din motive de securitate considerati ca trebuie sa revalidati campurile in php, inainte de introducerea in baza de date, aceasta revalidare poate fi realizata automat de nucleul phprel fara nici o linie de cod scrisa de dumneavoastra. Utilizand sesiunea php si informatiile referitoare la validarea javascript, nucleul phprel va putea revalida campurile in php.

Aceasta functionalitate este implicit dezactivata, si poate fi activata daca este necesara din cfgs/advanced/optimizers.inc.php, prin setarea variabilei $Topt_phpenhancedforms pe "true". Optiunea permite atat revalidarea formularelor in php, cat si confirmarea faptului ca un formular a fost trimis din pagina site-ului si nu folosind metode alternative/ frauduloase de trimitere a unui POST. Suplimentar, aceasta optiune asigura compatibilitatea validarilor efectuate prin cereri asincrone cu browsere care nu permit astfel de cereri de tip A.J.A.X. Asa cum am vazut, operatiile efectuate prin metode de tip A.J.A.X. automat de catre nucleul phprel sunt realizate transparent prin metode clasice in cazul in care browser-ul nu permite realizarea cererilor asincrone. Folosind "php enhanced forms", aceasta caracteristica se pastreaza si pentru validarea cu A.J.A.X. Formularul va fi trimis prin POST, validat automat de phprel, si mesajele vor fi afisate prin reincarcarea paginii.

Un formular trimis fraudulos va atrage dupa sine apelarea functiei de control "secureforms___failed" (fara parametrii), daca functia exista, si incetarea executiei. In cazul unei erori de validare, se va reincarca pagina precedenta completandu-se formularul cu valorile date si div-ul de erori cu erorile descoperite.

Trebuie stiut ca formularele revalidate in php folosesc sesiunea pentru a retine schema de validare si ar putea, in anumite cazuri (foarte rar), genera inconcordante. In plus, in momentul activarii sesiunii necesare revalidarii, cache-ul devine indisponibil si paginile se vor incarca mai greu.

4.1.3. Criptarea cererilor asincrone javascript + xml (A.J.A.X.)

O cerere asincrona este trimisa impreuna cu o informatie, denumita "call", care uneori poate fi sensibila din punct de vedere al securitatii. De exemplu, atunci cand verificati unicitatea unei adrese de email sau a unui nume de utilizator, sau atunci cand trimiteti un formular intreg catre server, datele nu trebuie sa fie disponibile nodurilor de pe traseu. Pentru a va oferi siguranta in transmiterea informatiilor, valoarea parametrului "call" a cererilor asincrone este criptata in javascript inainte de lansarea cererii si decriptata pe server in php. Criptarea este realizata pe baza unei chei dependenta de sesiunea php si se poate activa sau dezactiva din cfgs/advanced/core.settings.inc.php prin variabila $Tcfg_asyncTransmitKey. Astfel, cheia va fi valabila doar pentru sesiunea curenta si datele nu vor mai putea fi decriptate ulterior de cineva care a interceptat pachetul in lipsa cheii.

4.1.4. Prevenirea preluarii sesiunii

Nucleul phprel pune la dispozitie un mecanism de stocare a datelor in sesiune care previne atacurile de tip "session hijacking" in care o persoana neautorizata preia sesiunea unui utilizator pentru a accesa resurse la care in mod normal nu are acces. Astfel sesiunea este codificata si accesata folosind o cheie generata pe baza identitatii de retea a utilizatorului care a initiat sesiunea si orice citire a datelor din sesiune se face pe baza respectivei chei. Presupunand ca o alta persoana ar incerca si reusi sa preia sesiunea curenta, cheia folosita pentru a citi sesiunea nu ar corespunde din cauza identitatii de retea diferite si datele concrete din sesiune ar fi inaccesibile. Pentru a stoca o informatie in sesiunea phprel, folositi functia toSes ('(denumire variabila)', (date)). Similar, pentru a citi o informatie din sesiunea phprel, folositi functia fromSes ('(denumire variabila)'). Datele sunt in continuare tinute in sesiunea php, dar ca subchei ale unei chei generate. Cheia va fi generata la fiecare citire si datele vor fi extrase din respectiva cheie, in cazul in care cheia generata este diferita (sesiunea a fost preluata de o alta persoana) datele nefiind disponibile.

Toate informatiile stocate in sesiune de nucleul phprel folosesc automat un principiu similar de stocare pentru a asigura siguranta informatiilor. Diferenta consta in faptul ca datele sunt stocate intr-o subcheie de sistem ("system") a sesiunii php, apoi intr-o subcheie generata pe baza identitatii, pentru a nu se confunda cu cele stocate de dumneavoastra.

4.1.5. Escape automat al query-urilor

Toate query-urile execute automat de phprel sunt asigurate impotriva atacurilor de tip "sql injection" prin "escaping". Nucleul phprel va formata corect fiecare query pentru ca sirurile de caractere exterioare sa fie tratate ca siruri de caractere si nici o comanda/ fragment de comanda sql sa nu se poata strecura in query-ul principal. Aceasta masura este luata pentru a proteja query-urile scrise de asistentul phprel sau de nucleul phprel in general, dar dumneavoastra va trebui sa realizati "escaping"-ul atunci cand introduceti intr-un query conditii de tip "where", "order by" sau "group by" chiar si folosind metode puse la dispozitie de nucleul phprel, pentru ca in aceste situatii nucleul phprel nu va stii ce date au provenit din variabile. Este de asenemea indicat sa verificati faptul ca optiunea "magic quotes" din unele versiuni mai vechi de php este dezactivata sau sa o dezactivati din ".htaccess" sau din "php.ini" daca acest lucru este posibil.

4.1.6. Escape manual al query-urilor

Pentru query-urile sau fragmentele de query-uri scrise manual si care trebuie protejate prin "escaping" impotriva atacurilor de tip "sql injection", aveti la dispozitie doua metode: prima, sa folositi functia phprel qescape ($variabila) in locul folosirii directe a unei variabile GET sau POST sau a unei variabile provenite din GET sau POST. De exemplu, apelul urmator getData:

```
$produse = getData('produse.search', "id_categorie='".$_POST['idcat']."'");
```

ar putea fi scris:

```
$produse = getData('produse.search', "id_categorie='".qescape($_POST['idcat'])."'");
```

Alternativ, se poate folosi o a doua metoda, in care indicati nucleului phprel ce variabile trebuie sa fie trecute prin functia qescape direct in sirul de caractere, folosind caracterul "&" precedat de un apostrof "'" si urmat de numele unei variabile globale cu valoare diferita de "null" si care nu va contine caracterul apostrof:

```
$produse = getData('produse.search', "id_categorie='&_POST[idcat]'");
```

4.1.7. Restrictionarea accesului folosind grupuri de utilizatori

In phprel, pentru aplicatiile de tip "backend" care necesita un grad ridicat de securitate, se pot configura politici de acces la pagini si liniile tabelelor pe baza grupurilor de utilizatori. Utilizatorii aplicatiei fac parte din unul sau mai multe grupuri, si fiecare grup are acces de vizualizare, modificare si/ sau stergere pe anumite pagini. La accesarea unei pagini, daca utilizatorul nu apartine unui grup care are drept de vizualizare pe pagina respectiva, browser-ul va fi redirectionat catre o pagina de eroare. Daca utilizatorul are drept de vizualizare dar nu si de modificare, eventualele formulare de editare din pagina vor fi convertite in pagini de vizualizare sau nu vor fi disponibile. Mai mult, toate linkurile de editare vor fi sterse din pagina. Similar, in cazul in care utilizatorul nu detine drept de stergere, linkurile de stergere din pagina nu vor fi disponibile. Nucleul phprel va modifica dinamic continutul paginii pentru a exclude linkurile catre pagini pe care utilizatorul nu are drepturi, linkuri de stergere sau de editare indisponibile, etc. fara nici o interventie din partea dumneavoastra. Suplimentar, anumite zone care nu trebuie sa apara in lipsa unui anumit drept pot fi manual incluse intr-un tag "if" cu o conditie in functie de drepturi.

Mai mult decat atat, se poate restrictiona accesul la liniile anumitor tabele. Un utilizator apartinand mai multor grupuri va putea fi restrictionat in a vizualiza si/ sau opera modificari doar asupra liniilor proprii (create de el) sau asupra liniilor apartinand grupurilor sale.

Drepturile date pe pagina si pe liniile tabelelor vor fi impuse atat la nivel de continut html (vizual) cat si la nivel de query-uri, astfel incat chiar daca un utilizator reuseste sa acceseze zone restrictionate folosind url-uri directe, si reuseste sa treaca de securitatea standard a paginii care va verifica daca utilizatorul are drept de vizualizare/ modificare/ stergere asupra paginii, eventualele query-uri pe care reuseste sa le execute nu vor intoarce nici un rezultat si nu vor produce nici o modificare.

Pentru mai multe detalii, consultati sectiunea 5. Elemente avansate, subsectiunea 5.4. Grupuri de utilizatori.

4.2. Performanta

Am vazut in sectiunile precedente ca phprel este un framework unic, care intr-adevar realizeaza foarte multe operatii automat (fara sprijin din partea dumneavoastra) sau asistat (cu indicatii din partea dumneavoastra) in scopul accelerarii procesului de dezvoltare web. Un framework tipic are un rol in primul rand structural in crearea aplicatiei: anumite componente au un anumit loc al lor si framework-ul se ocupa doar de interconectarea tuturor fragmentelor pe baza unei scheme bine organizate si bine definite. Suplimentar, anumite framework-uri ofera si biblioteci de functii sau clase care pot fi folosite de dumneavoastra in diverse scopuri. Functionarea acestor framework-uri nu implica decat indeplinirea unui

rol structural in care diversele componente scrise de dumneavoastra sunt interconectate si un rol de biblioteca, de cod inactiv care este sau nu folosit. Acest lucru nu inseamna un efort foarte mare de calcul iar consumul de resurse este adesea datorat dimensiunii bibliotecilor incarcate (fiecare biblioteca incarcata, indiferent daca este folosita sau nu, va folosi memorie pentru definitiile de clase, functii, etc. precum si timp de procesare pentru incarcare – acest lucru este valabil in special in cazul limbajelor de "scripting" care interpreteaza sursele la fiecare includere).

In cazul nucleului phprel insa, lucrurile stau un pic diferit. Aceleasi doua roluri sunt indeplinite si de phprel, dar acestea sunt doar roluri secundare. Rolul principal este acela de sprijin activ: phprel literalmente analizeaza sursele, cautand indicatii, indicii, solicitari si legaturi pentru a realiza cat mai multe operatii care va sunt de folos, pe care le-ati solicitat sau ati dat de inteles ca aveti de gand sa le realizati. Acest rol este unul serios, care implica foarte multe calcule, foarte multi algoritmi si, in cele din urma, timp de executie si resurse consumate. Cand vorbim despre versiunile initiale de test ale nucleului phprel, premergatoare versiunii finale, trebuie sa ne imaginam un framework cu un concept nou, dezvoltat pentru piata de site-uri web medii sau mici, cu un numar mediu de accese si un trafic mediu realizat lunar: foarte multe site-uri, daca nu chiar 80%, intra in aceasta categorie. In acest sector, scopul este dezvoltarea intr-un timp cat mai mic, mentenanta extrem de usoara si modificari ulterioare realizate intr-o clipita. Timpul de procesare si resursele nu sunt o problema, avand in vedere tehnica de calcul actuala si segmentul de piata: mediu sau mic, cu un numar rezonabil de accese. Calculele foarte numeroase realizate suplimentar de phprel nu sunt un impediment, site-urile raspund aproape instantaneu la numarul de solicitari carora se adreseaza, in schimb sunt realizate de minim trei ori mai repede. Aceasta a fost strategia initiala de lansare a produsului, dar am realizat in stadiile ulterioare de dezvoltare si testare ca cei din segmentul "high-end" se vor simti marginalizati si am decis ca mediul web merita mai mult: merita un framework capabil sa ofere impresionantul sprijin oferit de phprel la performante demne de sectorul de piata "superior", al aplicatiilor web cu trafic intens si un numar foarte mare de accese. Am pornit astfel intr-o noua misiune, aceea de a optimiza nucleul phprel in asa masura incat sa se ridice la inaltimea acestei provocari. Ceea ce a rezultat este versiunea actuala de phprel, care ofera functionalitati unice si inovative la un standard de performanta bun. Este adevarat, sunt framework-uri pe piata sau in medii private care ofera o performanta "foarte buna", dar pentru sarcinile indeplinite de phprel, o performanta "buna" este acceptabila. Si acest lucru in situatia in care exista si framework-uri care nu dau randamentul oferit de phprel si au o performanta comparativa "slaba". Am recurs atat la optimizari clasice ale codului, prin condensari de cod, eliminarea oricaror redundante, impunerea unui nivel maxim de complexitate al algoritmilor, renuntarea la tiparele "comfortabile" de programare php in favoarea unor abordari mai rudimentare dar mai eficiente ca timp, cat si la inovatii precum "incarcarea dinamica" a componentelor nucleului si o tehnologie de caching adaptiv, ajutat de concepte noi precum "cacheSentinels".

Ce am obtinut? Iata cateva cifre pentru ca dumneavoastra sa fiti in masura sa evaluati daca produsul se potriveste necesitatilor diferitelor proiecte la care veti lucra:

- pe un server linux slab ca performanta (Athlon 64 x2, 4000+, 2.1 GHz, 2 GB RAM, hdd SATA II 7200 rpm), pentru o complexitate de la mica la mare, si excluzand pagini foarte simple sau foarte complexe:
    - timp de procesare fara cache: 0.06 – 0.25 sec
    - memorie utilizata: 1.5 – 5 MB
    - timp de procesare cu cache: 0.010 – 0.015 sec
    - memorie utilizata cu cache: 300 – 400 KB
    - medie rezultata pentru 10 incarcari cu rata succes cache 90%: 0.015 – 0.0385 sec

    - medie rezultata pentru 100 incarcari cu rata de succes cache 50%: 0.035 – 0.132 sec
- pe un server de inalta performanta, performantele vor fi evident mai bune.

Cifrele afisate sunt obtinute prin masurarea timpului necesar executiei script-ului, masurand un timp initial pe prima linie din "index.php" si un timp final pe ultima linie inainte de afisarea continutului, rezultand o diferenta folosita de pagina pentru a incarca nucleul phprel si a procesa continutul paginii.

Aceste cifre sunt orientative si nu constituie un "best case scenario", timpii pot varia inclusiv intr-un sens pozitiv (spre mai mic) in functie de tipul de server, incarcarea serverului, tipul de aplicatie, complexitatea paginii. De asemenea, rata de succes a cache-ului va fi in multe situatii mult mai mare de 50% pe termen lung, iar pe termen scurt (10 afisari) ar putea fi 100%.

Nu in ultimul rand, pe paginile cu trafic foarte mare, penalizarea unui "cache miss" este redusa prin tehnologia inovativa "cacheSentinels".

4.2.1. Caching adaptiv si tehnologia cacheSentinels

Nucleul phprel include un modul de caching construit pentru a accelera afisarea paginilor. Paginile vor fi servite aproape instantaneu din cache, imediat ce respectiva pagina a fost solicitata prima data pentru a genera propriu-zis continutul. Cache-ul phprel este stocat in baza de date, spre deosebire de alte cache-uri care folosesc fisiere. Acest lucru permite o gestiune mai complexa dar si un timp de acces mai bun, in special in situatia in care serverul mysql este unul dedicat. De asemenea, spre deosebire de alte cache-uri, phprel nu foloseste aceasta tehnica pentru a retine rezultate partiale ci pentru a retine continutul propriu-zis al paginilor sau elementelor de pagina. Cache-ul nu este implementat pentru realizarea operatiilor interne ale framework-ului, ci pentru afisarea concreta a continutului. Rezultatul este unul vizibil: o pagina afisata din cache este disponibila in cateva milisecunde.

Configurarea cache-ului se realizeaza din cfgs/advanced/engine.settings.inc.php prin urmatoarele variabile:

- $Tche_cache_tables_prefix – prefixul tabelelor interne phprel folosite pentru cache. Cache-ul va folosi o serie de tabele, iar denumirile lor vor fi prefixate de cuvantul dat prin aceasta variabila. Implicit "cache".
- $Tche_cacheifsession – pe "true", permite afisarea din cache a paginilor si salvarea in cache a paginilor chiar si atunci cand exista informatii in sesiunea php. Implicit "false". Urmatoarele variabile din sesiune sunt exceptate de la regula:
    - id-ul utilizatorului autentificat pe "backend" - acest id va fi adaugat ca subfolder al cache-ului, fiecare utilizator de pe backend avand propriile sale versiuni de pagini in cache
    - limba curenta in cazul aplicatiilor multilanguage – aceasta va fi adaugata ca subfolder al cache-ului, fiecare limba avand propriile sale versiuni de pagini in cache
- $Tche_validSessionKeys – array asociativ de forma (variabila) => true, care permite anumitor chei din sesiune sa fie acceptate in procesul de caching. Paginile vor putea fi afisate din cache si salvate in cache chiar daca respectiva cheie data prin (variabila) este cheie in sesiunea obisnuita php, subsesiunea cu cheie de identitate phprel, sau subsesiunea "sistem" cu cheie de identitate. Implicit "false".

- $Tche_cacheAllowPartialHits – pe "true", permite afisarea din cache a elementelor secundare de pagina conform configurarilor din interfata de management a aplicatiei – Web Assistant. Implicit "false".
- $Tche_cacheSentinels – pe "true", activeaza optiunea "cacheSentinels".
- $Tche_cache_global_settings – setarile globale ale cache-ului, care se aplica pe toate paginile care nu au alte setari particulare specificate prin intermediul interfetei de management – Web Assitant. Astfel:
    - type – "static" sau "dynamic", dupa cum cache-ul este static sau dinamic. Tipurile de cache static includ: regenerarea cache-ului dupa un anumit numar de vizite sau dupa un anumit timp. Tipurile de cache dinamic implica regenerarea cache-ului in functie de schimbarile produse in baza de date sau la un numar maxim de afisari.
    - refresh – modul de regenerare a cache-ului pentru o anumita pagina. In cazul in care tipul de cache este static, poate fi unul dintre urmatoarele moduri:
        - "x-visits" – regenerarea are loc dupa un numar de X vizite, x dat prin "f_k"
        - "x-minutes" – regenerarea are loc dupa un numar de X minute, x dat prin "f_k"
        - "x-hours" – regenerarea are loc dupa un numar de X ore
        - "x-days" – regenerarea are loc dupa un numar de X zile

      In cazul in care tipul de cache este dinamic, poate fi unul dintre urmatoarele moduri:
        - "max-f-visits-with-backend" – regenerarea are loc dupa maxim F vizite, F dat prin "f_k", sau atunci cand prin intermediul aplicatiei in sine (intr-o fereastra de editare din oricare componenta a aplicatiei) s-a produs o schimbare in baza de date care ar putea afecta continutul
        - "max-f-visits-with-db" – regenerarea are loc dupa maxim F vizite, F dat prin "f_k", sau atunci cand in baza de date s-a produs o schimbare care afecteaza continutul. Baza de date este monitorizata activ si doar acele valori care schimba pagina vor semnala necesitatea regenerarii cache-ului.
        - "closed-circuit-with-db-only" – regenerarea are loc doar atunci cand s-a produs o schimbare in baza de date care afecteaza continutul. Baza de date este monitorizata activ.
        - "closed-circuit-with-backend-only" – regenerarea are loc doar atunci cand din aplicatia curenta s-a schimbat ceva in baza de date care ar putea afecta continutul paginii.
    - f_k – in functie de modul de regenerare a cache-ului, specifica intervalul fix/ maxim la care cache-ul se regenereaza (in numar de afisari, minute, ore sau zile)
    - f_up – in cazul selectarii unui tip dinamic cu mod de regenerare "max-f", cache-ul este unul adaptiv: in functie de esecul sau succesul precedent, noul F maxim dupa care are loc afisarea este recalculat. In cazul unui succes (considerat daca F a fost atins fara nici o schimbare detectata in baza de date), noul F va fi mai mare cu "f_up" procente. Implicit "30".
    - f_down – in cazul unui esec (considerat pentru cache de tip adaptiv, cand F nu a fost atins inainte ca o schimbare sa fie sesizata in baza de date), noul F va fi un procent din vechiul F, procent dat prin "f_down". Asadar "f_up" reprezinta "cu cate procente creste F" si "f_down" reprezinta "la cate procente se micsoreaza F". Setarea implicita, "auto",

lasa nucleul phprel sa decida dinamic valoarea in functie de momentul cand s-a sesizat schimbarea.

- $Tche_monitor_only – pe "true", trece cache-ul intr-un mod semiactiv in care doar se monitorizeaza baza de date, nu se realizeaza salvari si afisari din cache. Util atunci cand doriti sa dezactivati cache-ul pentru o anumita sectiune a aplicatiei dumneavoastra, dar alte sectiuni folosesc caching in modul "cu backend" si necesita monitorizarea bazei de date pentru a detecta schimbari provenite din toate sectiunile aplicatiei.

Trebuie stiut faptul ca pentru varianta "back" a nucleului phprel, cache-ul este intotdeauna setat in modul "closed-circuit-with-db-only". Setarile complete sunt valabile pentru varianta "front".

Anumite tipuri de cache sunt folositoare in anumite situatii. Astfel, in functie de flexibilitate (probabilitatea de a detecta o schimbare si de a afisa mereu continut actualizat) si de performanta diferitele moduri de regenerare, denumite in continuare "tipuri de cache", se claseaza astfel:

- flexibilitate:
  1. max-f-visits-with-db
  2. max-f-visits-with-backend
  3. closed-circuit-with-db-only
  4. closed-circuit-with-backend-only
  5. x-visits
  6. x-minutes
  7. x-hours
  8. x-days
- performanta:
  1. x-visits / x-minutes / x-hours / x-days
  2. closed-circuit-with-backend-only / max-f-visits-with-backend
  3. closed-circuit-with-db-only / max-f-visits-with-db-only

Este usor de vazut ca cel mai indicat tip de cache din punct de vedere al raportului flexibilitate/ performanta este "max-f-visits-with-backend", cache-ul adaptiv cu monitorizarea schimbarilor efectuate prin intermediul phprel. Pentru maximum de flexibilitate, se poate alege "max-f-visits-with-db".

Exista diferite strategii de alegere a cache-ului, in functie de situatie:

- pentru aplicatiile live, daca se alege intre monitorizarea activa sau "circumstantiala" a bazei de date, se va prefera a doua metoda, deci tipurile de cache "with-backend".
- pentru aplicatiile in curs de dezvoltare, se va alege monitorizarea activa a bazei de date
- pentru aplicatiile cu baze de date foarte mari (din punct de vedere al numarului de inregistrari), se va evita folosirea tipurilor de cache cu monitorizare activa a bazei de date
- pentru site-uri al caror continut se modifica foarte rar sau este in general static, se va prefera un tip de cache static in functie de situatie.
- pentru site-uri al caror modul multilanguage este activat, se va evita folosirea tipurilor de cache cu monitorizare activa a bazei de date

Indiferent de tipul de cache ales, veti avea disponibile date privind performanta in interfata de management a aplicatiei – Web Assistant – sectiunea "cache", subsectiunea "performanta".

Important: nu dezactivati cache-ul unei sectiuni care nu foloseste cache, daca alte sectiuni folosesc caching cu monitorizarea schimbarilor efectuate prin intermediul phprel. In schimb, pastrati activat cache-ul ($Topt_cache "true") dar doar in modul de monitorizare ($Tche_monitor_only "true").

Dupa ce o pagina a fost generata si continutul a fost salvat in cache, urmatorii factori vor conduce la regenerarea paginii:

- existenta unei chei in sesiune care nu este considerata "valida" (prin $Tche_validSessionKeys) atunci cand pentru pagina respectiva (prin setarile individuale pe pagina sau setarile globale) nu este permis caching-ul in timpul existentei unor date in sesiune (prin $Tche_cacheifsession)
- existenta unor date in POST
- aplicatia este in modul de dezvoltare ("devel")
- schimbarea sau prima folosire a setarilor globale de cache
- fisierele care construiesc pagina curenta au fost modificate din punct de vedere al continutului
- contextul "natural" (dat de tipul de cache) necesita regenerarea cache-ului
- continutul stocat in cache este un sir vid

Anumiti factori vor impiedica salvarea continutului in cache, paginile respective fiind generate la fiecare afisare:

- existenta unor date in POST
- afisarea paginii in modul de dezvoltare ("devel")
- existenta unei cereri asincrone catre handler-ul special "formpost"
- existenta unei cereri asincrone cu parametrul "call" criptat
- solicitarea unei modificari catre sistemul de cos de cumparaturi, sau existenta unui parametru GET "op" (asociat conventional sistemului de cos de cumparaturi)

Important: Se pot salva in cache si rezultatele unei cereri asincrone javascript + xml (A.J.A.X.), pentru a asigura un raspuns mai rapid la cerere, insa trebuie dezactivata transmisia criptata a cererilor. Alternativ, criptarea poate fi implicit dezactivata, si activata prin setarea cheii partiale doar pe paginile care transmit date importante prin A.J.A.X. Cererile asincrone cu call zero "0", de exemplu cele de continut, nu vor fi tranmise criptat.

Pentru a pastra o anumita flexibilitate a paginilor chiar si atunci cand sunt afisate din cache, exista tag-uri speciale phprel care pot fi inlocuite folosind "resurse" de cache: astfel, un tag de forma <&cache.(denumire resursa)> va fi inlocuit atat in cazul generarii paginii cat si in cazul afisarii din cache cu rezultatul unei functii speciale denumita "resursa de cache", declarabila in web/settings/cache.resorces.php. Functia va purta denumirea "cache___(denumire resursa)" si nu va avea parametrii, intorcand sirul de caractere care trebuie inlocuit in cazul intalnirii unui tag de forma <&cache.(denumire resursa)>. Pentru ca functia sa fie luata in considerare, trebuie declarata in array-ul asociativ $Tche_register_resource (de forma denumire resursa => true) din acelasi fisier de configurare. Continutul paginii va fi salvat in cache cu tag-urile neinlocuite; acestea vor putea fi inlocuite de fiecare data cand este afisata pagina, fie ca este afisata din cache fie ca este generata. Metoda permite salvarea in cache a unei pagini care are anumite date "dinamice", care se actualizeaza foarte des, sau imprevizibil, de exemplu data de azi, un curs valutar, etc.

O pagina afisata din cache va fi servita fara incarcarea unei mari portiuni din nucleul phprel sau din celelalte fisiere de configurare. In ceea ce priveste configurarile, doar urmatoarele fisiere vor fi incluse:

- optimizers.inc.php
- smartlocate.inc.php
- engine.settings.inc.php
- config.inc.php

Dintre modulele care alcatuiesc nucleul phprel, doar modulul de cache va fi incarcat.

In scrierea codului php al unei resurse trebuie avut deci in vedere ca toate functiile disponibile in mod obisnuit in phprel nu vor fi disponibile in cazul resurselor. Codul trebuie sa fie unul rudimentar, sau trebuie sa apelati la un cache "de nivel inalt", care va incarca mai intai toate modulele phprel. Aceasta metoda va fi costisitoare ca timp, dar ar putea fi necesara in anumite situatii. Pentru activarea acestui mod de caching, in interiorul functiei resursei respective apelati functia highLevelCache().

De asemenea, trebuie stiut ca nici macar evenimentul de "preincarcare" al paginii nu va fi inclus, dar cel de "postincarcare" va fi. Similar resurselor, codul din postincarcare trebuie sa tina cont de faptul ca functiile phprel ar putea fi indisponibile.

Cache-ul va salva inclusiv raspunsurile la cererile asincrone si le va reda direct din baza de date la o eventuala noua cerere identica. O cerere http este considerata identica daca toti parametrii get sunt identici. Fiecare cerere http va fi salvata separat si redata in momentul in care exact acelasi url este accesat sau solicitat.

Din interfata de management a aplicatiei – Web Assistant – se pot stabili comportamente diferite pentru cererile http care apartin unei anumite pagini. Aceste setari vor avea prioritate in fata setarilor globale. In plus, se pot stabili anumite setari pentru elementele secundare de pagina (care vor avea efect doar daca $Tche_ cacheAllowPartialHits este "true"). In cazul in care continutul unei pagini salvat in cache a expirat, si pagina trebuie regenerata, elementele secundare cu setari speciale vor fi preluate direct din cache, economisindu-se timp (si, in general, solicitari adresate serverului mysql). Un element secundar poate avea o politica separata de caching dar doar folosind cache de tip static. Exista anumite situatii in care prin caching veti urmarii realizarea unui numar cat mai mic de query-uri pe baza de date. Un element de pagina care realizeaza mai multe query-uri dar care se actualizeaza rar, sa spunem o data la cateva zeci de minute sau o data pe zi, ar putea fi mai usor incarcat dintr-un cache static pentru a "economisi" solicitarile respective catre serverul mysql.

Pentru aplicatiile cu un trafic intens si un numar mare de accesari, phprel va ofera o metoda inovativa de incarcare, "cacheSentinels". Tehnologia functioneaza pentru accesele simultane, mai exact aproape simultane. Pe un site cu trafic intens, sau pe un site care publica anumite informatii care vor fi accesate intens intr-o anumita perioada, foarte multe cereri http vor veni aproape simultan. In situatia in care continutul din cache corespunzator paginii este valabil, toate cererile vor fi servite aproape instantaneu si nu va fi o problema. Dar in situatia in care tocmai atunci cache-ul trebuie regenerat, toate accesele simultane vor primi sarcina sa regenereze pagina si vor consuma resurse, pentru acelasi rezultat. Cu cacheSentinels, accesele simultane comunica intre ele si de indata ce primul a terminat de generat pagina, celelalte preiau direct continutul si abandoneaza sarcina pe care tocmai o efectuau, economisind o parte din resurse si o parte din timpul de procesare. Acest lucru este posibil datorita faptului ca accesele nu sunt chiar simultane, dar vor fi intotdeauna decalate cu cateva zecimi de secunda. Acele zecimi de secunda conteaza, pentru ca in acel timp vor fi servite alte cereri http, mai ales direct din cache (la un timp de cateva milisecunde). Daca generarea completa a paginii dureaza 0.25 sec, si primul acces are loc la T1 = 0, al doilea la T2 = 0.05 si al treilea la T3 = 0.15 sec, fara cacheSentinels timpul total pentru ca cele trei accese sa termine procesarea va fi TF = T3 + 0.25 = 0.4 sec. Cu cacheSentinels, procesele vor avea loc dupa cum urmeaza:

- la T1 = 0 se incepe generarea paginii pentru primul utilizator
- la T2 = 0.07 se incepe generarea pentru al doilea utilizator
- la T3 = 0.15 se incepe generarea pentru al treilea utilizator (in total 3 accese "simultane")

- la Ta = 0.25 se termina generarea paginii pentru primul utilizator, generarea pentru utilizatorii 2 si 3 mai continua un pic, pana cand cacheSentinels sincronizeaza informatiile intre accesele simultane
- la Tb = 0.3 sec se termina generarea paginii pentru al doilea utilizator, sincronizarea nu s-a efectuat la timp
- la TF = 0.31 sec cacheSentinels descopera faptul ca al doilea utilizator a terminat generarea si preia continutul asa cum a fost generat de acesta, incetand orice alte operatii si afisand rezultatul.

Pentru un numar mai mare de utilizatori, mai multi vor fi avantajati. Un alt scenariu ar putea fi urmatorul: generarea dureaza 0.25 sec, primul acces are loc la T1 = 0, al doilea la T2 = 0.17, al treilea la T3 = 0.18 sec. Fara cacheSentinels, timpul total ca cele trei accese sa se termine va fi TF = 0.43 sec. Cu cacheSentinels, procesele vor avea loc dupa cum urmeaza:

- la T1 = 0 se incepe generarea paginii pentru primul utilizator
- la T2 = 0.17 se incepe generarea pentru al doilea utilizator
- la T3 = 0.18 se incepe generarea pentru al treilea utilizator (in total 3 accese "simultane")
- la Ta = 0.25 se termina generarea paginii pentru primul utilizator, generarea pentru utilizatorii 2 si 3 mai continua un pic, pana cand cacheSentinels sincronizeaza informatiile intre accesele simultane
- la Tb = 0.26 sec cacheSentinels descopera faptul ca primul utilizator a terminat generarea si preia continutul asa cum a fost generat de acesta, incetand orice alte operatii si afisand rezultatul pentru al doilea utilizator.
- La TF = 0.27 sec cacheSentinels repeta procesul pentru utilizatorul cu numarul trei, incheind astfel generarea.

Mai mult, trebuie stiut ca cu cat numarul de accese simultane creste, cu atat mai probabil este sa se ocupe intreaga memorie ram si sa se treaca la folosirea memoriei virtuale cu o executie de cel putin 10 ori mai lenta. Cu cacheSentinels, se consuma mai putina memorie si se serverc mai repede paginile, astfel ca se micsoreaza numarul mediu de accese simultane si consumul mediu de resurse si se previne utilizarea completa a memoriei.

Nu in ultimul rand, trebuie observat ca pentru dezactivarea rapida a cache-ului pentru o anumita pagina, se poate include in template-ul paginii tag-ul special "<&cache off>" care nu va mai permite salvarea in cache a paginii. Daca pagina a fost deja salvata in cache, se impune si resetarea cache-ului pentru ca schimbarile sa aiba efect.

Important: orice redirectionare manuala a paginii intr-o aplicatie scrisa cu phprel in care cache-ul este activat trebuie urmata de o intrerupere a executiei (de exemplu folosind un apel la functia die). In caz contrar, pagina va fi salvata in cache si "redata", fara sa realizeze ulterior redirectionarea si, mai important, fara sa realizeze ulterior eventualele query-uri manuale executate.

4.2.2. Incarcarea dinamica a modulelor

Nucleul phprel este alcatuit din mai multe module, fiecare dintre ele continand functii ce realizeaza anumite procese sau pun la dispozitie anumite solutii. Incarcarea intregului nucleu phprel pentru fiecare pagina ar fi costisitoare ca timp si probabil niciodata nu ar fi folosit intreg nucleul pe o singura pagina. Anumite functionalitati oferite de anumite module vor fi folosite pe o pagina, in timp ce pe alta vor fi

folosite alte functionalitati. Pe de alta parte, daca nucleul nu este incarcat, un apel catre o anumita functie va necesita incarcarea in prealabil a modulului respectiv.

Solutia inovativa adoptata in phprel este incarcarea dinamica a modulelor: nucleul phprel inregistreaza in php functiile existente in diferitele module dar nu le incarca propriu-zis, asteptand un prim apel pentru a le incarca. In momentul apelarii functiei, nucleul phprel include automat fragmentul de modul (un modul este impartit in mai multe fragmente incarcabile dinamic) necesar si executa functia. Se asigura astfel balanta perfecta intre performanta (prin includerea doar acelor fragmente care sunt necesare) si disponibilitate (toate functiile phprel sunt disponibile fara nici un "include" scris de dumneavoastra).

Din aceasta cauza s-ar putea sa sesizati ca un prim apel la o functie este mult mai lent decat apelurile urmatoare. Nucleul phprel se auto-include in functie de caz, pe fragmente. Paginile se vor incarca foarte rapid, fara sa trebuiasca sa astepte (indelung) procesul de includere a tuturor modulelor.

4.2.3. Optimizarea nucleului prin excluderea componentelor nefolosite

In phprel, aveti flexibilitate si control complet asupra aplicatiei dumneavoastra. Daca exista componente sau functionalitati pe care nu vi le doriti si vreti sa economisiti timp de procesare folosit pentru includerea si/ sau verificarea necesitatii respectivelor functionalitati, modulele respective pot fi dezactivate si nu vor fi incarcate la afisarea unei pagini, castigandu-se astfel resurse. Configurarile se realizeaza din cfgs/advanced/optimizers.inc.php, prin urmatoarele variabile:

- $Topt_datatools – activeaza sau dezactiveaza modulul de acces la baza de date. Include descrierile de tabel si functiile de construire rapida a query-urilor.
  Dependinte:
  - $Topt_mysqldatalayer
  - $Topt_multilanguage
  - functia addToArray din modulul de functii extinse nu va functiona
- $Topt_formtools – activeaza sau dezactiveaza modulul de generare formulare.
  Dependinte:
  - subfunctiile getData care intorc optiuni (pentru select-box-uri) nu vor functiona
  - functia a() ar putea (rar) avea probleme in construirea tag-ului html "a"
  - asistentul nu va inlocui tag-urile de formular, formularele fiind indisponibile. In schimb, toate tag-urile <&frm.(...)> vor fi inlocuite cu un sir vid
- $Topt_listingtools – activeaza sau dezactiveaza modulul de generare liste.
  Dependinte:
  - asistentul nu va inlocui tag-urile de lista, aceastea fiind inlocuite cu un sir vid.
- $Topt_assistant – activeaza sau dezactiveaza modulul Assistant – asistentul phprel.
  Dependinte:
  - regulile mysql nu vor fi deduse/ intuite si construite automat
  - nici un tag nu va fi inlocuit automat, toate inlocuirile se vor face manual, inclusiv incluziunile de elemente secundare.
- $Topt_multilanguage – activeaza sau dezactiveaza modulul de suport multilingv. Implicit "false".
  Dependinte:
  - tag-urile de traducere <&lng{...}> nu vor functiona, si vor fi inlocuite cu siruri vide.

    - campurile multilingve de forma lng()_(denumire) nu vor functiona si nu vor fi folosite in accesul la baza de date, liste, formulare, etc. Prezenta lor intr-o descriere de tabel nu va fi semnalata ca eroare, dar query-urile vor fi eronate.
- $Topt_securelinks – activeaza sau dezactiveaza modulul de validare a url-urilor.
  Dependinte:
    - tag-urile de adaugare a codului de validare <&sec{...}> nu vor functiona si vor fi inlocuite cu siruri vide
- $Topt_secureaccess – activeaza sau dezactiveaza modulul de securitate pe baza grupurilor de utilizatori. Implicit "false". Are efect doar pe varianta "back".
- $Topt_urlrewrites – activeaza sau dezactiveaza modulul de rescriere a url-urilor. Are efect doar pe varianta "front".
- $Topt_asyncs – activeaza sau dezactiveaza modulul de cereri asincrone.
- $Topt_shoppingcart – activeaza sau dezactiveaza modulul de cos de cumparaturi. Are efect doar pe varianta "front".
- $Topt_misctools – activeaza sau dezactiveaza modulul de functii extinse.
- $Topt_querytools – activeaza sau dezactiveaza modulul de management al query-urilor.
  Dependinte:
    - $Topt_secureaccess
    - query-urile pe multiple baze de date nu vor functiona
    - tabelele virtuale nu vor functiona
    - "escaping"-ul manual folosind "&" nu va functiona
    - debugging-ul query-urilor nu va functiona
    - cache-ul cu monitorizare a bazei de date (activa sau "circumstantiala") nu va functiona
- $Topt_cache – activeaza sau dezactiveaza modulul de cache
- $Topt_mem – activeaza sau dezactiveaza modulul de sesiuni persistente
- $Topt_sessionnavs – activeaza sau dezactiveaza suportul pentru linkuri de navigare folosind sesiunea. Implicit "false".
- $Topt_urladjustments – activeaza sau dezactiveaza auto-completarea url-urilor (inclusiv cu modul de dezvoltare, subdirectorul "web/" si $websiteURL)
- $Topt_smartlocates – activeaza sau dezactiveaza functia de smartLocate pentru includerea fisierelor din diferite locatii
- $Topt_securerows – activeaza sau dezactiveaza securitatea la nivel de linii in tabel pentru modulul de securitate pe baza de grupuri de utilizatori
- $Topt_phpenhancedforms – activeaza sau dezactiveaza revalidarea formularelor in php
- $Topt_mysqldatalayer – activeaza sau dezactiveaza descrierile de tabel.
  Dependinte:
    - $Topt_shoppingcart
    - $Topt_securerows
    - asistentul nu va deduce/ intui regula mysql
    - asistentul nu va inlocui tag-uri de lista sau de formulare
    - getData nu va functiona
    - buclele cu denumiri de tabel nu vor fi inlocuite automat
    - regulile mysql nu vor functiona
    - revalidarea in php a formularelor nu va functiona

        - anumite functionalitati care depind in functie de context de descrierile de tabel nu vor functiona uneori
    - $Topt_htmlsourceoptimize – activeaza sau dezactiveaza optimizarea automata a sursei html. Implicit "false" pentru dezvoltarea aplicatiei (cand veti avea nevoie de o sursa html lizibila), recomandat sa fie "true" pentru site-uri "live".
    - $Topt_mailingtools – activeaza sau dezactiveaza functia phprel de trimitere mail (impreuna cu aplicatia third-party "phpmailer")
    - $Topt_jquery – activeaza sau dezactiveaza aplicatia "third-party" Jquery.
    - $Topt_calendars – activeaza sau dezactiveaza calendarele dhtml
    - $Topt_editors – activeaza sau dezactiveaza editoarele html

In plus, se poate optimiza fiecare pagina in parte, dezactivand diferitele optiuni in evenimentele de "preincarcare" a paginilor.

4.2.4. Optimizarea automata a sursei html

In timp generarea unei pagini este de ordinul zecimilor de secunda, sau chiar sutimilor, transferul paginii la client poate dura cateva secunde, in functie de tipul conexiunii. Nucleul phprel poate optimiza automat dimensiunea continutului html trimis browser-ului, excluzand spatiile libere, caracterele "sfarsit de linie" si grupand intr-un singur tag javascript-urile. Castigul, in functie de complexitatea paginii, poate fi intre 20-50%, ceea ce poate insemna un timp relativ mare (de ordinul secundelor) castigat, in functie de conexiune, sau cel putin de ordinul zecimilor, compensand generarea paginii.

Pentru a activa optiunea, configurati valoarea variabilei $Topt_htmlsourceoptimize pe "true" in cfgs/advanced/optimizers.inc.php. Se recomanda activarea optiunii pe site-urile "live" pentru o incarcare accelerata a paginilor.

5. Elemente avansate

Sectiunea curenta isi propune sa clarifice unele functionalitati ramase inca neexplicate, care vor fi sau nu utilizate in functie de complexitatea proiectului. "Avansate" nu inseamna ca vor fi dificil de folosit, dimpotriva, toate optiunile phprel sunt simple si naturale, ci mai degraba vor fi folosite doar uneori, in functie de situatie, fiind de aceea rezervate unui utilizator care este deja "avansat", fiindca a deprins celelalte functionalitati phprel.

5.1. Suport pentru multiple baze de date

Pentru proiectele mari, care folosesc simultan mai multe baze de date, sau impart aceeasi baza de date in mai multe parti pentru marirea performantei, phprel ofera suport transparent de lucru cu multiple baze de date mysql. Nucleul phprel decide automat pe care baza de date sa execute query-ul, in timp ce dumneavoastra va veti concentra pe problema in sine de rezolvat.

5.1.1. Configurare

Pentru a stabili conexiunile cu mai multe baze de date, adaugati datele de conectare pentru fiecare baza de date suplimentara celei principale in cfgs/advanced/more.dbs.inc.php, folosind array-ul asociativ $Tcfg_newdb:

- cheia va contine o denumire data bazei de date, valoarea va contine datele de conectare (host, utilizator, parola)

Nucleul phprel nu se conecteaza automat la bazele de date specificate in more.dbs.inc.php, conexiunea se va realiza in momentul executarii primului query destinat respectivei baze de date.

Baza de date de care apartine fiecare tabel se configureaza din descrierea respectivului tabel prin parametrul $database. In cazul in care tabelul apartine bazei de date principale, acest parametru va contine denumirea "primary", altfel va contine denumirea bazei de date asa cum apare ea in array-ul $Tcfg_newdb.

Trebuie stiut ca tabelele interne phprel, cu exceptia celor destinate cosului de cumparaturi (care au si descrieri in web/rules/tables/) nu pot fi mutate pe alta baza de date decat cea principala. In plus, doua tabele cu aceeasi denumire dar din baze de date diferite vor trebui sa aiba denumiri diferite ca descrieri de tabel (descrierile de tabel trebuie evident sa aiba denumiri unice). In acest caz, dati o alta denumire unuia dintre tabele si folositi optiunea de tabel virtual pentru a indica spre tabelul real.

5.1.2. Query-uri pe o singura baza de date

Atunci cand realizati un query prin intermediul unei functii phprel (de exemplu: getData, sau select pentru query-urile mai generale/ mai complicate) baza de date este dedusa automat in functie de tabelul din care se selecteaza date. Dumneavoastra doar scrieti apelul functiei, nucleul phprel va selecta automat baza de date necesare si va executa query-ul pe acea baza de date.

In situatia in care trebuie sa scrieti query-ul manual, functia run_query ($query, $database) va permite sa specificati denumirea unei baze de date din array-ul $Tcfg_newdb (pentru "primary", nu specificati parametrul $database).

5.1.3. Query-uri pe multiple baze de date simultan

Avand in vedere ca nucleul phprel va selecta automat baza de date in functiile de tabelele din query, ar putea exista situatii cand nu toate tabelele apartin aceleiasi baze de date. Nucleul phprel poate rezolva intr-o oarecare masura aceasta situatie, desi este recomandat ca toate tabelele dintr-un query sa apartina aceleiasi baze de date.

In primul rand, pentru a folosi corect query-uri pe multiple baze de date, asigurati-va ca ele nu sunt destinate unor liste (in cazul in care sunt, folositi functiile de transformare si selectati datele din mai multe query-uri). In al doilea rand, asigurati-va ca folositi fie functia getData fie combinatia fetch(select()) pentru a obtine rezultatele.

Un query care necesita simultan mai multe baze de date va fi rezolvat automat de nucleul phprel, dar cu anumite restrictii:

- va exista un tabel sau un grup de tabele principale, apartinand bazei de date a primului tabel, pentru care se va executa primul query si la care se vor "lipi" celelalte informatii folosind query-uri suplimentare
- conditiile din "where" trebuie sa contina join-uri implicite (prin relatii de egalitate) intre tabele
- eventualele conditii de "order" sau "group" trebuie sa fie doar pe tabelul principal (considerat ca fiind primul tabel din lista)
- query-ul va fi realizat de asa maniera incat la fiecare pas datele din celelalte baze de date sa fie considerate ca date aditionale la nucleul de informatii deja selectat.

In general, orice query realizat cu getData pe multiple baze de date va functiona (in masura in care conditia privind "order" si "group" sunt respectate).

Aceasta facilitate este oferita pentru situatiile in care este absolut necesara selectarea datelor din multiple baze de date simultan, si are o serie de limitari, inclusiv legate de performanta. Suportul este binevenit in cazul query-urilor standard, intalnite in 80% din cazuri, dar query-urile complicate vor trebui executate pe aceeasi baza de date.

5.2. Suport pentru multiple limbi

Aplicatiile traduse in mai multe limbi pot fi scrise fara nici o grija privind problema "multilanguage": fara griji suplimentare legate de tabele, de template-uri multiple pentru diverse limbi sau de selectarea corecta a datelor in functie de limba. Nucleul phprel ofera un suport inovativ pentru aplicatiile multilingve, care va face dezvoltarea unei astfel de aplicatii la fel de usoara ca dezvoltarea unei aplicatii intr-o singura limba. Veti folosi un singur "set" de template-uri, care vor fi traduse de nucleul phprel conform traducerilor introduse in baza de date si veti scrie formulare, liste si query-uri ca si cum tabelele contin date pentru o singura limba.

5.2.1. Template-uri in multiple limbi

O prima problema care apare in realizarea site-urilor in multiple limbi este ca template-urile trebuie traduse pentru fiecare limba in parte. In general, acest lucru ar putea insemna pastrarea mai multor template-uri ale aceluiasi site, cate unul pentru fiecare pagina. Dar in phprel, aceasta problema este rezolvata transparent. Scrieti un singur template in care textele care pot aparea in mai multe limbi sunt incluse in tag-uri de forma <&lng{(text)}>. Modulul multilanguage va cauta automat aceste tag-uri si le va traduce in limba curenta pe baza tabelului de traduceri completat de dumneavoastra (sau de un alt utilizator). Respectivele texte vor fi considerate "etichete de traducere" si vor fi in prima faza adaugate in baza de date pentru traduceri, atat ca eticheta cat si ca traducere a etichetei pentru limba principala.

La procesarea unui tag de traducere, nucleul phprel va proceda dupa cum urmeaza:

- va cauta traducerea etichetei date pentru pagina curenta (fiecare eticheta are o "pagina destinatie" reprezentand pagina pentru care este tradusa eticheta intr-un anumit fel) in limba curenta selectata
- daca nu exista, va cauta traducerea etichetei date pentru orice pagina in limba curenta selectata

- daca nici aceasta traducere nu exista, va cauta traducerea etichetei date pentru pagina curenta in limba principala
- in final, daca nici chiar aceasta traducere nu exista, va cauta traducerea etichetei date pentru orice pagina in limba principala
- tag-ul impreuna cu eticheta vor fi inlocuite cu traducerea gasita.

Asadar, etichetele noi adaugate vor fi inlocuite cu acelasi text al etichetei, care va putea mai tarziu fi tradus. Ca regula, etichetele de regula reprezinta textele respective in limba de baza, atat pentru a face template-ul lizibil cat si pentru a economisi timpul necesarii adaugarii unei traduceri suplimentare, dar trebuie stiut faptul ca eticheta poate fi tradusa diferit chiar si pentru limba principala: eticheta nu reprezinta de fapt un text care va fi afisat in template, ci doar o "eticheta" care indica un alt text din tabelul de traduceri.

Pentru a traduce un text afisat din php (prin intermediul unei variabile, etc.) folositi functia phprel asemanatoare tag-ului, lng ($text). De asemenea, apelul la lng() fara o eticheta va intoarce limba curenta. La fel si tag-ul fara eticheta, va fi inlocuit cu limba curenta.

Pentru a configura modulul de traduceri, editati in cfgs/advanced/core.settings.inc.php urmatoarele variabile:

- $Tcfg_translationsTable – denumirea tabelului care contine traducerile. Implicit "translations".Tabelul de traduceri contine urmatoarele campuri:
  - id – cheia primara, calculata ca md5(strtolower((limba).'#'.(pagina destinatie).'#'.(eticheta))), unde (limba) este limba in care se traduce eticheta, (pagina destinatie) este pagina destinatie pentru care se traduce eticheta (sau caracterul asterix "*" pentru "orice pagina") iar (eticheta) este eticheta tradusa.
    Important: etichetele nu sunt case sensitive. O eticheta "AbCd" va fi aceeasi cu "abcd" si inlocuita cu aceeasi traducere.
  - lang – limba pentru care se traduce eticheta
  - targetpage – pagina destinatie pentru care se traduce eticheta sau "*" pentru "orice pagina"
  - sourcepage – pagina sursa in care eticheta a fost descoperita prima data
  - identifier – eticheta
  - translation – traducerea etichetei
  - lasthit – data si ora in format unixtime cand traducerea a fost folosita ultima oara
- $Tcfg_defaultLanguage – limba principala. Limbile vor fi codate de regula ca prescurtarea lor la doua litere, de exemplu "en", "ro", "fr".
- $Tcfg_allLanguages – toate limbile folosite in aplicatie, codate ca prescurtari de doua litere si separate prin virgula.
- $Tcfg_performCleanUp – pe "true", indica nucleului phprel ca poate sterge automat intrarile invechite din tabel
- $Tcfg_cleanUpTimeout – indica numarul de zile dupa care o eticheta nefolosita este considerata invechita.

Tag-ul <&lng{...}> poate fi imbricat in orice alt tag si va functiona corect. De exemplu, se poate scrie un camp de formular de forma:

- <&frm.framework_preferat{text; defaultvalue="<&lng{intotdeauna phprel}>">

Trebuie stiut de asemenea ca urmatoarele configurari din cfgs/advanced/core.settings.inc.php vor fi automat trecute prin functia de traducerere lng() si vor putea fi traduse in multiple limbi:

- $Tfrm_defaultEmptyOption
- $Tfrm_Vloading
- $Tfrm_Vempty
- $Tfrm_Vother
- $Tfrm_Vasync
- $Tfrm_Verrors
- $Tfrm_Vdiv
- $Tfrm_form_post_async_ok
- $Tlst_loadingmessage
- $Tcfg_alink_loading

Traducerea template-urilor se poate realiza: din interfata de management a aplicatiei, Web Assistant, dintr-un backend propriu in care ati importat pagina din Web Assistant, dintr-un backend propriu in care ati construit propriul script de traducere.

5.2.2. Tabele cu campuri in multiple limbi

O data template-urile traduse, o a doua problema o reprezinta baza de date: tabelele vor contine campuri care trebuie traduse in diverse limbi. Metoda aleasa pentru a rezolva aceasta problema in phprel este folosirea campurilor prefixate pentru a indica traducerea unui camp intr-o anumita limba. Sa presupunem ca avem un camp cu denumire "titlu" intr-o aplicatie care nu este multilanguage, acelasi camp intr-o aplicatie multilanguage va fi constituit din mai multe campuri, prefixate de limba, de exemplu: "en_titlu", "ro_titlu", "fr_titlu". Dar phprel merge un pas mai departe, si face transparenta folosirea acestor campuri. In realitate, toate cele trei campuri se vor referi la traduceri diferite ale aceluiasi camp "abstract", "titlu". In aplicatie, va veti referi la camp ca fiind "titlu", iar nucleul phprel va face automat "conversia" catre campul real in diferite situatii. De exemplu, intr-un formular de editare veti scrie tag-ul <&frm.titlu> si nucleul phprel va incara automat datele si va salva datele din campul corespunzator limbii curente, sa spunem "ro_titlu". Intr-o lista, veti scrie de asemenea la fields="titlu" si nucleul phprel va selecta automat datele din "ro_titlu". Chiar si in functiile de control la citirea din baza de date veti specifica respectivul camp ca fiind "titlu": "function std___output___tabel___titlu ($val, $d)".

Pentru a indica nucleului phprel ca un anumit camp este reprezentat printr-o serie de campuri prefixate de limba, in descrierile de tabel la parametrul $fields veti scrie "lng()_(denumire camp)" unde (denumire camp) este denumirea campului "abstract" iar "lng()" va tine locul limbii concrete. Acest tip de denumire poate fi folosit inclusiv in parametrii $namefield, $defaultorder si chiar $filterfield.

Important: functiile de control la scrierea in baza de date nu sunt comune pentru toate campurile prefixate, fiecare camp prefixat (de exemplu, "ro_titlu", "en_titlu") va avea propria functie de control: "function std___input___tabel___ro_titlu ($val)".

Evident, campurile vor fi create de dumneavoastra in tabelele corespunzatoare cu denumirile prefixate de limba, dar veti putea "adauga o limba" din Web Assistant (adauga noi campuri prefixate de o noua limba pe baza celor vechi). De asemenea, daca anumite campuri nu mai sunt folosite, va trebui sa le stergeti dumneavoastra.

Strategia pentru backend este sa realizati un backend care editeaza datele pentru o singura limba simultan, utilizatorul alegandu-si limba in orice moment dintr-un meniu. Astfel veti putea folosi aceleasi campuri "abstracte" si in backend.

Trebuie stiut ca o inserare realizata prin intermediul functiilor sau metodelor phprel va insera valoarea pentru limba curenta in toate campurile prefixate (pentru toate limbile). Asadar, din start orice camp este tradus in toate limbile cu aceeasi traducere, care vor putea fi modificate ulterior. Aceasta este o practica des folosita pe site-urile multilanguage pentru comfort si pentru siguranta ca exista continut chiar si atunci cand nu a fost tradus inca un text.

5.2.3. Setarea limbii curente

Limba curenta a site-ului poate fi setata prin trimiterea unui parametru GET "lang" cu codul prescurtat la doua caractere al limbii in care se schimba limba. Daca nici o limba nu a fost inca precizata, limba principala va fi setata ca limba curenta.

Pentru a seta manual limba, folositi functia setlng ($cod_prescurtat_limba). Pentru a obtine limba curenta, folositi un apel la lng() fara parametru.

In cazul in care frontend-ul dumneavoastra foloseste url rewrite, limba curenta va fi setata automat inainte de incarcarea si procesarea regulilor de rewrite. Ulterior, dupa incarcarea si procesarea regulilor, limba este din nou setata pentru a acoperi si modificarile cauzate de un eventual parametru "lang" rescris. Se recomanda asadar, pentru schimbarea imediata a limbii, trimiterea parametrului GET "lang" ca parametru clasic, "&lang=(cod prescurtat limba)", iar pentru schimbarea ulterioara rescrierii trimiterea sa ca parametru rescris de forma "lang/(cod prescurtat limba)".

5.2.4. Stergerea automata a intrarilor invechite

Nucleul phprel va sterge automat etichetele invechite din tabelul de traduceri in functie de data ultimei utilizari a unei traduceri a etichetei. Aceasta functie poate fi dezactivata din cfgs/advanced/core.settings.inc.php, prin variabila $Tcfg_performCleanUp.

5.3. Memorarea variabilelor de la o sesiune la alta (sesiune persistenta)

Vor exista situatii in care veti dori sa creati o sesiune persistenta, pe baza de cookie pentru acelasi browser sau pe baza de utilizatori pentru multiple browsere si/ sau calculatoare. Asa cum am afirmat deja, phprel va ajuta in orice situatie tipica de pe web, inclusiv in generarea si utilizarea unor astfel de sesiuni.

5.3.1. Setarea si folosirea unei variabile memorate

O sesiune persistenta este creata de nucleul phprel fie pe baza utilizatorului curent autentificat (utilizatorul "frontend"-ului autentificat prin sistemul de cos de cumparaturi) fie pe baza unui cod salvat intr-un cookie. Unui cod de autentificare sau unui utilizator ii vor corespunde in baza de date oricat de multe inregistrari cu valori ale unor variabile php. In cookie va fi salvat doar eventualul cod de autentificare, toate datele fiind stocate pe server.

O data sesiunea initializata cu un cod de autentificare sau un id de utilizator, vechile variabile vor putea fi extrase si altele noi vor putea fi salvate. Daca este o sesiune noua, se vor putea salva noi date. Astfel:

- la salvarea sau solicitarea unei informatii din sesiunea persistenta se intializeaza sesiunea dupa cum urmeaza:
  - daca exista utilizator autentificat, sesiunea se intializeaza folosind id-ul de utilizator
  - daca nu, si exista cod de autentificare in cookie, sesiunea se intializeaza folosind codul respectiv
  - daca nici codul nu exista, se genereaza un nou cod de autentificare care se salveaza intr-un cookie si se foloseste la initializarea unei noi sesiuni
- la solicitarea unei variabile, daca ea exista in sesiunea persistenta se va intoarce valoarea ei (fie ca este variabila simpla fie ca este array), altfel se va intoarce "null"
- la salvarea unei variabile, daca ea exista in sesiunea persistenta va fi suprascrisa, altfel va fi creata o noua inregistrare in tabel
- la autentificarea unui utilizator, dupa ce o sesiune a fost inceputa cu un cod de autentificare, daca utilizatorul avea la randul sau sesiune persistenta inregistrata, se revine la respectiva sesiune, altfel sesiunea curenta se inregistreaza pentru utilizatorul autentificat. Un cod de autentificare este valid atata vreme cat sesiunea spre care indica nu a fost inregistrata ca apartinand unui utilizator.

Pentru a salva o variabila in sesiunea persistenta se va folosi functia toMem ($denumire_variabila, $valoare). Pentru a solicita valoarea unei variabile din sesiunea persistenta se va folosi functia fromMem ($denumire_variabila). Pentru a forta incarcarea sesiunii persistente indicate de un anumit cod de autentificare, se poate apela functia initMem ($cod_autentificare).

Tabelele in care se retin sesiunile persistente vor fi prefixate de denumirea data in cfgs/advanced/core.settings.inc.php prin $Tmem_table.

5.3.2. Extinderea sesiunii pe mai multe browsere si/ sau calculatoare

Prin autentificarea unui utilizator de "frontend" (prin sistemul de cos de cumparaturi), sesiunea persistenta va fi inregistrata pentru id-ul respectivului utilizator si va fi accesibila de pe orice calculator sau orice alt browser, in momentul reautentificarii. Se poate folosi asadar sesiunea persistenta pentru a retine preferintele utilizatorilor pe site, sau pentru a crea obiecte de tip "gadget", etc.

5.3.3. Stergerea automata a variabilelor invechite

Dupa un anumit interval definit in cfgs/core.settings.inc.php prin $Tmem_timeoutForNotUsedVars, variabilele nefolosite vor fi sterse automat de nucleul phprel pentru a nu aglomera tabelele mysql. Cookie-ul cu codul de autentificare va fi de asemenea valabil pentru acelasi interval de timp (in zile).

5.4. Grupuri de utilizatori

Pentru aplicatiile de tip "backend" se pot configura politici de acces la resurse pe baza grupurilor de utilizatori. Un utilizator va apartine unuia sau mai multor grupuri si va acces la anumite pagini, iar pe acele pagini va avea anumite drepturi. De exemplu, anumite pagini vor fi complet restrictionate pentru grupurile din care face parte, altele vor fi disponibile doar pentru vizualizare, altele si pentru modificare. Configurarea este cat se poate de usoara iar stabilirea politicilor se face din interfata de management, Web Assistant. Alternativ, paginile necesare pot fi importate in aplicatia dumneavoastra sau se pot crea noi pagini care sa administreze drepturile.

5.4.1. Definirea utilizatorilor, grupurilor si politicilor de acces

Accesul pe pagini este restrictionat din punct de vedere a trei drepturi:

- de vizualizare
- de modificare (inserare sau actualizare)
- de stergere

Un anumit grup va contine oricati utilizatori si va avea un anumit drept pe o anumita pagina:

- fie de vizualizare
- fie de vizualizare si modificare
- fie de vizualizare, modificare si stergere

O pagina pentru care un grup nu are drepturi este restrictionata si nu poate fi incarcata de un utilizator apartinand respectivului grup.

Un utilizator poate apartine mai multor grupuri, dintre care unul va fi grupul principal, iar celelalte vor fi grupuri secundare. Drepturile pentru un utilizator se cumuleaza in functie de grupurile in care se afla, de exemplu daca grupul A are acces pe pagina X si grupul B are acces pe pagina Y, iar un utilizator apartine grupurilor A si B, atunci utilizatorul va avea acces pe paginile X si Y (in timp ce un utilizator care apartine doar grupului A nu va avea acces pe pagina Y si un utilizator care apartine doar grupului B nu va avea acces pe pagina X). De asemenea, daca utilizatorul apartine grupului A si B si utilizatorii grupului A au acces de vizualizare pe o pagina X iar utilizatorii grupului B au acces de vizualizare si modificare pe pagina X, atunci utilizatorul va avea acces de vizualizare si modificare (accesul mai mare).

Pentru gestionarea accesului la resurse, phprel foloseste urmatoarele tabele:

- back_users – tabelul de utilizatori continand campurile:
  - id – cheia primara numerica
  - user – denumirea utilizatorului
  - pass – parola, ca hash md5
  - email – un eventual email al utilizatorului
  - status – 1 sau 0 dupa cum contul este activ sau suspendat
- access_groups – tabelul de grupuri, cu urmatoarele campuri:
  - id – cheia primara numerica
  - name – denumirea grupului

    - admin – 0 sau 1 dupa cum grupul este grup de administratori sau nu. Un grup de administratori are drepturi depline in orice situatie (echivaleaza cu un grup care are stabilite drepturi maxime pentru orice resursa)
- access_pages – tabelul de pagini, cu urmatoarele campuri:
    - id – cheia primara numerica
    - name – denumirea paginii (egala cu variabila globala $page de pe respectiva pagina)
    - category – o eventuala categorie atribuita paginii (sub forma unui sir de caractere) pentru tinerea evidentei
    - description – o eventuala descriere realizata paginii pentru a fi clara identitatea ei (denumirea ar putea fi prescurtata sau ambigua)
      Important: nucleul phprel detecteaza automat noile pagini si le adauga la tabel, fie navigand pe ele fie descoperindu-le din linkurile existente pe o pagina incarcata
- access_tables – tabelul de tabele mysql restrictionate, care vor fi de asemenea descoperite automat de nucleul phprel o data cu prima folosire a lor intr-un query, avand urmatoarele campuri:
    - id – cheia primara numerica
    - name – denumirea tabelului in format mysql
    - description – o eventuala descriere a tabelului pentru a clarifica rolul lui
- access_user_groups – tabelul stabileste apartenenta utilizatorilor la grupuri. Campul "main" stabileste care este grupul principal pentru un anumit utilizator. In absenta unui grup principal explicit, primul grup in ordinea relevantei (grupurile "admin" au prioritate, apoi ordonate descrescator dupa denumire) va fi grupul principal. Contine urmatoarele campuri:
    - id – cheia primara numerica
    - id_user – id-ul utilizatorului din tabelul "back_users"
    - id_group – id-ul grupului din tabelul "access_groups"
    - main – 1 sau 0, reprezentand grupul principal al unui utilizator
- access_group_pages – tabelul stabileste politicile de acces ale unui grup la o pagina. Contine campurile:
    - id – cheia primara numerica
    - id_page – id-ul paginii din tabelul "access_pages"
    - id_group – id-ul grupului din tabelul "access_groups"
    - access – nivelul de acces:
        - 0 – reprezinta acces restrictionat si este echivalent cu inexistenta unei perechi (id_page, id_group) in tabel.
        - 1 – reprezinta acces de citire/ vizualizare a paginii
        - 5 – reprezinta acces de vizualizare si de modificare
        - 7 – reprezinta access de vizualizare, modificare si stergere
- access_user_tables – similar tabelului "access_group_pages" dar stabileste politicile de acces ale utilizatorilor apartinand unui anumit grup la liniile unui anumit tabel restrictionat. Contine campurile:
    - id – cheia primara numerica
    - id_table – id-ul tabelului din "access_tables"
    - id_group – id-ul grupului din "access_groups"
    - access – nivelul de acces:

- 0 – reprezinta acces restrictionat la toate liniile tabelului
- 1 – reprezinta acces la liniile proprii ale utilizatorului
- 5 – acces la liniile grupurilor din care face parte utilizatorul
- 7 – acces la toate liniile tabelului

Denumirile tabelelor pot fi configurate din cfgs/core.settings.inc.php dupa cum urmeaza:

- tabelul de utilizatori "backend" poate fi schimbat prin $Tcfg_backUsersTable
- prefixul tabelelor referitoare la acces poate fi schimbat prin $Tcfg_accessTablesPrefix

Trebuie stiut ca tabelul de utilizatori este folosit si atunci cand modulul de restrictionare a accesului pe baza grupurilor de utilizatori este dezactivat: accesul la o aplicatie "backend" este standard restrictionat pe baza utilizatorilor din acest tabel. Incercarea de a accesa orice pagina, inclusiv index-ul, va fi redirectionata catre pagina speciala "login" (cu corespondentele login.php si login.html) atata timp cat un utilizator valid nu este autentificat. La introducerea unui utilizator si a unei parole (intr-un formular de login identificat similar celui de pe "frontend" prin hidden-ul special "goLogin"), nucleul phprel va verifica datele de conectare in tabel. Daca utilizatorul sau parola sunt incorecte, nucleul phprel va crea variabila globala $Tshp_backLoginError cu valoarea "1" (codul de eroare conventional pentru "Utilizator si/ sau parola incorecte") si nu va autentifica utilizatorul, accesul la aplicatie fiind in continuare limitat la pagina "login". Daca utilizatorul si parola sunt corecte, dar campul de "status" este "0", nucleul phprel va crea variabila globala $Tshp_backLoginError cu valoarea "2" (codul de eroare conventional pentru "Utilizator suspendat") si nu va autentifica utilizatorul. Daca utilizatorul si parola sunt corecte iar "status" este un numar diferit de zero, se va apela suplimentar (daca exista) functia de control "back___login" cu parametru id-ul utilizatorului. Functia poate intoarce "true" sau "false" dupa cum login-ul este permis sau nu. In cazul in care login-ul este permis, utilizatorul va fi autentificat iar id-ul sau va fi disponibil printr-un apel la functia getBackUser(). In cazul in care login-ul nu este permis, variabila $Tshp_backLoginError va fi creata cu valoarea "true".

Pentru logout, se va trimite un parametru GET logout cu valoare diferita de zero si utilizatorul va fi intors in ecranul de "login".

In interfata de management al aplicatiei, Web Assistant, aveti la dispozitie o sectiune de stabilire a politicilor de acces, "acces aplicatie". Se pot crea grupuri din subsectiunea "grupuri" prin introducerea unei denumiri si selectarea statutului de "Admin". In mod similar se pot crea utilizatori iar prin click pe actiunea de "grupuri" a listei de utilizatori se poate stabili caror grupuri apartine fiecare utilizator. De asemenea, se pot edita paginile sau tabele adaugate in lista de phprel, si stabili politicile de acces la pagini si la tabele. Interfata este realizata intuitiv si contine toate explicatiile necesare. Sectiunea de "acces aplicatie" nu va aparea in meniul Web Assistant decat in momentul in care ati activat optiunea de restrictionare a accesului pe baza grupurilor de utilizatori (folosind $Topt_secureaccess din cfgs/advanced/optimizers.inc.php).

5.4.2. Restrictionarea accesului pe pagini

O data politicile stabilite, nucleul phprel va reactiona in urmatoarele situatii:

- este accesata o pagina pentru care utilizatorul nu are drept – nucleul phprel va redirectiona pagina catre "no.access" (cu corespondentele "no.access.php" si "no.access.html").
- este accesata o pagina pentru care utilizatorul are drept doar de vizualizare – nucleul phprel va interzice executia oricarui query de modificare sau stergere (query-urile, chiar daca sunt

solicitate de dumneavoastra in cod vor fi inlocuite cu query-uri "de forma": "SELECT '1'"), va cauta in continutul paginii eventualele formulare de modificare si va converti pagina in pagina de vizualizare

- este accesata o pagina pentru care utilizatorul are drept doar de vizualizare si modificare – nucleul phprel va interzice executia oricarui query de stergere
- este accesata orice pagina – nucleul phprel va cauta toate linkurile din pagina si va identifica pagina destinatie, linkurile de modificare (prin parametrul GET "edit" si orice parametru care incepe printr-un identificator dat in cfgs/advanced/core.settings.inc.php prin $Tcfg_additionalEditIdentifier) si cele de stergere (prin parametru GET "del", "(denumire lista):del" siu orice parametru care incepe printr-un identificator dat prin $Tcfg_additionalDeleteIdentifier), va verifica daca utilizatorul are respectivele drepturi in pagina destinatie, iar daca nu va:
    - sterge linkul respectiv din continutul html al paginii
    - sterge un eventual tag in care era continut linkul
    - sau sterge un eventual tag <td> in care era continut linkul
    - sau sterge un eventual tag <tr> in care exista un singur <td> care continea linkul

Astfel, resursele vor fi protejate concret impotriva acceselor neautorizate prin interzicerea rularii query-urilor (realizate prin functiile phprel sau de catre nucleul phprel, evident query-urile realizate direct cu mysql_query nu vor fi interceptate). Vor fi de asemenea protejate formal prin modificarea continutului paginii in functie de situatie:

- modificarea unui formular de editare in pagina de vizualizare, daca este cazul
- stergerea linkurilor catre pagini pe care utiliatorul nu are nici macar drept de vizualizare
- stergerea linkurilor catre pagini de editare in care utilizatorul nu are drept de modificare
- stergerea linkurilor catre pagini de stergere in care utilizatorul nu are drept de stergere

Suplimentar aveti la dispozitie functia checkCredentials ($pagina) care poate sa primeasca denumirea unei pagini ca parametru sau sa nu primeasca parametru caz in care va fi considerata pagina curenta, si va intoarce dreptul pe respectiva pagina (0, 1, 5 sau 7). Aceasta functie poate fi folosita si in tag-uri conditionale pentru a modifica pagina in functie de drepturi atunci cand nucleul phprel nu a descoperit automat o modificare ce trebuie realizata.

5.4.3. Restrictionarea accesului la informatiile din tabele

In cadrul modulului de securitate pe baza de grupuri de utilizatori se poate activa optiunea $Topt_securerows care va restrictiona accesul la informatiile din anumite tabele. In descrierile de tabel, activati optiunea $secure si creati campurile aditionale necesare:

- ownerid – id-ul utilizatorului care "detine" linia
- groupid – id-ul grupului principal al utilizatorului care "detine" linia
  De obicei, ownerid va corespunde utilizatorului care a creat linia si groupid va corespunde grupului sau principal
- createdon – data si ora in format unixtime la care a fost creata linia
- modifiedid – id-ul utilizatorului care a modificat ultima oara linia
- lastchange – data si ora in format unixtime la care a fost modificata ultima oara linia

O data $secure trecut pe "true", la primul query efectuat pe tabelul respectiv, nucleul phprel va sesiza modificarile si va adauga tabelul la lista tabelelor restrictionate. Un utilizator va avea acces la datele tabelului doar daca este dat explicit acces unuia dintre grupurile din care face parte. De asemenea, accesul la informatiile tabelului va putea fi limitat doar la liniile care ii apartin lui sau doar la liniile care apartin grupurilor din care face parte.

Query-urile de SELECT, INSERT, UPDATE si DELETE vor fi automat modificate pentru a reflecta drepturile unui utilizator. Trebuie stiut ca toate query-urile vor fi controlate si modificate in masura in care:

- sunt query-uri de tip SELECT si sunt efectuate de nucleul phprel sau de dumneavoastra prin functiile select() sau getData()
- sunt query-uri de tip INSERT, UPDATE sau DELETE si sunt efectuate de nucleul phprel sau de dumneavoastra prin functia writeData()

Query-urile efectuate direct folosind run_query nu vor fi modificate.

Dumneavoastra veti putea scrie cod phprel ca si cum utilizatorul ar avea acces la toate liniile sau ca si cum tabelul ar contine doar liniile la care utilizatorul are acces, iar nucleul phprel va modifica automat query-urile pentru ca aplicatia sa functioneze corect si informatiile sa fie protejate.

5.5. Functii extinse

Pentru a va oferi sprijin si in anumite situatii tipice de programare, aveti la dispozitie cateva functii ajutatoare folosite si de nucleul phprel. Functiile indeplinesc sarcini simple, dar pot fi uneori folosite pentru a scurta codul scris in mod obisnuit in rezolvarea unei "subprobleme".

5.5.1. Diverse functii folositoare

Exista o serie de functii folositoare incluse in biblioteca centrala phprel, care vor fi disponibile ori de cate ori se incarca nucleul phprel:

- (array simplu) strToArray ($sir_de_elemente_separate_prin_virgula) - converteste un sir intr-un array simplu de elemente pe baza virgulelor din sir
- (array asociativ) strToAssocArray ($sir_de_elemente) - converteste un sir intr-un array asociativ de forma element => true pe baza virgulelor din sir
- parseloop ('(denumire bucla)', $array_continut) - inlocuieste o bucla cu denumirea data in elementul de pagina "curent" (de exemplu, apelata in header.php, va inlocui bucla din header.html, etc.)
- parsevar ('(denumire variabila)', $valoare) - inlocuieste un tag de variabila cu denumirea data in elementul de pagina "curent"
- (array asociativ) getNavData() - pentru un url curent care are in componenta un url de intoarcere, intoarce un array asociativ cu cheile si valorile de POST din url-ul de intoarcere
- backLogin ($id) - logheaza fortat un utilizator dat prin id pe "backend"

Mai exista, de asemenea, cateva functii utile in diverse situatii care au fost deja prezentate:

- qescape

- defaultUpload
- defaultUploadImage
- addThumb
- clearThumb
- sessionNavBack
- toSes
- fromSes

Suplimentar, daca optiunea $Topt_misctools este activata, se pot folosi urmatoarele functii:

- (array simplu) toIds ($array_de_randuri) - intoarce un array simplu continand id-urile de pe fiecare rand din array-ul dat. Id-ul este fie cheia denumita "id" fie prima cheie din array-ul asociativ al unei linii.
- (array de randuri) splitForHtml ($continut_bucla, $coloane = 2, $tag_separator = '<tr>') - intoarce un array impartit pe coloane pentru afisarea in html (printr-un tag de bucla). Impartirea pe coloane se va realiza adaugand o noua cheie "split" care va fi folosita pentru a trece pe randul urmator. De exemplu:

```
<table>
<tr>
	<&loop "elemente">
	<td><&denumire></td>
	<&split>
	<&endloop>
</tr>
</table>
```

- (array de randuri) addToArray ($cheie, $array_de_randuri) - adauga o cheie la un array de randuri (pe fiecare rand in parte). Cheia este adaugata cu valoarea "1" dar functia va fi folosita in conjunctie cu o functie de control "alter___(cheie) ($val, $d)" pentru a adauga valorile concrete.
- (sir de caractere) randomString ($lungime) - intoarce un sir de caractere alfanumeric aleator, de lungime data
- (array de randuri) sortArray ($array_de_randuri, $cheie_sortare, $tip_sortare = 'ASC', $cheie_secundara = false, $tip_sortare_secundara = 'ASC') - sorteaza un array de randuri dupa cheia sau cheile date
- (sir de caractere) percent ($valoare, $total, $zecimale = 0) - intoarce procentul valorii din total, inclusiv cu caracterul "%"
- (data in format unixtime) tomorrow ($data_in_orice_format_inclusiv_unixtime = false) - intoarce data in format unixtime pentru ziua urmatoare zilei date, sau pentru ziua urmatoare zilei de azi
- (data in format unixtime) friday ($data_in_orice_format_inclusiv_unixtime = false) - intoarce data in format unixtime pentru prima vineri de dupa data specificata sau dupa ziua de azi
- (numar) age ($data_nastere, $format = 'dmy') - intoarce varsta unei persoane nascuta in ziua data
- (sir de caractere) shortDesc ($sir_de_caractere, $numar_maxim_caractere, $incheiat_cu = '...') - intoarce primele cuvinte pana la $numar_maxim_caractere din sirul dat, iar daca nu s-au putut

intoarce toate cuvintele sirul de caractere intors va fi completat cu incheierea data (de exemplu puncte de suspensie)

- (continut url) getUrl ($link, $https = false) - intoarce continutul unui url dat

5.5.2. Trimiterea mail-urilor

Pentru a trimite rapid un email, aveti la dispozitie functia mailSend ($adresa_email, $subiect, $mesaj) din modulul de functii extinse care va trimite mail la adresa sau adresele specificate separate prin virgula (ca parte a unui singur email, pentru a trimite pe rand la mai multe adrese, apelati de mai multe ori functia) cu subiectul si mesajul dat.

Se pot de asemenea atasa fisiere, apeland inaintea functiei de trimitere functia mailAttach ($cale_si_denumire_fisier, $nume = ''). Atasamentele vor fi adaugate la primul apel mailSend efectuat.

Email-urile sunt trimise folosind aplicatia third-party phpmailer. Pentru a adauga configurari suplimentare obiectului de trimitere a mesajului, adaugati o functie de control "mail___config" care primeste ca parametru obiectul $mail si intoarce obiectul $mail modificat cu functionalitatile sau proprietatile necesare.

5.6. Configurari avansate

Configurarile, asa cum am vazut, se impart in doua categorii:

- cele care stabilesc modul de functionare al nucleului phprel, centralizate in cfgs/
- cele care stabilesc caracteristici de layout/ prezentare ale aplicatiei, centralizate in web/settings/

In web/settings/ se configureaza setari precum:

- constante folosite in aplicatie, preexistente sau create de dumneavoastra, in settings/constants.inc.php
- setari ale paginilor, precum layout de popup, pagini virtuale sau template comun, in settings/pages.settings.inc.php
- tipare de validare si template-uri de tag pentru formulare, in form.settings.inc.php
- reguli de rescriere a url-urilor si operatii avansate efectuate la rescrierea url-urilor, in settings/rewrite.rules.php respectiv settings/extended.rewrite.rules.php
- resurse de cache, in cache.resources.php
- cadre de lista, in fisiere de forma settings/(denumire cadru).frame.php

In cfgs/ se configureaza setari precum:

- adresa url a site-ului si conexiunea la baza de date, in cfgs/config.inc.php
- locatia nucleului phprel si a altor fisiere/ directoare necesare, in cfgs/smartlocate.inc.php
- modulele si optiunile phprel active, in cfgs/advanced/optimizers.inc.php
- varianta phprel si setarile de cache, in cfgs/advanced/engine.settings.inc.php
- comportamentul nucleului phprel in diferite situatii pentru diferitele module, in cfgs/advanced/core.settings.inc.php
- adresa url a site-ului si conexiunea la baza de date pentru modul de dezvoltare, in cfgs/advanced/devel.config.inc.php

- conexiunile la bazele de date aditionale, in cfgs/advanced/more.dbs.inc.php
- comportamentul asistentului, in cfgs/advanced/assistant.inc.php

Anumite setari nu au fost inca explicate. In cfgs/advanced/engine.settings.inc.php:

- $Tcfg_engine – specifica varianta phprel folosita: "front" sau "back". Varianta poate fi detectata automat in functie de fisierele care au fost copiate in radacina site-ului.
- $Tche_clear_password – specifica o "parola" care poate fi folosita ca valoare a parametrului GET "cache:clear" pentru a sterge continutul tabelelor de caching pentru directorul (folder-ul curent)
- $Tche_cache_folder – fiecare sectiune a aplicatiei isi va gestiona continutul de cache intr-un "director" virtual separat, pentru a nu interfera cu alte sectiuni. Prin aceasta sectiune se poate stabili denumirea directorului.

In cfgs/advanced/core.settings.inc.php:

- $Tcfg_prevent_nav_updates_post – pe "true", linkurile de navigare si linkurile generate de liste nu vor rescrie parametrii POST
- $Tcfg_keyprefix – valoare folosita ca intermediar in calculul cheilor de securitate a url-urilor si sesiunilor si cookie-urilor de sesiune persistenta. Valoarea se poate modifica, dar este recomandat sa fie dependenta de site-ul in cauza si trebuie sa fie o constanta.
- $Tcfg_asyncTransmitKey – pe "true", solicita criptarea la trimiterea informatiilor prin cereri asincrone (A.J.A.X.). Cererile asincrone criptate nu vor fi salvate in cache. O cerere asincrona nu va fi criptata daca nu trimite date (parametrul "call" este 0).
- $Tcfg_recursiveDelete – pe "true", activeaza functia de stergere recursiva a liniilor pe baza legaturilor parent – child sau "desfacerea" legaturilor "linked".
- $Tfrm_live_forms – pe "true", activeaza optiunea de recunoastere si auto-activare a campurilor de formular. Orice tag clasic (html) de formular (de forma <input ... />, <form>, <select>, <textarea>) va fi automat identificat si codat sa se comporte ca un tag creat cu <&frm.(...)>. Util pentru a integra rapid un design deja realizat ca html si css intr-o pagina functionala. Se accepta pentru tag-uri si pseudo-atribute html similare atributelor phprel precum *, group, etc.
- $Tlst_allowDeleteIfSubrows – pe "true", permite stergerea liniilor dintr-o lista fara nici o verificare suplimentara. Pe "false", nucleul phprel va verifica automat daca linia respectiva are dependinte (linii child) si in caz afirmativ va ascunde actiunea de stergere pentru respectiva linie.

In plus, fisierul de configurare cfgs/advanced/devel.config.inc.php se incarca in locul fisierului cfgs/config.inc.php atunci cand se activeaza modul de dezvoltare. Implicit, configurarea include configurarea de baza config.inc.php, dar orice parametru poate fi suprascris in devel.config.inc.php ulterior, de exemplu se poate configura o alta baza de date pentru modul de dezvoltare decat pentru modul "live" prin adaugarea parametrilor de configurare respective la finalul fisierului.

In cfgs/advanced/assistant.inc.php:

- $Tasi_ruleprediction – pe "true", asistentul phprel va deduce/ intui regula mysql asociata paginii, atunci cand acest lucru este necesar
- $Tasi_selfparsers – pe "true", asistentul phprel va inlocui automat tag-urile speciale phprel
- $Tasi_autoglobalize – pe "true", asistentul phprel va declara automat ca globale anumite variabile din elementele secundare de pagina care par a fi folosite in template-uri
- $Tasi_rowsperpageidentifier – atunci cand aveti de construit multe liste care trebuie sa aiba un numar variabil de linii pe pagina (specificate, de exemplu, dintr-un select-box), se poate indica

prin acest parametru denumirea select-box-ului standard care se ocupa de aceasta functionalitate si asistentul phprel va prelua din eventualul parametru POST cu respectivul nume numarul de linii afisate.

- $Tasi_setdefaultview – indica asistentului phprel textul sau imaginea folosita pentru a desemna actiunea de vizualizare pe o lista
- $Tasi_defaultviewclass – indica asistentului phprel clasa css a tag-ului care contine actiunea de vizualizare
- $Tasi_setdefaultedit – similar pentru actiunea de editare
- $Tasi_defaulteditclass – similar pentru actiunea de editare
- $Tasi_setdefaultdel – similar pentru actiunea de stergere
- $Tasi_defaultdelclass – similar pentru actiunea de stergere
- $Tasi_dontusepage, $Tasi_requiredminimum, $Tasi_deltarating, $Tasi_certaintyrating – parametrii avansati care seteaza comportamentul asistentului la predictia regulii mysql. Parametrii se pot modifica pentru a obtine un alt comportament al predictiei, cum ar fi sa nu se foloseasca denumirea paginii, sa se schimbe rating-ul minim pe care un tabel trebuie sa il obtina pentru a fi luat in considerare, sa se schimbe diferenta de rating minima intre doua tabele pentru a se considera ca exista o diferenta reala intre ele si sa se schimbe rating-ul minim pentru ca asistentul sa fie sigur de descoperire si sa inceteze orice alte cautari.

Important: toate setarile prezentate in aceasta sectiune sunt implicit configurate pentru maximum de functionalitate sau, in cazul unor functionalitati secundare, pentru viteza in executie si nu necesita modificari decat in momentul in care intra in conflict cu ceea ce incercati sa realizati. Se recomanda utilizarea lor cu configurarile implicite, si realizarea unei modificari doar in momentul in care este absolut necesar.

5.7. Importarea paginilor din Web Assistant

In situatia in care aveti de creat o aplicatie multilanguage sau cu un nivel de securitate ridicat, bazat pe grupuri de utilizatori, interfata Web Assistant va pune la dispozitie paginile necesare configurarii respectivelor functii, insa va fi evident necesar sa creati pagini similare si destinate utilizatorilor finali (care de regula nu vor avea acces la interfata Web Assistant). Pentru a nu fi nevoiti sa creati o pagina care deja a fost creata pentru interfata Web Assistant, se poate copia comportamentul unei pagini Web Assistant in aplicatia dumneavoastra, modificand doar template-ul/ aspectul paginii pentru a corespunde cerintelor dumneavoastra.

Astfel, pentru a copia o pagina din Web Assistant in aplicatia dumneavoastra, nu trebuie decat sa incarcati pagina cu denumirea "phprel/(pagina phprel corespunzatoare)" (folosind, evident, parametrul GET "page") si sa va asigurati ca paginii ii corespunde un template html in directorul web/templates/ al aplicatiei dumneavoastra. Bineinteles, template-ul respectiv il veti copia pentru inceput din core/phprel/web/templates/, si il veti modifica ulterior sa corespunda cerintelor proiectului dumneavoastra. Paginii respective nu va putea sa-i corespunda un php in web/pages/, php-ul fiind imprumutat din Web Assistant, dar ii va putea corespunde un eveniment preincarcare sau postincarcare.

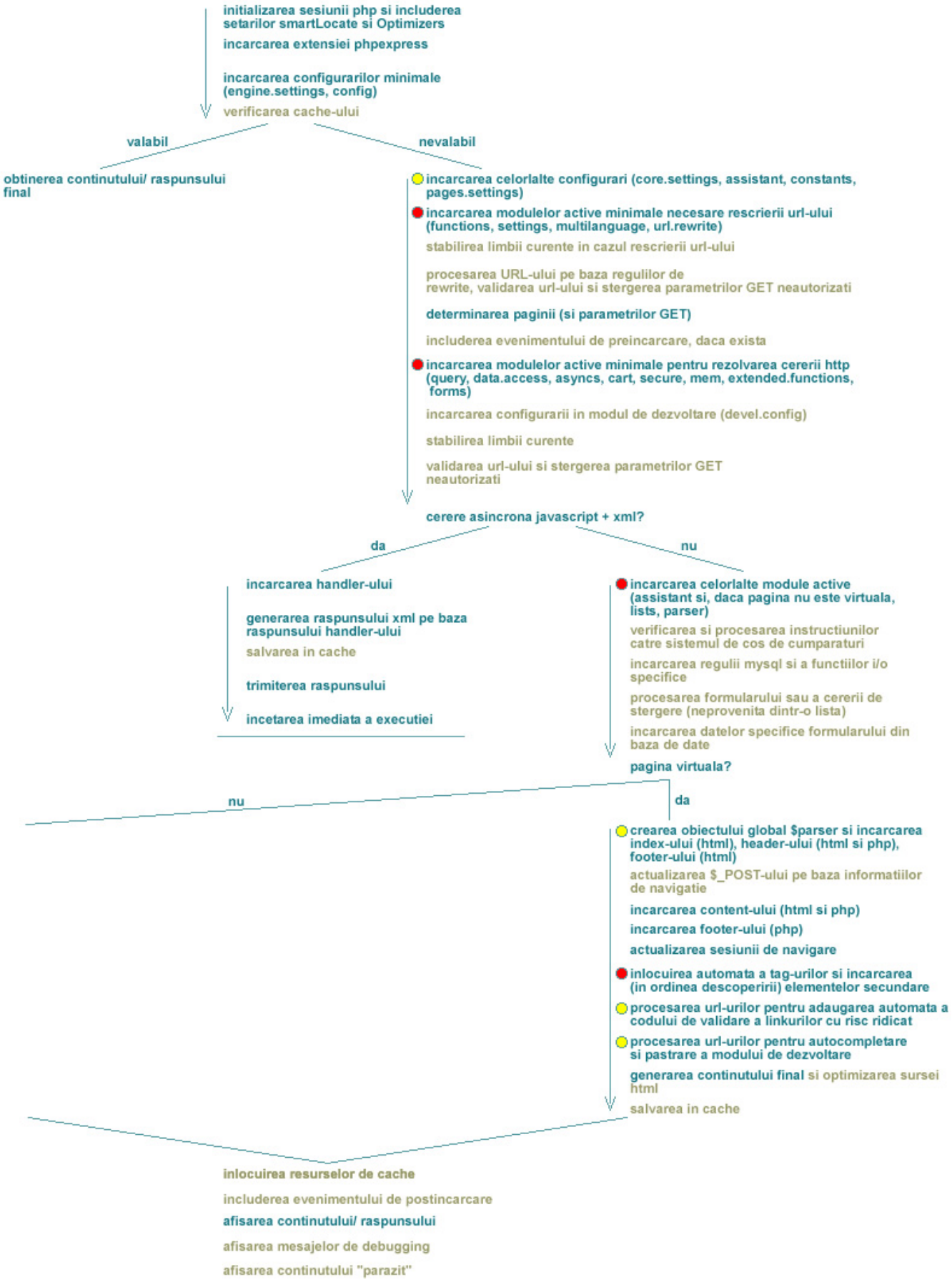
Paginile care pot fi importate din Web Assistant prin aceasta metoda simpla sunt:

- translations.content – incarcata specificand pagina ca fiind "phprel/translations.content" si copiind template-ul "translations.content.html" in web/templates/. Pagina corespunde paginii de traduceri din Web Assistant.
- translations.add.content – corespunzatoare paginii de modificare a traducerilor
- access.groups – corespunzatoare paginii de management al grupurilor de acces
- access.users – corespunzatoare paginii de management al utilizatorilor
- access.pages – corespunzatoare sectiunii de administrare a paginilor pentru restrictionarea accesului
- access.tables – corespunzatoare sectiunii de administrare a tabelelor pentru restrictionarea accesului
- access.set.pages – corespunzatoare sectiunii de administrare a politicilor de acces la pagini
- access.set.tables – corespunzatoare sectiunii de administrare a politicilor de acces la tabele
- maintenance.manage – corespunzatoare sectiunii de mentenanta a datelor din tabelele mysql (cu parametru GET "table" pentru a specifica tabelul in format phprel care obligatoriu are descriere de tabel activa)
- maintenance.manage.edit – corespunzatoare sectiunii de adaugare/ modificare a datelor intr-un tabel mysql (cu parametru GET "table" pentru a specifica tabelul in format phprel care obligatoriu are descriere de tabel activa)

Pentru pagini care sunt similare ca functionalitate acestor pagini, se poate copia usor template-ul, modifica acolo unde este necesar pentru a corespunde proiectului si importa pagina din Web Assistant folosind prefixul "phprel/" la denumirea paginii.

5.8. Schema completa de functionare

Pentru a avea o imagine clara a succesiunii proceselor in nucleul phprel, este recomandat sa studiati schema completa de functionare prezentata mai jos. Procesele descrise cu culoare gri vor fi rulate doar in anumite conditii. Procesele marcate cu rosu sunt mari consumatoare de resurse si timp de executie, iar cele marcate cu galben sunt medii consumatoare de resurse si timp de executie.


initializarea sesiunii php si includerea setarilor smartLocate si Optimizers
incarcarea extensiei phpexpress
incarcarea configurarilor minimale (engine.settings, config)
verificarea cache-ului
valabil
nevalabil
obtinerea continutului/ raspunsului final
incarcarea celorlalte configurari (core.settings, assistant, constants, pages.settings)
incarcarea modulelor active minimale necesare rescrierii url-ului (functions, settings, multilanguage, url.rewrite)
stabilirea limbii curente in cazul rescrierii url-ului
procesarea URL-ului pe baza regulilor de rewrite, validarea url-ului si stergerea parametrilor GET neautorizati
determinarea paginii (si parametrilor GET)
includerea evenimentului de preincarcare, daca exista
incarcarea modulelor active minimale pentru rezolvarea cererii http (query, data.access, asyncs, cart, secure, mem, extended.functions, forms)
incarcarea configurarii in modul de dezvoltare (devel.config)
stabilirea limbii curente
validarea url-ului si stergerea parametrilor GET neautorizati
cerere asincrona javascript + xml?
da
nu
incarcarea handler-ului
generarea raspunsului xml pe baza raspunsului handler-ului
salvarea in cache
trimiterea raspunsului
incetarea imediata a executiei
incarcarea celorlalte module active (assistant si, daca pagina nu este virtuala, lists, parser)
verificarea si procesarea instructiunilor catre sistemul de cos de cumparaturi
incarcarea regulii mysql si a functiilor i/o specifice
procesarea formularului sau a cererii de stergere (neprovenita dintr-o lista)
incarcarea datelor specifice formularului din baza de date
pagina virtuala?
nu
da
crearea obiectului global $parser si incarcarea index-ului (html), header-ului (html si php), footer-ului (html)
actualizarea $_POST-ului pe baza informatiilor de navigatie
incarcarea content-ului (html si php)
incarcarea footer-ului (php)
actualizarea sesiunii de navigare
inlocuirea automata a tag-urilor si incarcarea (in ordinea descoperirii) elementelor secundare
procesarea url-urilor pentru adaugarea automata a codului de validare a linkurilor cu risc ridicat
procesarea url-urilor pentru autocompletare si pastrare a modului de dezvoltare
generarea continutului final si optimizarea sursei html
salvarea in cache
inlocuirea resurselor de cache
includerea evenimentului de postincarcare
afisarea continutului/ raspunsului
afisarea mesajelor de debugging
afisarea continutului "parazit"

6. Index

Aceasta sectiune acopera exhaustiv toate entitatile din phprel, cu o scurta descriere.

6.1. Tag-uri

1. <&(nume)> - variabila globala sau speciala, (nume) poate fi:
   - denumirea unei variabile globale
   - referinta la o cheie a unui array global
   - apel al unei functii permise
   - un tag special de variabila globala:
     - <&async{(cerere asincrona)}>
     - <&nav.generate>
     - <&nav.back>
     - <&nav.session.back>
     - <&nav.session.jump>
     - <&nav.session.jumpto{(pagina)}>
     - <&cart.add{(id produs)[, (cantitate)]}>
     - <&cart.more{(id produs)[, (cantitate)]}>
     - <&cart.less{(id produs)[, (cantitate)]}>
     - <&cart.remove{(id produs)}>
     - <&cart.total>
     - <&cart.items>
     - <&cache off>
2. <&loop "(nume)">...<&endloop> - variabila de bucla, (nume) poate fi:
   - orice denumire a unui array global
   - un sir de caractere reprezentand o instructiune getData
   - o denumire de tabel
   - "cart.contents[*(nivel)]"
   - "cart.orders[*(nivel)]"
   - se pot folosi de asemenea urmatoarele tag-uri in contextul unui tag de loop:
     - <&row> - randul curent, incepand cu 0
     - <&row+1> - randul curent, incepand cu 1
     - <&odd> - "true" daca randul este impar, "false" altfel
     - <&(denumire loop).row>
     - <&(denumire loop).row+1>
     - <&(denumire loop).odd>
3. <&include "(nume)" /> - incluziune a unui element, (nume) poate fi un sir de caractere reprezentand un element de pagina existent in web/templates/ si, eventual, in web/pages/.

4. <&if{(conditie)}>...<&else>...<&endif> - tag conditional. Conditia va fi in limbaj php cu cateva limitari privind apelul de functii si anumiti operatori. Este afisata una din cele doua ramuri, iar ramura "else" poate lipsi.
5. <&frm.(nume)[{(indicatii)}]> - tag de formular, convertit intr-unul din tag-urile html <form>, <input />, <select>, <textarea>. (indicatii) pot fi unul sau mai multe atribute referitoare la un tag de formular, separate prin punct-si-virgula.
6. <&lst.(nume){(indicatii)}> - tag de lista, inlocuit cu un "include" al unei liste construita pe baza unui "layout" si a unui "frame", folosind indicatiile date ca unul sau mai multe atribute referitoare la un tag de lista, separate prin punct-si-virgula.
7. <&async{(cerere asincrona)}> - tag de cerere asincrona, inlocuit cu apelul functiei de trimitere a unei cereri asincrone, configurat pe baza instructiunii date sub forma (hadler).(caller).(target)[.(url)[.(processor)]].
8. <&lng{(eticheta)}> - tag de traducere a unui text identificat printr-o eticheta in limba curenta.
9. <&js> ... <&endjs> - tag prescurtat pentru tag-ul <script type="text/javascript" language="javascript">...</script>
10. <&cache.(resursa)> - tag de resursa cache, inlocuita cu valoarea intoarsa de functia ce desemneaza respectiva resursa.
11. <&postload.(variabila)> - tag de variabila postincarcare, inlocuita cu valoarea variabilei corespunzatoare din evenimentul postincarcare al paginii
12. <&sec{(link)}> - tag de solicitare manuala a atribuirii codului de validare a unui url

6.2. Atribute

1. * - formulare, specifica validarea pentru campul respectiv.
   - valori: fara valoare, confirm, async.(...), (denumire pattern)
   - exemple: *, *="email"
2. action – formulare, specifica atributul html "action" corespunzator url-ului catre care se trimit datele din formular
   - valori: (url)
   - exemple: action="virtual.produse"
3. additional – formulare, specifica orice alte atribute html care trebuie incluse in tag-ul final, fara a fi modificate in nici un fel de nucleul phprel
   - valori: (continut html)
   - exemple: additional="onmouseover="javascript: &doc.explicatie.innerHTML='Completati denumirea produsului'""
4. align – liste, indica diferitele alinieri ale datelor in diferite coloane. Lista de coloane si alinieri separate prin virgula.
   - valori: "(coloana1): (coloana2): (aliniament), (coloana3): (aliniament), (...)"
   - exemple: align="lst.actions: right", align="titlu: descriere: center, pret: right"
5. async – formulare, specifica o cerere asincrona atribuita campului pe un anumit eveniment (implicit "onchange")
   - valori: (handler).(target) sau (handler).(caller).(target)[.(url)[.(processor)]] sau (eveniment): (handler).(caller).(target)[.(url)[.(processor)]]

- exemple: async="getproduse.id_produs", async="getrezervare.grupRezervare.divRezultat", async="onkeyup: getposibilitati.divPosibilitati"

6. autopost – formulare, specifica faptul ca formularul va fi trimis la fiecare modificare ("onchange") a campului
   - valori: nu are valoare
   - exemple: autopost
7. calendar – formulare, indica faptul ca respectivul camp va fi folosit pentru a selecta o data calendaristica incarcand un calendar dhtml
   - valori: fara valoare, "time"
   - exemple: calendar, calendar="time"
8. class – formulare, clasa css a campului
   - valori: (clasa css)
   - exemple: class="checkbox"
9. columns – liste, precizeaza denumirea coloanelor listei
   - valori: "(coloana1), (coloana2), (coloana3), (...)"
   - exemple: columns="Prenume, Nume, Data nastere, CNP"
10. defaultvalue – formulare, indica valoarea initiala a campului, la prima afisare, in cazul in care campul nu va prelua o valoare din POST sau MySQL
    - valori: (valoare initiala)
    - exemple: defaultvalue="5"
11. detailscolumn – liste, precizeaza coloana la click pe care se expandeaza/inchide zona de detalii corespunzatoare liniei
    - valori: (denumire camp corespunzator coloanei)
    - exemple: detailscolumn="nume"
12. detailstemplate – liste, indica denumirea template-ului de detalii (fara extensia ".html", din web/templates/)
    - valori: (denumire template detalii)
    - exemple: detailstemplate="utilizatori.detalii"
13. editinpopup – liste, indica faptul ca actiunea de editare si vizualizare trebuie deschisa in popup-ul principal, configurabil din web/settings/constants.inc.php
    - valori: nu are valoare
    - exemple: editinpopup
14. editlink – liste, indica linkul actiunii de editare si vizualizare
    - valori: (url)
    - exemple: editlink="edit.produs", editlink="index.php?page=edit.produs"
15. fields – liste, precizeaza in ordine denumirile campurilor asociate coloanelor listei
    - valori: "(camp asociat coloanei 1), (camp2), (camp3), (...)"
    - exemple: fields="prenume, nume, data_nastere, cnp"
16. frame – liste, precizeaza cadrul folosit la construirea listei (din web/settings/)
    - valori: (denumire cadru)
    - exemple: frame="phprel"
17. fromrule – formulare secundare, specifica faptul ca tabelul formularului este folosit in regula mysql
    - valori: nu are valoare

- exemple: fromrule

18. globaltemplate – formulare, specifica "template"-ul de tag folosit pentru toate campurile formularului, daca alt template nu este specificat local
    - valori: (denumire template tag)
    - exemple: globaltemplate="long"
19. group – formulare, grupeaza unul sau mai multe campuri pentru o validare comuna, o cerere asincrona comuna sau un cod prescurtat javascript. Atributul precizeaza grupurile in care se afla campul respectiv, separate prin virgula.
    - valori: (denumire grup)
    - exemple: group="rezervare"
20. group – liste, specifica o conditie de tip "group by" inclusa nemodificat in query-ul de selectare a datelor listei
    - valori: (conditie group by)
    - exemple: group="id_parent"
21. halign – liste, indica diferitele alinieri ale denumirilor coloanelor. Lista de coloane si alinieri separate prin virgula.
    - valori: "(camp asociat coloanei): (coloana2): (aliniament), (coloana3): (aliniament), (...)"
    - exemple: halign="pret: right", halign="titlu, descriere: center, pret: right"
22. having – liste, precizeaza o conditie de tip "having" adaugata nemodificat la query-ul de selectare a datelor listei
    - valori: (conditie having)
    - exemple: having="pretmediu<500"
23. image – formulare, specifica o imagine folosita pentru submit, in locul unui text. Imaginea este data prin cale relativ la directorul web/ (daca este din images/) sau relativ la radacina aplicatiei in orice alt caz
    - valori: (imagine)
    - exemple: image="images/salveaza.gif"
24. label – formulare, specifica denumirea campului asa cum apare in validare
    - valori: (denumire reala camp)
    - exemple: label="Titlul stirii"
25. layout – liste, indica "layout"-ul folosit la construirea listei (template-ul html)
    - valori: (denumire layout)
    - exemple: layout="phprel"
26. maximumpages – liste, indica numarul maxim de pagini afisate in zona de paginare (navigare pe paginile listei), restul paginilor fiind ascunse (inlocuite cu "...")
    - valori: (numar pagini)
    - exemple: maximumpages="11"
27. method – formulare, specifica metoda de trimitere a datelor din formular, fie post, fie get, fie async
    - valori: async, get, post
    - exemple: method="async"
28. method – liste, specifica metoda de join a tabelelor, implicit INNER JOIN realizat direct prin conditia "where", se poate specifica "LEFT" pentru "LEFT JOIN".
    - valori: LEFT
    - exemple: method="LEFT"

29. minheight – liste, indica inaltimea minima a listei, chiar si in absenta a suficiente linii
    - valori: (inaltime minima)
    - exemple: minheight="300"
30. multiple – formulare, seteaza atributul html "multiple" al unui select-box care permite selectarea mai multor optiuni simultan
    - valori: nu are valoare
    - exemple: multiple
31. name – formulare principale, specifica atributul html "name" al tag-ului <form> rezultat, daca acest atribut trebuie sa fie diferit de sirul de caractere "form"
    - valori: (atributul html name)
    - exemple: name="booking"
32. noactions – liste, indica faptul ca lista nu va avea actiuni (asistentul phprel nu va adauga actiunile standard si coloana de actiuni nu va exista)
    - valori: nu are valoare
    - exemple: noactions
33. noautofilter – liste, indica nucleului phprel sa nu filtreze automat lista dupa datele din POST
    - valori: nu are valoare
    - exemple: noautofilter
34. noheader – liste, indica faptul ca lista nu va avea coloanele denumite in clar (linia de inceput a tabelului nu va exista)
    - valori: nu are valoare
    - exemple: noheader
35. options – formulare, indica optiunile select-box-ului
    - valori: (sir valid de definire a optiunilor)
    - exemple: options="inactiv, activ", options="1::activ, 0::inactiv", options="categorii.searchoptions::id_parent='<&_GET['id_parent']>'", options="subcategorii.searchoptions::1", options="$produse"
36. order – liste, specifica o conditie de tip "order by" inclusa nemodificat in query-ul de selectare a datelor pentru afisare in lista
    - valori: (conditie order by)
    - exemple: order="nume ASC, prenume ASC"
37. rows – formulare, specifica numarul de randuri al unui textarea
    - valori: (numar de randuri)
    - exemple: rows="7"
38. rowsperpage – liste, indica numarul de randuri pe pagina
    - valori: (numar de randuri)
    - exemple: rowsperpage="10"
39. setactionsanyway – liste, indica asistentului phprel sa adauge actiunile standard chiar daca alte actiuni au fost deja adaugate de dumneavoastra
    - valori: nu are valoare
    - exemple: setactionsanyway
40. setviewaction – liste, indica asistentului sa adauge si actiunea de vizualizare
    - valori: nu are valoare
    - exemple: setviewaction

41. size – formulare, dimensiunea unui select-box (in linii)
    - valori: (dimensiune)
    - exemple: size="5"
42. sortablecolumns – liste, precizeaza coloanele dupa care se pot ordona datele, la click pe denumirea coloanei. Implicit, la primul click se ordoneaza ascendent, la al doilea descendent. Alternativ, se pot preciza coloanele pentru care la primul click se va ordona descendent. Coloana dupa care este ordonata lista va aparea in *italics*. Lista de campuri atasate coloanelor, separate prin virgula, eventual precizand dupa doua-puncte sirul de caractere "DESC" pentru acele coloane care se vor ordona descrescator la primul click.
    - valori: "(camp asociat coloanei), (coloana2), (coloana3): DESC, (coloana4), (...)"
    - exemple: sortablecolumns="nume, data: DESC", sortablecolumns="denumire, pret"
43. style – formulare, atributele css ale campului, corespunzatoare atributului html "style"
    - valori: (atribute css)
    - exemple: style="width: 100px[sc]", style="width: 100px[sc] padding-left: 10px[sc]"
44. supportingtables – liste, specifica tabelele aditionale necesare construirii listei
    - valori: "(tabel1), (tabel2), (tabel3), (...)"
    - exemple: supportingtables="subcategorii, categorii, produse.imagini"
45. table – formulare, specifica tabelul folosit in determinarea automata a tipurilor campurilor si legaturilor dintre ele
    - valori: (denumire tabel)
    - exemple: table="produse.caracteristici"
46. table – liste, specifica tabelul principal din care se selecteaza datele pentru afisare in lista si de care se vor lega celelalte tabele (daca exista)
47. template – formulare, specifica "template"-ul de tag folosit pentru construirea tag-ului html final
    - valori: (denumire template de tag)
    - exemple: template="short"
48. type – formulare, tipul campului: text, hidden, select, textarea, checkbox, radio, file, etc.
    - valori: (tip camp)
    - exemple: type="textarea"
49. pseudo-atribut de tip – formulare, valoarea pentru "type" poate fi precizata direct, ca atribut, daca este primul atribut al tag-ului
    - valori: nu are valoare
    - exemple: select, textarea, checkbox, file, editor
50. value – formulare, valoarea fixa a campului, nu va fi suprascrisa la POST sau din MySQL
    - valori: (valoare fixa camp)
    - exemple: value="5"
51. width – liste, indica latimea anumitor coloane. Lista de coloane si latimi separate prin virgula.
    - valori: "(camp asociat coloanei): (latime), (coloana2): (latime), (coloana3): (coloana4): (latime), (...)"
    - exemple: width="nume: 150", width="nume: prenume: 150", width="lst.actions: 5%"

Un atribut phprel care nu are valoare va fi scris fara semnul egal sau orice alt semn la sfarsit, urmat eventual doar de punct-si-virgula. Un atribut care are valoare va fi urmat de egal si valoarea sa intre ghilimele, valoare in care urmatoarele caractere vor fi inlocuite:

- caracterul mai-mare ">", inlocuit cu [gt] si mai-mare-sau-egal ">=" inlocuit cu [gtoe]

- caracterul punct-si-virgula ";", inlocuit cu [sc]

Atributele vor fi separate prin punct-si-virgula ";", de exemplu:

- <&lst.lista{columns="Prenume, Nume, Data nastere, CNP"; fields="prenume, nume, data_nastere, cnp"}>

6.3. Alte entitati html

1. &frm.validate___(denumire camp): (mesaj); – prescurtare a scrierii "function validate___(denumire camp) { return (mesaj); }", defineste o functie de control javascript pentru mesaje de validare particularizate
2. function validateDisplay___(denumire formular) – folosita pentru a inlocui metoda clasica de afisare a erorilor intr-un div, functia preia ca parametru mesajul de eroare si poate actiona in orice mod e necesar situatiei particulare
3. - va contine erorile de validare ale formularului cu denumire data. In cazul in care nu exista un asemenea div, nucleul phprel va crea unul automat.
4. - folosit pentru incarcarea continutului respectivului element printr-o cerere asincrona (A.J.A.X.) realizata cu functia a() sau pageByAsync()
5. - folosit pentru transmiterea unui mesaj de confirmare dupa trimiterea datelor unui formular printr-o cerere asincrona (A.J.A.X.) se recomanda ca div-ul sa contina formularul in sine, pentru a preveni o refolosire a lui
6. &doc.(id/ grup) – prescurtare a scrierii "document.getElementById('(id)')" sau a unei scrieri similare repetate pentru un intreg grup de campuri (grupate folosind atributul "group").

6.4. Variabile

1. **Variabile de optimizare**
    1. $Topt_assistant – activeaza sau dezactiveaza asistentul phprel
    2. $Topt_asyncs – activeaza sau dezactiveaza modulul de cereri asincrone
    3. $Topt_cache – activeaza sau dezactiveaza modulul de cache
    4. $Topt_calendars – activeaza sau dezactiveaza calendarele dhtml
    5. $Topt_datatools – activeaza sau dezactiveaza modulul de acces la baza de date (care include functiile de acces rapid precum getData sau select, descrierile de tabel, regulile mysql)
    6. $Topt_editors – activeaza sau dezactiveaza editoarele html wysiwyg
    7. $Topt_formtools – activeaza sau dezactiveaza modulul de formulare
    8. $Topt_htmlsourceoptimize – activeaza sau dezactiveaza functia automata de optimizare a sursei html
    9. $Topt_jquery – activeaza sau dezactiveaza biblioteca javascript third-party jquery
    10. $Topt_listingtools – activeaza sau dezactiveaza modulul de liste
    11. $Topt_mailingtools – activeaza sau dezactiveaza functia de email
    12. $Topt_mem – activeaza sau dezactiveaza modulul de sesiuni persistente
    13. $Topt_misctools – activeaza sau dezactiveaza functiile extinse

14. $Topt_multilanguage – activeaza sau dezactiveaza modulul multilanguage
15. $Topt_mysqldatalayer – activeaza sau dezactiveaza descrierile de tabele si regulile mysql
16. $Topt_phpenhancedforms - – activeaza sau dezactiveaza re-validarea automata a formularelor in php
17. $Topt_querytools – activeaza sau dezactiveaza modulul de query-uri
18. $Topt_secureaccess – activeaza sau dezactiveaza modulul de securitate pe baza grupurilor de utilizatori
19. $Topt_securelinks – activeaza sau dezactiveaza modulul de validare a url-urilor cu risc ridicat
20. $Topt_securerows – activeaza sau dezactiveaza functia extinsa de securitate a tabelelor la nivel de linii
21. $Topt_sessionnavs – activeaza sau dezactiveaza navigarea pe baza sesiunii
22. $Topt_shoppingcart – activeaza sau dezactiveaza modulul de cos de cumparaturi
23. $Topt_smartlocates – activeaza sau dezactiveaza functia de includere a modulelor si fisierelor din alte directoare decat cele standard
24. $Topt_urladjustments – activeaza sau dezactiveaza completarea automata a url-urilor
25. $Topt_urlrewrites – activeaza sau dezactiveaza modulul de rescriere a url-urilor

2. Variabile de configurare generale

1. $Tcfg_accessTablesPrefix – prefixul tabelelor folosite de modulul de securitate pe baza grupurilor de utilizatori
2. $Tcfg_additionalDeleteIdentifier – identificator suplimentar al linkurilor de editare (de regula o parte a unui parametru GET folosit des pentru editare)
3. $Tcfg_additionalEditIdentifier – identificator suplimentar al linkurilor de stergere (de regula o parte a unui parametru GET folosit des pentru stergere)
4. $Tcfg_alink_delay_before_showloading – intervalul de timp in milisecunde care va trece inainte de afisarea imaginii de incarcare in cazul solicitarii continutului unei paginii prin functia a()
5. $Tcfg_alink_loading – imaginea sau mesajul afisat in timpul incarcarii continutului cu functia a()
6. $Tcfg_allLanguages – in cazul aplicatiilor multilingve, va contine prescurtarile la doua caractere ale tuturor limbilor folosite, separate prin virgula
7. $Tcfg_allow_alink_function – "true" daca functia a() este permisa si poate fi definita de nucleul phprel
8. $Tcfg_asyncTransmitKey – pe "true", solicita criptarea datelor trimise printr-o cerere asincrona (A.J.A.X.). O cerere criptata nu va fi salvata in cache. Cererile asincrone cu parametrul "call" egal cu 0 nu vor fi criptate.
9. $Tcfg_auto_add_wrapper_div – "true" daca div-ul cu id "async.(denumire element pagina)" va fi adaugat automat de nucleul phprel la inceputul elementului in cazul in care nu este gasit
10. $Tcfg_auto_virtual_pages – "true" daca paginile virtuale pot fi intuite automat pe baza prefixului "virtual." sau "virtual-"
11. $Tcfg_autosecure – activeaza optiunea de gasire si prelucrare automata a linkurilor pentru adaugarea codului de validare
12. $Tcfg_backUsersTable – denumirea tabelului de utilizatori ai backend-ului
13. $Tcfg_check_if_target_page_is_compatible – in cazul folosirii functiei a(), "true" daca nucleul phprel verifica faptul ca pagina destinatie (elementul "content") este compatibila cu incarcarea

printr-o cerere asincrona (A.J.A.X.), mai exact nu contine un tag de submit creat cu phprel, sau functii javascript
14. $Tcfg_cleanUpTimeout – numarul de zile care trebuie sa treaca de la ultima folosire a unei traduceri inainte ca aceasta sa fie stearsa automat in cazul folosirii optiunii de mentenanta automata a etichetelor
15. $Tcfg_db – denumirea bazei de date principale de pe serverul mysql
16. $Tcfg_dbhost – adresa serverului mysql
17. $Tcfg_dbpass – parola de conectare la serverul mysql
18. $Tcfg_dbuser – utilizatorul cu care se realizeaza conectarea la serverul mysql
19. $Tcfg_distributionDirectory – directorul distributiei phprel, care contine directoarele "core/" si "include/" corespunzatoare versiunii de sistem de operare si php instalate
20. $Tcfg_DEBUG – pe "true", activeaza mesajele de depanare
21. $Tcfg_defaultLanguage – in cazul aplicatiilor multilanguage, specifica limba principala
22. $Tcfg_defaultrecursivelevel – nivelul implicit de adancime al unei solicitari recursive de date, in cazul in care un nivel nu este specificat
23. $Tcfg_engine – specifica varianta de phprel folosita, "back" sau "front"
24. $Tcfg_from_mail – specifica adresa de email de la care se trimit in mod standard email-uri folosind functia mailSend
25. $Tcfg_from_name – specifica numele si prenumele atasate adresei de mail de la care se trimit in mod standard email-uri folosind functia mailSend
26. $Tcfg_keyprefix – cheie intermediara folosita in construirea cheii de securitate pentru validarea url-urilor si sesiunii
27. $Tcfg_keysTable – denumirea tabelului in care se pastreaza cheile de securitate a url-urilor deja emise
28. $Tcfg_keyTimeout – intervalul de timp dupa care o cheie de securitate a url-urilor expira
29. $Tcfg_manualsecure – activeaza sau dezactiveaza optiunea de inlocuire a tag-urilor care indica manual url-urile carora trebuie sa li se ataseze cod de validare
30. $Tcfg_maxsessionnaventries – numarul maxim de linkuri de intoarcere retinute in sesiunea de navigare
31. $Tcfg_newdb – array asociativ continand conexiuni la baze de date aditionale
32. $Tcfg_nodirectivesforfront – pe "true", modulul de validare a url-urilor nu va permite nici unul din parametrii GET "edit", "del" sau "view" pentru varianta phprel "front"
33. $Tcfg_performCleanUp – pe "true", functia de mentenanta automata a traducerilor va fi activata. La incarcarea unei pagini de backend (varianta phprel "back"), nucleul phprel va cauta traducerile vechi, nefolosite, si le va sterge.
34. $Tcfg_popup_pages – array asociativ indicand paginile pentru care se incarca "layout"-ul de popup
35. $Tcfg_postload_vars – array asociativ indicand variabilele "postincarcare" care trebuie inlocuite in continutul html final al paginii, impreuna cu valorile de inlocuire
36. $Tcfg_prevent_nav_updates_post – pe "true", nucleul phprel nu va actualiza variabila superglobala POST cu datele din linkuri de intoarcere sau linkuri ale listei
37. $Tcfg_preventfileextensions – array asociativ continand extensiile care nu vor fi acceptate la "upload"-ul de fisiere prin metodele puse la dispozitie de phprel

38. $Tcfg_projectname – denumirea proiectului, folosita pentru a construi chei de securitate in diverse situatii, si implicit ca titlu al paginii in cazul unei aplicatii nou incepute
39. $Tcfg_recursiveDelete – pe "true", activeaza functia phprel de stergere recursiva a liniilor din tabelele relationate
40. $Tcfg_safetyLimit – in cazul selectarii recursive a datelor din tabele relationate, indica o limita de siguranta pentru query-urile secundare (maximum de linii care vor fi selectate), pentru a nu genera rezultate mult prea costisitoare din punct de vedere al memoriei si timpului de executie
41. $Tcfg_secure – array asociativ indicand parametrii GET considerati cu risc ridicat, care trebuie validati de modulul de securitate a url-urilor
42. $Tcfg_smartincludedevel – array asociativ care indica pentru anumite directoare, directorul corespondent in situatia in care aplicatia este incarcata in modul de dezvoltare
43. $Tcfg_smartincludedir – array asociativ care indica locatia reala a unui director
44. $Tcfg_smartincludefile – array asociativ care indica locatia reala a unui fisier
45. $Tcfg_switchtemplate – array asociativ care precizeaza template-ul folosit de o anumita pagina in cazul in care nu are template propriu, ci imprumuta templete-ul altei pagini
46. $Tcfg_translationsTable – denumirea tabelului de traduceri
47. $Tcfg_uniquekey – "true" daca modulul de validare a url-urilor va folosi mereu chei de securitate unice, pe baza tabelului de chei anterioare
48. $Tcfg_urlrewritechars – array asociativ indicand ce caractere speciale dintr-un url vor fi inlocuite si cu ce anume, la apelul functiei rewrite___output, si care vor fi folosite pentru a reconstitui url-ul initial la incarcarea unei pagini
49. $Tcfg_useSession_withSecure – pe "true", indica nucleului phprel sa foloseasca sesiunea php in generarea codurilor de validare ale url-urilor cu risc ridicat
50. $Tcfg_virtual_pages – array asociativ care precizeaza ce pagini sunt pagini virtuale

3. Variabile de configurare a asistentului

1. $Tasi_autoglobalize – pe "true", activeaza functia de auto-globalizare a variabilelor locale din elementele secundare de pagina, care par a fi folosite in template
2. $Tasi_certaintyrating – valoarea minima a unui rating al unui tabel candidat la regula mysql curenta, pentru care tabelul este considerat ca sigur fiind corelat cu pagina
3. $Tasi_defaultdelclass – clasa css a actiunii de stergere dintr-o lista, atunci cand este adaugata automat de asistentul phprel
4. $Tasi_defaulteditclass – clasa css a actiunii de editare dintr-o lista, atunci cand este adaugata automat de asistentul phprel
5. $Tasi_defaultviewclass – clasa css a actiunii de vizualizare dintr-o lista, atunci cand este adaugata de asistentul phprel
6. $Tasi_deltarating – diferenta minima intre rating-urile a doua tabele, dincolo de care rating-urile sunt considerate diferite
7. $Tasi_detailstemplatepostfix – sufixul standard al unui template "de detalii" corespunzator unei linii dintr-o lista. Daca un fisier cu denumirea de forma "(lista).(pagina).(acest sufix)" exista, asistentul phprel va intelege automat ca lista necesita zona de detalii, nefiind nevoie sa mai declarati in tag atributul "detailstemplate".
8. $Tasi_dontusepage – pe "true", indica asistentului phprel sa nu se foloseasca de denumirea paginii in determinarea regulii mysql atasate

9. $Tasi_guesstype – array asociativ indicand tipurile campurilor html pe baza unor portiuni de denumire
10. $Tasi_requiredminimum – rating-ul minim necesar pentru ca un tabel candidat sa fie considerat eligibil pentru atasarea la o regula mysql
11. $Tasi_rowsperpageidentifier – variabila POST cea mai des folosita pentru specificarea numarului de randuri pe pagina al unei liste, atunci cand acest numar variaza (de exemplu prin selectarea dintr-un select-box). Asistentul phprel va folosi respectivul parametru pentru a pagina corect lista fara interventii din partea dumneavoastra.
12. $Tasi_ruleprediction – pe "true", activeaza optiunea de predictie automata a regulii mysql incorporata in asistentul phprel
13. $Tasi_selfparsers – pe "true", activeaza optiunea de inlocuire automata a tag-urilor phprel de catre asistentul phprel
14. $Tasi_setdefaultdel – textul sau imaginea folosita in crearea actiunii de stergere dintr-o lista, atunci cand aceasta actiune este adaugata de asistentul phprel
15. $Tasi_setdefaultedit – textul sau imaginea folosita in crearea actiunii de editare dintr-o lista, atunci cand aceasta actiune este adaugata de asistentul phprel
16. $Tasi_setdefaultview – textul sau imaginea folosita in creara actiunii de vizualizare dintr-o lista, atunci cand aceasta actiune este adaugata de asistentul phprel

4. Variabile de configurare a modulului de formulare

1. $Tfrm_defaultEmptyOption – textul folosit la crearea unei optiuni vide de catre nucleul phprel sau folosind functia addEmptyOpt()
2. $Tfrm_editorClass – clasa css de baza a editorului wysiwyg, asa cum este ea solicitata de aplicatia third-party care creaza editorul
3. $Tfrm_form_post_async_ok – mesajul de confirmare afisat dupa trimiterea cu succes a datelor unui formular printr-o cerere asincrona (A.J.A.X.)
4. $Tfrm_live_forms – pe "true", activeaza functia phprel de cautare si transformare automata a campurilor de formular obisnuite, inca nefunctionale, in campuri functionale
5. $Tfrm_prevent_stdIO – pe "true", previne folosirea functiilor de control "std___input" si "std___output" (atentie, nu cele de forma "std___input___*", "std___output___*")
6. $Tfrm_stdIO – lista de campuri separate prin spatii care trebuie transformate prin functiile I/O standard "std___input" si "std___output" (atentie, nu cele de forma "std___input___*", "std___output___*")
7. $Tfrm_Vasync – tiparul de afisare al mesajelor de eroare provenite dintr-o validare prin cerere asincrona (A.J.A.X.). Poate contine simbolul {message} inlocuit cu mesajul intors de handler ca eroare de validare.
8. $Tfrm_Vdiv – tiparul folosit pentru crearea div-ului de afisare a mesajelor de eroare la validare pentru un formular pentru care nu s-a gasit div-ul de erori. Trebuie sa aiba neaparat id-ul "frm.errors" care va fi inlocuit cu id-ul real in functie de denumirea formularului.
9. $Tfrm_Vempty – mesajul de eroare la validare a unui camp care trebuie completat obligatoriu. Poate contine simbolul {label} inlocuit cu denumirea reala a campului.
10. $Tfrm_Verrors – tiparul de afisare a mesajelor de eroare in div. Poate contine simbolul {errors} inlocuit cu erorile de validare.
11. $Tfrm_Vloading – textul afisat pe butonul de "submit" in timp ce formularul este validat.

12. $Tfrm_Vother – mesajul de eroare la validare a unui camp care trebuie sa respecte un anumit tipar. Poate contine simbolurile {label} si {type} inlocuite cu denumirea reala a campului respectiv denumirea tiparului necesar.
13. $Tfrm_Vpatterns – array asociativ continand tiparele de validare pe baza de "regular expressions"

5. Variabile de configurare a modulului de liste

1. $Tlst_allowDeleteIfSubrows – pe "true", permite stergerea unei linii dintr-o lista fara a verifica daca respectiva linie are dependinte in alte tabele (linii "child"). Pe "false", actiunea de stergere a unei linii cu dependinte nu va fi afisata.
2. $Tlst_defaultactionverifyifnonegiven – denumirea unei functii standard de confirmare a afisarii actiunilor corespunzatoare unei linii dintr-o lista, in cazul in care nu este pusa la dispozitie nici o functie de control
3. $Tlst_frame – array asociativ continand setarile "cadru" pentru diferite liste
4. $Tlst_loadingmessage – textul sau imaginea afisata in timpul incarcarii zonei de detalii a unei liste printr-o cerere asincrona (A.J.A.X.)
5. $Tlst_maximumpagestodisplay – numarul standard de pagini afisate in zona de paginare a unei liste (restul paginilor fiind ascunse prin "...")

6. Variabile de configurare a modulului de cos de cumparaturi

1. $Tshp_autoClearCartAfterCheckout – pe "true", indica nucleului phprel sa stearga automat continutul cosului dupa ce comanda a fost preluata
2. $Tshp_backLoginError – va indica daca a avut loc o eroare la autentificarea utilizatorului pe backend. Valoarea "1" va insemna "Utilizator/ parola incorecte" iar "2" va insemna "Contul a fost suspendat" (campul "status" este null sau 0).
3. $Tshp_cartFields – array asociativ de regula continand doar cheia "price" care indica denumirea reala a campului de pret din tabelul de produse
4. $Tshp_cartLoginError – va indica daca a avut loc o eroare la autentificarea utilizatorului pe frontend (in sistemul de cos de cumparaturi). Valoarea "1" va insemna "Utilizator/ parola incorecte" iar "2" va insemna "Contul nu a fost inca activat" (campul "confirmed" este null sau 0).
5. $Tshp_cartTransportFields – lista de campuri separate prin virgula care vor fi copiate din tabelul de produse in campurile corespondente din tabelul de sesiuni de cumparaturi, la adaugarea unui produs in cos
6. $Tshp_currentCartOperation – array asociativ continand informatii privind operatiunea curenta intreprinsa de sistemul de cos de cumparaturi, impreuna cu detalii relevante
7. $Tshp_ordersTable – denumirea tabelului de comenzi a sistemului de cos de cumparaturi
8. $Tshp_productsTable – denumirea tabelului de produse
9. $Tshp_redirectAfterLogin – url-ul catre care se va redirectiona pagina dupa actiunea de "login". In cazul in care nu se specifica nici un url (valoarea variabilei este "false"), pagina va fi redirectionata catre pagina curenta.
10. $Tshp_redirectAfterLogout – url-ul catre care se va redirectiona pagina dupa actiunea de "logout". In cazul in care nu se specifica nici un url (valoarea variabilei este "false"), pagina va fi redirectionata catre pagina curenta.
11. $Tshp_sessionTable – denumirea tabelului de sesiuni de cumparaturi
12. $Tshp_shipmentTable – denumirea tabelului continand adresele de livrare

13. $Tshp_usersTable – denumirea tabelului de utilizatori inregistrati ai sistemului de cos de cumparaturi

7. Variabile de configurare a modulului de sesiuni persistente

1. $Tmem_table – denumirea tabelului de sesiuni persistente precum si prefixul aplicat tabelului care stocheaza informatiile propriu-zise ale sesiunilor persistente (la care se adauga "_store").
2. $Tmem_timeoutForNotUsedVars – intervalul de timp dupa care o sesiune persistenta inca nefolosita este considerata invechita si este stearsa automat de nucleul phprel

8. Variabile de configurare a "cache"-ului

1. $Tche_cache_folder – denumirea "directorului" virtual care va contine toate intrarile salvate in cache pentru sectiunea curenta a aplicatiei
2. $Tche_cache_global_settings – array asociativ continand setarile globale ale cache-ului, care se aplica atunci cand o alta politica nu a fost explicit precizata pentru pagina incarcata
3. $Tche_cache_tables_prefix – prefixul tabelelor interne folosite pentru cache
4. $Tche_cacheAllowPartialHits – pe "true", permite redarea individuala din cache a elementelor secundare de pagina
5. $Tche_cacheifsession – pe "true", permite salvarea in cache a continutului chiar si atunci cand exista informatii in sesiunea php (care ar putea particulariza pentru utilizatorul curent continutul)
6. $Tche_cacheSentinels – pe "true", activeaza optiunea cacheSentinels, care va imbunatati timpul de raspuns si consumul de resurse la afisarea paginilor in cazul acceselor simultane, pe site-uri cu trafic intens
7. $Tche_clear_password – indica o "parola" utilizata pentru stergerea continutului cache-ului corespunzator "directorului" de cache curent, folosind parametrul GET "cache:clear" a carui valoare trebuie sa fie respectiva "parola"
8. $Tche_INCACHE – va indica (daca este folosita variabila in rewrite.rules.php) faptul ca fisierul de configurare a regulilor de rescriere a fost inclus de cache si nu de nucleul phprel, pentru a lua masuri in cazul in care anumite operatiuni intreprinse de dumneavoastra nu sunt compatibile cu incarcarea din cache
9. $Tche_monitor_only – pe "true", trece cache-ul intr-o stare semiactiva, in care salvarile si afisarile din cache vor fi dezactivate, pastrandu-se activa doar functia de monitorizare a bazei de date pentru a detecta modificarile ce pot influenta continutul din cache corespunzator altor sectiuni ale aplicatiei.
10. $Tche_register_resource – array asociativ care indica toate resursele de cache prezente in fisierul cache.resources.php
11. $Tche_validSessionKeys – array asociativ indicand cheile din sesiunea php care nu afecteaza cache-ul si sunt permise fara a cauza dezactivarea cache-ului

9. Variabile globale speciale

1. $parser – obiect din clasa "htmlparser" reprezentand instanta curenta a motorului de template-uri, cu template-urile incarcate in prezent.
2. $page – contine denumirea paginii incarcate
3. $rule – contine regula mysql atasata paginii
4. $openPopup – contine codul javascript necesar deschiderii popup-ului principal
5. $_MySQL – array asociativ care contine campurile unui formular incarcat pe baza regulii mysql

6. $edit – valoarea parametrului GET "edit" sau "view"
7. $websiteURL – contine url-ul radacina al aplicatiei, asa cum este specificat de dumneavoastra in fisierul de configurare

10. Variabile cu denumire variabila

1. $Tfrm_(denumire formular)_async_ok – indica mesajul de confirmare afisat dupa trimiterea datelor printr-o cerere asincrona (A.J.A.X.) particularizat pentru un anumit formular
2. $Tfrm_(denumire formular)_errors – indica erorile suplimentare de validare rezultate in urma unei validari din php (in rules/forms.process.php) care vor fi afisate in div-ul de erori de validare

11. Variabile sistem

1. $Tsys_* – variabilele care incep cu "Tsys_" sunt variabile sistem, folosite intern de phprel, ale caror valori nu trebuie modificate si nu necesita sa fie accesate.
2. $Tsys_jqyeryWasUsed – poate fi setata pe "true" daca ati folosit jQuery si nucleul phprel nu a detectat corect acest lucru
3. $Tsys_ lastDefinedAsync – "descriere" a ultimei cereri asincrone definite, folosita in conjunctie cu anumite functii (in cazuri foarte rare).
4. $Tsys_navback – linkul de intoarcere curent

6.5. *Functii*

1. Functii generale

1. addThumb ($nume_thumb, $latime, $inaltime, $tip = 'scale') - adauga un thumbail intr-un "buffer" temporar, pentru a fi creat in momentul apelarii functiei defaultUploadImage
   - parametrii:
     - $nume_thumb – denumirea thumbnail-ului, subdirectorul cu aceeasi denumire trebuie sa existe in directorul in care se salveaza imaginea dupa trimitere, director specificat prin defaultUploadImage sau dedus de nucleul phprel
     - $latime – latimea thumbnail-ului, maxima sau exacta in functie de tip
     - $inaltime – inaltimea thumbnail-ului, maxima sau exacta in functie de tip
     - $tip – "scale" sau "crop", scale va pastra dimensiunile pozei si le va scala sa incapa in $latime x $inaltime pastrand latura mai mare, "crop" va pastra latura mai mica, scaland imaginea la $latime x $inaltime, apoi suprafata din poza pe latura mai mare care depaseste aria dorita va fi taiata.
   - valoare intoarsa: true
2. backLogin($id) – autentifica un utilizator in "backend" pe baza unui id dat
   - parametrii:
     - $id – id-ul utilizatorului
   - valoare intoarsa: true/ false – utilizatorul a fost sau nu autentificat
3. backLogout() - efectueaza operatiunea de logout a utilizatorului de pe "backend"
   - parametrii:
     - nu are parametrii
   - valoare intoarsa: true/ false

4. defaultUpload ($denumire_variabila_upload = '', $director = '', $tabel = '') - muta un fisier trimis printr-un formular intr-un director final, prefixand denumirea fisierului cu un id corespunzator din tabel
   - parametrii:
     - $denumire_variabila_upload – denumirea campului de tip "file" trimis in formular, corespunzator cheii din array-ul superglobal $_FILES care desemneaza fisierul. Implicit, prima cheie a array-ului $_FILES care inca nu a fost procesata.
     - $director – directorul final in care va fi mutat fisierul, implicit "uploads/(denumire tabel)"
     - $tabel – tabelul mysql de care apartine linia pentru care s-a trimis fisierul (de regula unui fisier ii corespunde un camp intr-un tabel care tine denumirea fisierului), din care se va prelua id-ul maxim pentru prefixarea denumirii fisierului, in cazul adaugarii unei linii noi
   - valoare intoarsa: denumirea finala a fisierului (fara cale), incluzand prefixul cu id-ul liniei modificate, in cazul in care variabila globala $edit este mai mare ca 0, sau id-ul maxim din tabel plus 1, in cazul in care $edit este 0
5. defaultUploadImage ($denumire_variabila_upload = '', $director = '', $tabel = '') - muta o imagine trimisa printr-un formular intr-un director final, prefixand denumirea fisierului cu un id corespunzator din tabel si procesand imaginea pentru a crea thumbnail-urile indicate
   - parametrii:
     - $denumire_variabila_upload – denumirea campului de tip "file" trimis in formular, corespunzator cheii din array-ul superglobal $_FILES care desemneaza fisierul. Implicit, prima cheie a array-ului $_FILES care inca nu a fost procesata.
     - $director – directorul final in care va fi mutat fisierul, implicit "uploads/(denumire tabel)"
     - $tabel – tabelul mysql de care apartine linia pentru care s-a trimis fisierul (de regula unui fisier ii corespunde un camp intr-un tabel care tine denumirea fisierului), din care se va prelua id-ul maxim pentru prefixarea denumirii fisierului, in cazul adaugarii unei linii noi
   - valoare intoarsa: denumirea finala a fisierului (fara cale), incluzand prefixul cu id-ul liniei modificate, in cazul in care variabila globala $edit este mai mare ca 0, sau id-ul maxim din tabel plus 1, in cazul in care $edit este 0
6. fromSes ($denumire_variabila) - citeste din sesiunea protejata o variabila stocata anterior
   - parametrii:
     - $denumire_variabila – specifica variabila catre trebuie citita
   - valoare intoarsa: valoarea respectivei variabile sau NULL daca variabila nu exista.
7. getBackUser () - intoarce id-ul utilizatorului curent de pe "backend"
   - parametrii:
     - nu are parametrii
   - valoare intoarsa: id-ul utilizatorului curent sau false
8. getNavData () - intoarce un array asociativ corespunzator datelor din POST ale linkului precedent pe baza url-ului de intoarcere curent
   - parametrii:
     - nu are parametrii

- valoare intoarsa: array asociativ

9. locate ($fisier) - intoarce calea reala si denumirea fisierului cautat, pe baza setarilor smartLocate
   - parametrii:
     - $fisier – subdirectorul si denumirea fisierului, sau doar un subdirector
   - valoare intoarsa: calea reala, sir de caractere
   -
10. parseloop ($denumire_loop, $array_date, $element_pagina = 'auto') - inlocuieste un tag de bucla cu continutul dat
    - parametrii:
      - $denumire_loop – denumirea tag-ului de bucla
      - $array_date – array de randuri
      - $element_pagina – implicit, elementul curent din care se apeleaza functia, dar se poate specifica elementul de pagina in care se gaseste bucla
    - valoare intoarsa: true/ false
11. parsevar ($variabila, $valoare, $element_pagina = 'auto') - inlocuieste un tag de variabila cu valoarea data
    - parametrii:
      - $variabila – denumirea tag-ului
      - $valoare – valoarea cu care va fi inlocuit
      - $element_pagina – implicit, elementul curent din care se apleaza functia, dar se poate specifica elementul de pagina in care se gaseste tag-ul
    - valoare intoarsa: true
12. qescape ($sir_de_caractere) - asigura un sir de caractere care urmeaza a fi folosit intr-un query impotriva atacurilor de tip sql injection. Valoarea intoarsa va fi inclusa totdeauna intre apostroafe.
    - parametrii:
      - $sir_de_caractere – sirul de caractere care urmeaza a fi folosit intr-un query
    - valoare intoarsa: sirul de caractere protejat
13. rewrite___output ($fragment_url) - rescrie un fragment de url inlocuind caracterele speciale cu cele indicate in cfgs/advanced/core.settings.inc.php.
    - parametrii:
      - $fragment_url – fragmentul de url sau sirul de caractere care trebuie transformat pentru a fi inclus intr-un url
    - valoare intoarsa: sirul de caractere transformat pentru url
14. sessionNavBack ($jump = false, $return_page = false) - intoarce linkul precent celui curent din sesiunea de navigare (daca functia este activata)
    - parametrii:
      - $jump – pe "true", precizeaza functiei sa intoarca linkul anterior care are pagina ($page) diferita de pagina curenta ($page curent)
      - $return_page – se poate specifica o denumire de pagina prin acest parametru, la care se va face saltul inapoi pe linkurile de intoarcere. Se va intoarce primul link precedent care a incarcat pagina solicitata.
    - valoare intoarsa: url-ul de intoarcere

15. strToArray ($sir_de_elemente) - converteste un sir de caractere reprezentand elemente separate prin virgula intr-un array simplu
    - parametrii:
        - $sir_de_elemente – sir de elemente separate prin virgula
    - valoare intoarsa: array simplu de elemente
16. strToAssocArray ($sir_de_elemente) - converteste un sir de caractere reprezentand elemente separate prin virgula intr-un array asociativ
    - parametrii:
        - $sir_de_elemente – sir de elemente separate prin virgula
    - valoare intoarsa: array asociativ de elemente
17. toSes ($denumire_variabila, $valoare) - adauga in sesiunea protejata o variabila
    - parametrii:
        - $denumire_variabila – denumirea variabilei asa cum va fi solicitata ulterior din sesiune
        - $valoare – valoarea variabilei, un tip de date simplu sau array
    - valoare intoarsa: true

2. Functii de selectare si manipulare a datelor

1. alter ($array) - transforma valorile dintr-un array conform unor functii de control
    - parametrii:
        - $array – array de randuri rezultat in urma unui select sau construit prin alte metode
    - valoare intoarsa: array de randuri
2. fetch ($rezultat_mysql, $array = false) - intoarce rezultatele propriu-zise (ca informatie) dintr-o resursa de tip "rezultat mysql"
    - parametrii:
        - $rezultat_mysql – o resursa de tip "rezultat mysql", de regula intoarsa de functia select(), dar si de run_query() sau mysql_query(). Daca $rezultat_mysql nu este o resursa, de exemplu este un array, fetch() va intoarce valoarea parametrului $rezultat_mysql (util pentru query-uri pe multiple baze de date simultan, realizat prin functia select)
        - $array – pe "true", indica functiei sa intoarca rezultatul ca array de randuri indiferent de dimensiunea rezultatului
    - valoare intoarsa: sir de caractere (pentru un singur camp, o singura linie)/ array asociativ (pentru o singura linie)/ array de randuri (pentru mai multe linii sau cand $array e "true")
3. getData ($sir_instructiuni, $parametru_subfunctie) - selecteaza date din unul sau mai multe tabele conform unor instructiunii si pe baza unor subfunctii predefinite (line, search, filter, toid, toname, all, searchoptions, filteroptions, options)
    - parametrii:
        - $sir_instructiuni – precizeaza tabelul sau tabelele pe care se aplica o subfunctie, sau o functie creata de dumneavoastra
        - $parametru_subfunctie – un parametru propriu-zis trimis subfunctiei
    - valoare intoarsa: false/ array de randuri/ valoare simpla, rezultatul executarii instructiunii de catre subfunctie
4. getfield ($camp, $array) - intoarce valoarea unui camp/ cheie dintr-un array, fie dintr-un array asociativ, fie de pe prima linie a unui array de randuri

- parametrii:
  - $camp – denumirea campului sau cheii dorite
  - $array – un array asociativ sau un array de randuri
- valoare intoarsa: valoarea cheii/ sir vid

5. processPost ($regula_mysql) - proceseaza datele trimise prin POST de un formular conform regulii mysql date. De regula, se apeleaza in forms.process.php, pentru procesarea unui formular trimis unei paginii care are atasata o regula ($rule)
   - parametrii:
     - $regula_mysql – regula mysql conform cu care se proceseaza formularul, de regula $rule
   - valoare intoarsa: true
6. select ($campuri, $tabele, $where, $order = '', $group = '', $limit = '', $having = '') - construieste si executa un query de selectare a datelor conform parametrilor
   - parametrii:
     - conform unui query SELECT standard, dar intr-o ordine usor modificata
   - valoare intoarsa: rezultat mysql / null / array de randuri (in cazul query-urilor pe multiple baze de date simultan)
7. shift ($array) - trunchiaza un array la componenta imediat inferioara ca rang structural. Dintr-un array de randuri se va intoarce un array asociativ corespunzator primului rand, dintr-un array asociativ se va intoarce valoarea corespunzatoare primei chei, in cazul unei valori simple se va intoarce acea valoare.
   - parametrii:
     - $array – in general un array de randuri sau array asociativ, dar poate fi chiar si o valoare
   - valoare intoarsa: valoare/ array asociativ
8. simpleRule ($tabel, $campuri, $chei_corespondente_POST = '', $where = '') - construieste o regula mysql simpla
   - parametrii:
     - $denumire_variabila – denumirea variabilei asa cum va fi solicitata ulterior din sesiune
     - $valoare – valoarea variabilei, un tip de date simplu sau array
   - valoare intoarsa: array asociativ reprezentand o regula mysql
9. writeData ($sir_instructiuni, $parametru_subfunctie) - scrie date intr-un tabel pe baza conform unor instructiuni date si pe baza unei subfunctii solicitate (insert, update, mixed, delete)
   - parametrii:
     - $sir_instructiuni – sirul de instructiuni continand tabelul si subfunctia solicitata
     - $parametru_subfunctie – parametrul trimis subfunctiei
   - valoare intoarsa: true/ false/ id de inserare/ array de id-uri de inserare

3. Functii recomandate mysql

1. get_insert_id() - intoarce id-ul ultimei linii inserate intr-un tabel
   - parametrii:
     - nu are parametrii
   - valoare intoarsa: id-ul ultimei linii inserate

2. run_query ($sir_query, $denumire_baza_de_date_secundara = '') - executa un query mysql, monitorizand schimbarile referitoare la cache, inregistrand query-ul pentru afisare in debugger si selectand eventual baza de date indicata
   - parametrii:
     - $sir_query – query-ul mysql propriu-zis
     - $denumire_baza_de_date_secundara – daca nu se specifica, baza de date folosita va fi cea primara, altfel, baza de date va fi cea indicata din more.dbs.inc.php
   - valoare intoarsa: resursa corespunzatoare rezultatului mysql

4. Functii de construire sau configurare a formularelor

1. addEmptyOpt ($array_optiuni, $denumire_optiune_vida = false) - adauga o optiune vida la un array de optiuni deja construit
   - parametrii:
     - $array_optiuni – array de optiuni construite prin orice metoda valida
     - $denumire_optiune_vida – textul care apare pe optiunea "vida", in cazul in care nu se precizeaza, se va folosi cel standard configurat in core.settings.inc.php
   - valoare intoarsa: array de optiuni
2. addOpt ($valoare, $text) - adauga o optiune intr-un "buffer" temporar, folosit pentru a crea un numar arbitrar de optiuni, citite ulterior ca array cu functia retOpt()
   - parametrii:
     - $valoare – valoarea optiunii, care va aparea ca valoare a atributului html "value" al tag-ului <option>
     - $text – textul care apare pe optiune
   - valoare intoarsa: true
3. addValidate ($camp, $tip = 'std', $label = false) - adauga validare pe un camp conform specificatiilor. Validarile adaugate sunt retinute intr-un "buffer" temporar, folosit la crearea validarii formularului prin functia createValidation. De obicei, functia nu se apeleaza manual.
   - parametrii:
     - $camp – denumirea campului pe care se adauga validare
     - $tip – tipul validarii adaugate, fie "std" pentru validare de tip "camp obligatoriu nevid", fie "confirm", fie "async.(...)", fie denumirea unui pattern de validare
     - $label – denumirea reala a campului asa cum va aparea in mesajul de eroare
   - valoare intoarsa: true/ false
4. arrayToOpt ($array, $exclude = '') - transforma un array simplu sau array asociativ in array de optiuni eventual excluzand anumite chei date
   - parametrii:
     - $array – array simplu sau array asociativ. Optiunile vor avea ca valoare cheia si text valoarea.
     - $exclude – sir de chei separate prin virgula care vor fi excluse
   - valoare intoarsa: array de optiuni
5. createValidation () - creaza validare pentru ultimul formular declarat, conform specificatiilor date prin apeluri addValidate
   - parametrii:
     - nu are parametrii
   - valoare intoarsa: id-ul intern al tag-ului <script> adaugat

-
6. getInputValue ($denumire_camp, $prioritate = 'POST') - intoarce valoarea atribuita unui camp de formular
   - parametrii:
     - $denumire_camp – denumirea campului
     - $prioritate – implicit "POST", poate avea alternativ valoarea "MySQL". Valoarea campului se afla fie in array-ul superglobal POST fie in array-ul global MySQL, iar in mod normal cea din POST este prioritara celei selectate printr-o regula mysql
   - valoare intoarsa: valoarea campului/ false
7. mForm ($denumire, $action = '', $method = 'post', $class = '', $additional = '') - creaza un tag <form> pe baza specificatiilor pentru a inlocui ulterior un tag de forma <&frm.($denumire)> din template. Tag-urile vor fi create prin apeluri la diferite functii si apoi inlocuite concret in template printr-un apel la parseGroup().
   - parametrii:
     - conform unui tag <&frm.form> simplificat
   - valoare intoarsa: id-ul intern al tag-ului <form> creat
8. mHidden ($denumire, $valoare = '') - creaza un tag <input> hidden pentru a inlocui ulterior un tag de forma <&frm.($denumire)> din template. Tag-urile vor fi create prin apeluri la diferite functii si apoi inlocuite concret in template printr-un apel la parseGroup().
   - parametrii:
     - $denumire – denumirea tag-ului de formular <&frm.(denumire)>
     - $valoare – implicit, valoarea va fi preluata din POST sau MySQL, sau poate fi specificata si va fi completata in atributul "value" al tag-ului html rezultat
   - valoare intoarsa: id-ul intern al tag-ului hidden
9. mInfo ($denumire, $valoare = '') - retine o valoare pentru a inlocui ulterior un tag de forma <&frm.($denumire)> din template. Tag-ul de formular va fi inlocuit cu o simpla valoare (informativa), nu un camp. Inlocuirea concreta in template va avea loc in momentul unui apel la parseGroup().
   - parametrii:
     - $denumire – denumirea tag-ului de formular <&frm.(denumire)>
     - $valoare – valoarea cu care va fi inlocuit tag-ul. Implicit valoarea va fi preluata din POST sau MySQL
   - valoare intoarsa: id-ul intern al valorii create
10. mInput ($tip, $denumire = '', $valoare = '', $class = '', $style = '', $additional = '') - creaza un tag <input> conform specificatiilor pentru a inlocui ulterior un tag de formular de forma <&frm.($denumire)> existent in template. Tag-urile vor fi create prin apeluri la diverse functii si apoi inlocuite concret in template printr-un apel la parseGroup().
    - parametrii:
      - conform unui tag <&frm.(...)> simplificat
      - $daca denumirea nu se specifica, tipul este considerat "text" iar parametrul $tip se considera a fi denumirea
    - valoare intoarsa: id-ul intern al tag-ului creat

11. mOptions ($optiuni, $tag_select = ") - creaza tag-urile <option> corespunzatoare unui array de optiuni si la ataseaza unui select-box dat prin id intern ($tag_select) sau ultimului select-box definit.
    - parametrii:
        - $optiuni – array de optiuni valid
        - $tag_select – fara valoare, optiunile vor fi atasate ultimului select-box definit, apelul la mOptions() realizandu-se de obicei imediat dupa apelul la mInput()
    - valoare intoarsa: true/ false
12. mSubmit ($value, $imagine = '', $class = '', $style = '', $additional = '', $validate = false, $name = 'submit', $formhasGoPost = false) - creaza un tag <input> de submit conform specificatiilor pentru a inlocui ulterior un tag de forma <&frm.($name)> din template. Tag-urile vor fi create prin apeluri la diverse functii apoi inlocuite concret in template printr-un apel la parseGroup().
    - parametrii:
        - conform unui tag <&frm.submit> simplificat
        - $imagine – precizeaza calea si denumirea unei imagini pentru un submit de tip "image"
        - $validate – pe "true", formularului i se va atasa validare folosind un apel la createValidation() imediat dupa apelul la mSubmit().
        - $formhasGoPost – "true" daca formularul are si hidden "goPost"
    - valoare intoarsa: id-ul intern al tag-ului <input> creat
13. newTemplate ($denumire, $descriere) - creaza un nou template de tag-uri de formular.
    - parametrii:
        - $denumire – denumirea template-ului, folosita apoi in atributele "template", "globaltemplate" sau apeluri la setTemplate()
        - $descriere – array asociativ cu cheile "class", "style" si "additional", parametrii care sunt adaugati tag-urilor apartinand template-ului curent in cazul in care nu contin respectivii parametrii
    - valoare intoarsa: true
14. optionsByAsync ($async = false) - intoarce optiunile incarcate cu o cerere asincrona (A.J.A.X.). Functia este folosita pentru a incarca inca de la prima afisare un select-box "child" cu optiuni folosind codul din handler ca si cum a fost deja schimbata valoarea "parent"-ului.
    - parametrii:
        - $async – in cazul in care lipseste, functia intoarce optiunile corespunzatoare ultimei cereri asincrone. Parametrul trebuie sa fie valoarea variabilei $Tsys_lastDefinedAsync imediat dupa definirea cererii asincrone in cauza.
    - valoare intoarsa: array de optiuni / false
15. parseGroup ($denumire_grup, $element_de_pagina = 'content') - inlocuieste concret tag-urile de forma <&frm.(...)> din template-ul specificat cu tag-urile create prin apeluri la functii precum mInput, mForm, mSubmit, mHidden.
    - parametrii:
        - $denumire_grup – denumirea grupului care va fi inlocuit. Grupul implicit se numeste 'default'.
        - $element_de_pagina – elementul de pagina in care se inlocuieste grupul de tag-uri
    - valoare intoarsa: true

16. retOpt () - intoarce array-ul de optiuni rezultat in urma apelurilor la addOpt().
    - parametrii:
        - nu are parametrii
    - valoare intoarsa: array de optiuni / false
17. setGroup ($denumire_grup) - seteaza denumirea unui nou grup de tag-uri de formular, care vor fi inlocuite cu un apel la parseGroup().
    - parametrii:
        - $denumire_grup – denumirea noului grup
    - valoare intoarsa: true
18. setRows ($randuri) - seteaza numarul de randuri al unui "textarea" creat ulterior cu mInput().
    - parametrii:
        - $randuri – numarul de randuri al unui camp "textarea"
    - valoare intoarsa: true
19. setTemplate ($denumire_template) - seteaza template-ul curent valabil pentru apelurile ulterioare la mInput() sau mSubmit().
    - parametrii:
        - $denumire_template – denumirea template-ului folosit in continuare
    - valoare intoarsa: true

5. Functii de construire sau configurare a listelor

1. dfnAct ($text, $parametru, $adauga_la_link = '', $class = '', $confirmare = false, $adauga_la_click = '', $camp = '') - defineste o noua actiune pentru o lista.
    - parametrii:
        - $text – textul sau imaginea respectivei actiuni, asa cum apare in lista. Pentru o lista care inca nu a fost creata cu un apel la newList (de exemplu pentru liste create si inlocuite automat de asistentul phprel), textul va fi precedat de "(denumire lista)::". De exemplu, valoarea parametrului $text va fi: "lista::editare".
        - $parametru – parametrul GET al actiuni, de exemplu "edit".
        - $adauga_la_link – specifica alti parametrii GET adaugati la link, mai exact un fragment de url care va fi adaugat nemodificat la linkul actiunii. Fragmentul trebuie sa inceapa cu "&".
        - $class – clasa css a tag-ului care contine actiunea
        - $confirmare – pe "true", actiunea va cere confirmare printr-un dialog javascript inainte de a urma linkul.
        - $adauga_la_click – cod javascript adaugat la evenimentul "onclick"
        - $camp – implicit, campul din care se preia valoarea data parametrului GET al actiunii este cheia primara a tabelului principal al listei. In cazul in care actiunea trebuie sa preia valoarea unui alt camp din lista, se va specifica respectivul camp in format complet: "(tabel)___(camp)".
    - valoare intoarsa: true
2. dfnBind ($camp, $denumire_coloana, $latime = '', $aliniere_date = '', $aliniere_denumire_coloana = '') - defineste o coloana a unei liste si o leaga de un camp selectat.
    - parametrii:

        - $camp – campul de care este legata coloana. Pentru o lista care inca nu a fost creata cu un apel la newList (de exemplu pentru liste create si inlocuite automat de asistentul phprel), campul va fi precedat de "(denumire lista)::". De exemplu, valoarea parametrului $camp va fi: "lista::prenume".
        - $denumire_coloana – denumirea coloanei asa cum va aparea in lista
        - $latime – atributul html "width" al coloanei
        - $aliniere_date – atributul html "align" al datelor din lista (incepand cu primul rand si exceptand denumirea propriu-zisa a coloanei)
        - $aliniere_denumire_coloana – atributul html "align" corespunzator denumirii coloanei
    - valoare intoarsa: true
3. dfnFrame ($cadru) - defineste cadrul ("frame") folosit pentru o anumita lista.
    - parametrii:
        - $cadru – denumirea cadrului folosit din web/settings/. Pentru o lista care inca nu a fost creata cu un apel la newList (de exemplu pentru liste create si inlocuite automat de asistentul phprel), denumirea va fi precedata de "(denumire lista)::". De exemplu, valoarea parametrului $cadru va fi: "lista::phprel".
    - valoare intoarsa: true
4. dfnGroup ($conditie_group_by) - defineste o conditie "group by" pentru query-ul de selectare a datelor unei liste
    - parametrii:
        - $ conditie_group_by – conditia mysql "group by" asa cum va aparea in query. Pentru o lista care inca nu a fost creata cu un apel la newList (de exemplu pentru liste create si inlocuite automat de asistentul phprel), conditia va fi precedata de "(denumire lista)::". De exemplu, valoarea parametrului $ conditie_group_by va fi: "lista::id_parent".
    - valoare intoarsa: true
5. dfnHaving ($conditie_having) - defineste o conditie "having" pentru query-ul de selectare a datelor unei liste
    - parametrii:
        - $conditie_having – conditia mysql "having" asa cum va aparea in query. Pentru o lista care inca nu a fost creata cu un apel la newList (de exemplu pentru liste create si inlocuite automat de asistentul phprel), conditia va fi precedata de "(denumire lista)::". De exemplu, valoarea parametrului $conditie_having va fi: "lista::pretmediu<500".
    - valoare intoarsa: true
6. dfnLayout ($denumire_layout) - defineste "layout"-ul folosit pentru construirea listei, din web/templates/.
    - parametrii:
        - $denumire_layout – denumirea "layout"-ului folosit. Pentru o lista care inca nu a fost creata cu un apel la newList (de exemplu pentru liste create si inlocuite automat de asistentul phprel), denumirea va fi precedata de "(denumire lista)::". De exemplu, valoarea parametrului $denumire_layout va fi: "lista::phprel".
    - valoare intoarsa: true

7. dfnMethod ($tip_join) - defineste metoda de join a tabelelor query-ului de selectare a datelor unei liste
   - parametrii:
     - $tip_join – va avea valoarea "LEFT" in cazul unui join de tip "LEFT JOIN". Pentru o lista care inca nu a fost creata cu un apel la newList (de exemplu pentru liste create si inlocuite automat de asistentul phprel), valoarea va fi precedata de "(denumire lista)::". De exemplu, valoarea parametrului $tip_join va fi: "lista::LEFT".
   - valoare intoarsa: true
8. dfnSel ($campuri, $tabel, $conditii_where = '') - defineste un fragment de query referitor la un tabel si campurile necesare din tabelul respectiv, cu conditiile "where" necesare. Primul apel la dfnSel corespunzator unei liste defineste tabelul principal, iar prin conventie primeste conditiile "where" referitoare la filtrarea listei, in timp ce apelurile ulterioare definesc tabelele secundare si conditiile "where" de legare a tabelelor de tabelul principal.
   - parametrii:
     - $campuri – campurile care vor fi selectate din tabelul solicitat. Pentru o lista care inca nu a fost creata cu un apel la newList (de exemplu pentru liste create si inlocuite automat de asistentul phprel), campurile vor fi precedate de "(denumire lista)::". De exemplu, valoarea parametrului $campuri va fi: "lista::prenume, nume, data_nastere, cnp".
     - $tabel – tabelul solicitat pentru a fi inclus in query
     - $conditii_where – conditile de tip "where" atasate din acest fragment de query
   - valoare intoarsa: true
9. dfnSubSel ($campuri, $tabel, $conditii_where = '', $order = '', $as = '') - defineste un "subquery" al query-ului principal al unei liste, cu conditiile "where" si "order by" necesare.
   - parametrii:
     - $campuri – campurile care vor fi selectate din tabelul solicitat. Pentru o lista care inca nu a fost creata cu un apel la newList (de exemplu pentru liste create si inlocuite automat de asistentul phprel), campurile vor fi precedate de "(denumire lista)::". De exemplu, valoarea parametrului $campuri va fi: "lista::prenume, nume, data_nastere, cnp".
     - $tabel – tabelul solicitat pentru a fi inclus in "subquery"
     - $conditii_where – conditile de tip "where" atasate "subquery"-ului
     - $order – conditia "order by" a "subquery"-ului
     - $as – cum va fi denumit rezultatul "subquery"-ului in query-ul principal, folosind identificatorul mysql "AS"
   - valoare intoarsa: true
10. newList ($denumire_lista) - defineste configurarile de baza, cadrul si layout-ul pentru o noua lista. Pentru a inlocui o lista, trebuie mai intai apelata functia newList(), apoi apelate functiile dfnSel(), dfnSubSel(), dfbBind() necesare, iar in final trebuie apelata functia parseList() pentru inlocuirea propriu-zisa a tag-ului de lista din template. Pentru a apela newList(), trebuie sa existe un obiect din clasa "lstifc" cu denumirea listei in "scope"-ul global, care sa contina configurarile de baza ale listei.
    - parametrii:

        - $denumire_lista – denumirea listei nou creata
    - valoare intoarsa: true/ false
11. parseList ($element_de_pagina = '') - inlocuieste concret un tag de lista cu un "include" catre lista propriu-zisa
    - parametrii:
        - $element_de_pagina – se poate specifica elementul de pagina in care se inlocuieste tag-ul. Implicit, functia cauta in fiecare element pentru a gasi tag-ul, si il inlocuieste pe primul intalnit.
    - valoare intoarsa: true

6. Functii de manipulare a cosului de cumparaturi

1. addToCart ($id_produs, $cantitate = 1) – adauga un produs in cos.
    - parametrii:
        - $id_produs – id-ul produsului adaugat
        - $cantitate – cantitatea care se adauga in cos
    - valoare intoarsa: true/ false
2. cartContents ($nivel_recursivitate = false) - intoarce continutul cosului de cumparaturi
    - parametrii:
        - $nivel_recursivitate – pe "false", intoarce doar continutul tabelului de sesiuni de cumparaturi corespunzator sesiunii curente, impreuna cu o cheie aditionala "product" continand produsul atasat unei intrari din cos, o valoare intreaga pozitiva va indica selectarea recursiva a datelor pana la nivelul indicat
    - valoare intoarsa: array de randuri
3. cartItems () - intoarce numarul de produse din cos
    - parametrii:
        - nu are parametrii
    - valoare intoarsa: numarul de produse din cos
4. cartOrders ($nivel_recursivitate = false) - intoarce comenzile efectuate de utilizatorul curent.
    - parametrii:
        - $nivel_recursivitate – pe "false", intoarce doar continutul tabelului de comenzi corespunzator utilizatorului curent, o valoare intreaga pozitiva va indica selectarea recursiva a datelor pana la nivelul indicat
    - valoare intoarsa: array de randuri
5. cartTotal () - intoarce suma totala a produselor din cos.
    - parametrii:
        - nu are parametrii
    - valoare intoarsa: suma totala a produselor din cos
6. getSesid () - intoarce codul sesiunii de cumparaturi curente
    - parametrii:
        - nu are parametrii
    - valoare intoarsa: codul de sesiune
7. getUser () - intoarce id-ul utilizatorului autentificat
    - parametrii:
        - nu are parametrii
    - valoare intoarsa: id-ul utilizatorului curent

8. getUserDetails ($camp = false) - intoarce linia din tabelul de utilizatori corespunzatoare utilizatorului curent sau un anumit camp al liniei
    - parametrii:
        - $camp – implicit "false", functia intoarce toata linia din tabel, se poate specifica un camp anume care va fi intors de functie
    - valoare intoarsa: valoare/ array asociativ/ false
9. lessFromCart ($id_produs, $cantitate = 1) - scade cantitatea unui produs din cos.
    - parametrii:
        - $id_produs – id-ul produsului din cos caruia i se schimba cantitatea
        - $cantitate – cantitatea cu care se scade cantitatea curenta din cos
    - valoare intoarsa: true/ false
10. login ($id) - autentifica un utilizator dat prin id pe "frontend", prin sistemul de cos de cumparaturi.
    - parametrii:
        - $id – id-ul utilizatorului autentificat
    - valoare intoarsa: true/ false
11. logout () - efectueaza operatiunea de logout a utilizatorului de pe "frontend".
    - parametrii:
        - nu are parametrii
    - valoare intoarsa: true/ false
12. moreToCart ($id_produs, $cantitate = 1) - mareste cantitatea unui produs din cos.
    - parametrii:
        - $id_produs – id-ul produsului din cos caruia i se mareste cantitatea
        - $cantitate – cantitatea cu care se mareste cantitatea curenta din cos
    - valoare intoarsa: true
13. newCart ($cod_sesiune = false) - creaza o noua sesiune de cumparaturi sau reincarca o sesiune precedenta
    - parametrii:
        - $cod_sesiune – pe "false", creaza o noua sesiune, altfel, incarca sesiunea cu codul de sesiune dat
    - valoare intoarsa: true / false
14. previousShipmentDetails ($id_utilizator = false) - intoarce datele de livrare (linia din tabelul de adrese de livrare) folosite anterior de utilizatorul curent sau de un utilizator solicitat
    - parametrii:
        - $id_utilizator – pe "false", se considera utilizatorul curent, se poate specifica id-ul unui utilizator anumre
    - valoare intoarsa: array asociativ / false
15. removeFromCart ($id_produs) - sterge un produs din cos.
    - parametrii:
        - $id_produs – id-ul produsului din cos care va fi sters
    - valoare intoarsa: true/ false

7. Functii de construire a cererilor asincrone (A.J.A.X.)

1. a ($link, $element_pagina = 'content', $additional = '') - functie folosita din template-urile html intr-un tag de globala, pentru a crea un tag html <a> care incarca o pagina (mai exact un

element de pagina) printr-o cerere asincrona (A.J.A.X.), sau daca browser-ul nu accepta cereri asincrone, prin redirectionare catre linkul dat. Adauga automat div-ul cu id "async.(denumire element)" daca nu este adaugat manual de dumneavoastra, si verifica suplimentar daca pagina destinatie permite incarcare printr-o cerere asincrona (nu contine butoane de submit create cu phprel, sau functii javascript).
   - parametrii:
     - $link – linkul incarcat, fie prin redirectionare, fie printr-o cerere asincrona (A.J.A.X.), in functie de permisiunile si capabilitatile browser-ului
     - $element_pagina – implicit "content", se poate incarca orice alt element de pagina sau chiar multiple elemente de pagina printr-o lista de elemente separate prin virgula
     - $additional – orice alt continut html care trebuie adaugat la tag-ul <a> creat
   - valoare intoarsa: tag <a> corespunzator
2. contentByAsync ($async = false) - intoarce continutul unui div schimbat printr-o cerere asincrona data. Functia este folosita pentru a incarca div-ul de la prima afisare, ca si cum a fost actualizat de cererea asincrona in cauza (si fara a mai scrie cod suplimentar).
   - parametrii:
     - $async – in cazul in care lipseste, functia intoarce continutul div-ului corespunzator ultimei cereri asincrone. Parametrul trebuie sa fie valoarea variabilei $Tsys_lastDefinedAsync imediat dupa definirea cererii asincrone in cauza.
   - valoare intoarsa: continut div / false
3. makeAsyncPageRequest ($url, $handler, $caller, $target = false, $processor = false) - intoarce codul javascript de trimitere a unei cereri asincrone "cu pagina" conform parametrilor.
   - parametrii:
     - conform unei cereri asincrone
   - valoare intoarsa: codul javascript corespunzator cererii asincrone
4. makeAsyncRequest ($handler, $caller, $target = false, $processor = false) - intoarce codul javascript de trimitere a unei cereri asincrone simple, conform parametrilor.
   - parametrii:
     - conform unei cereri asincrone
   - valoare intoarsa: codul javascript corespunzator cererii asincrone
5. pageByAsync ($link, $element_pagina = 'content') - functie folosita din template-urile html intr-un tag de globala, pentru a genera codul javascript de solicitare printr-o cerere asincrona a unui element de pagina corespunzator unui link. Nu adauga automat div-ul cu id "async.(denumire element)" necesar, nici nu verifica automat daca elementul de pagina incarcat este compatibil cu incarcarea printr-o cerere asincrona (nu contine butoane de submit create cu phprel sau functii javascript).
   - parametrii:
     - $link – linkul paginii incarcate printr-o cerere asincrona
     - $element_pagina – elementul sau elementele separate prin virgula incarcate prin cererea asincrona
   - valoare intoarsa: true

8. Functii de utilizare a sesiunii persistente

1. fromMem ($denumire_variabila) - intoarce valoarea unei variabile din sesiunea persistenta

- parametrii:
    - $denumire_variabila – denumirea variabilei solicitate, asa cum a fost specificata la stocarea in sesiunea persistenta
- valoare intoarsa: valoarea variabilei/ false

2. toMem ($denumire_variabila, $valoare) - stocheaza valoarea unei variabile in sesiunea persistenta
    - parametrii:
        - $denumire_variabila – denumirea variabilei inregistrata in sesiunea persistenta
        - $valoare – valoarea variabilei, fie o valoare simpla fie un array
    - valoare intoarsa: true

9. Functii de verificare a accesului unui utilizator la o pagina

1. checkCredentials ($pagina = false) - intoarce accesul utilizatorului curent la o pagina data sau la pagina curenta. Acces 0 inseamna "acces restrictionat", 1 - "acces de vizualizare", 5 - "acces de vizualizare si modificare", 7 – "acces de vizualizare, modificare si stergere"
    - parametrii:
        - $pagina – pe "false", intoarce accesul la pagina curenta, altfel intoarce accesul la pagina specificata prin parametru
    - valoare intoarsa: 0/ 1/ 5 / 7

10. Functii de traducere si configurare multilanguage

1. lng ($eticheta = false) - intoarce traducerea in limba curenta a unei etichete date, sau codul limbii curente daca nu se specifica nici o eticheta
    - parametrii:
        - $eticheta – eticheta a carei traducere este solicitata, sau "false" pentru obtinerea codului limbii curente
    - valoare intoarsa: codul limbii curente/ traducerea etichetei
2. setlng ($cod_limba) - seteaza limba curenta ca fiind cea data prin parametru.
    - parametrii:
        - $cod_limba – codul prescurtat din doua caractere al noii limbi care va fi considerata in continuare "limba curenta"
    - valoare intoarsa: true

11. Functii de manipulare a cache-ului

1. highLevelCache () - functie apelata intr-o resursa de cache pentru a incarca modulele phprel, trecand astfel in cache de "nivel inalt" in care sunt disponibile toate functiile phprel.
    - parametrii:
        - nu are parametrii
    - valoare intoarsa: true

12. Functii de adaugare sau obtinere a codului de validare a unui url

1. secureCode ($link, $parametrii_riscanti = array(), $valori = array()) - intoarce codul de validare atasat unui link dat, pentru parametrii GET cu risc ridicat specificati sau descoperiti automat de nucleul phprel.
    - parametrii:
        - $link – url-ul pentru care se obtine codul de validare
        - $parametrii_riscanti – in cazul in care nucleul phprel nu poate determina automat parametrii GET cu risc ridicat din url (conform configurarii din

cfgs/advanced/core.settings.inc.php, $Tcfg_secure), parametrii pot fi specificati ca array simplu de parametrii
        - $valori – in cazul in care specificati parametrii GET, va trebui sa specificati si valorile lor din link, ca array simplu de valori
    - valoare intoarsa: cod de validare
2. secureLink ($link, $parametrii_riscanti = array(), $valori = array()) - intoarce url-ul dat ca parametru cu codul de validare atasat, pentru parametrii GET cu risc ridicat specificati sau descoperiti automat de nucleul phprel.
    - parametrii:
        - $link – url-ul caruia i se ataseaza codul de validare
        - $parametrii_riscanti – in cazul in care nucleul phprel nu poate determina automat parametrii GET cu risc ridicat din url (conform configurarii din cfgs/advanced/core.settings.inc.php, $Tcfg_secure), parametrii pot fi specificati ca array simplu de parametrii
        - $valori – in cazul in care specificati parametrii GET, va trebui sa specificati si valorile lor din link, ca array simplu de valori
    - valoare intoarsa: url-ul dat, cu cod de validare atasat

13. Metode ale obiectului global $parser, corespunzatoare clasei "htmlparser"

1. get () - intoarce continutul html complet al instantei curente
    - parametrii:
        - nu are parametrii
    - valoare intoarsa: continut html
2. loadfile ($denumire_element_nou, $fisier) - creaza un nou element de pagina incluzand continutul html dintr-un fisier dat.
    - parametrii:
        - $denumire_element_nou – denumirea elementului de pagina nou creat
        - $fisier – fisierul din web/templates/ incarcat, impreuna cu extensia
    - valoare intoarsa: true
3. parseloop ($element_pagina, $denumire_bucla, $date) - inlocuieste un tag de bucla dintr-un element de pagina cu continutul dat
    - parametrii:
        - $element_pagina – elementul de pagina in care se cauta si se inlocuieste bucla
        - $denumire_bucla – denumirea buclei cautate
        - $date – array-ul de randuri cu care se inlocuieste tag-ul
    - valoare intoarsa: true/ false
4. parsevar ($element_pagina, $denumire_tag, $valoare) - inlocuieste un tag de globala dintr-un element de pagina cu o valoare data.
    - parametrii:
        - $element_pagina – elementul de pagina in care se cauta tag-ul
        - $denumire_tag – denumirea tag-ului de forma <&(denumire)> cautat
        - $valoare – valoarea cu care se inlocuieste tag-ul
    - valoare intoarsa: true / false
5. retrievefile ($element_pagina) - intoarce continutul html al unui element de pagina
    - parametrii:

        - $element_pagina – elementul de pagina al carui continut este solicitat
    - valoare intoarsa: continutul html al elementului

14. Functii extinse

1. addToArray ($cheie, $array_de_randuri) - adauga o cheie la un array de randuri (pentru fiecare rand in parte). Cheia este adaugata cu valoarea "1", dar functia va fi folosita in conjunctie cu o functie de control "alter___(cheie) ($val, $d)", pentru a adauga valorile concrete.
    - parametrii:
        - $cheie – cheia adaugata la array
        - $array_de_randuri – array-ul la care se adauga cheia
    - valoare intoarsa: array de randuri
2. age ($data_nastere, $format_data = 'dmy') - intoarce varsta unei persoane nascuta la o data specificata
    - parametrii:
        - $data_nastere – data de nastere a persoanei
        - $format_data – formatul datei, "dmy", "mdy", "ymd"
    - valoare intoarsa: varsta persoanei
3. friday ($data = false) - intoarce data in format unixtime corespunzatoare urmatoarei vineri de dupa data specificata sau de dupa data de azi
    - parametrii:
        - $data – pe "false", echivalenta cu data de azi, altfel o data specificata in orice format, inclusiv unixtime
    - valoare intoarsa: data unixtime a urmatoarei vineri de dupa data specificata
4. getUrl ($link, $https = false) - intoarce continutul unui url specificat.
    - parametrii:
        - $link – url-ul al carui continut este solicitat
        - $https – pe "true", conectarea la url se va face prin HTTPS
    - valoare intoarsa: continutul url-ului dat
5. mailAttach ($cale_si_fisier, $nume = '') - adauga un "attachment" intr-un "buffer" temporar pentru a fi trimis impreuna cu urmatorul mail (trimis prin apel la mailSend)
    - parametrii:
        - $cale_si_fisier – calea si denumirea fisierului atasat la mail
        - $nume – in cazul in care denumirea fisierului in mail trebuie sa fie diferita de cea de pe disc, specificati denumirea
    - valoare intoarsa: true
6. mailSend ($catre_email, $subiect, $mesaj, $de_la = '') - trimite un mail la una sau mai multe adrese, de pe adresa specificata in web/settings/constants.inc.php.
    - parametrii:
        - $catre_email – email sau email-urile separate prin virgula catre care se trimite mesajul
        - $subiect – subiectul email-ului
        - $mesaj – continutul email-ului
        - $de_la – fara valoare, se va folosi numele configurat in web/settings/constants.inc.php, altfel se va trimite mail-ul de pe numele specificat

- valoare intoarsa: true

7. percent ($valoare, $total, $zecimale = 0) - intoarce procentul valorii din total, impreuna cu simbolul "%"
   - parametrii:
     - $valoare – valoarea parte din total
     - $total – valoarea din care se calculeaza procentul
     - $zecimale – precizia
   - valoare intoarsa: procent / "-"
8. randomString ($lungime) - intoarce un sir de caractere alfanumeric aleator, de lungime data
   - parametrii:
     - $lungime – lungimea sirului
   - valoare intoarsa: sir aleator
9. shortDesc ($sir_de_caractere, $numar_maxim_caractere, $incheiat_cu = '...') - intoarce o descriere scurta (primele caractere) dintr-un sir de caractere, pastrand intregi cuvintele.
   - parametrii:
     - $sir_de_caractere – sirul pentru care se solicita descrierea scurta
     - $numar_maxim_caractere – numarul maxim de caractere pe care le contine descrierea
     - $incheiat_cu – sirul de caractere cu care se incheie descrierea in cazul in care nu include intregul sir de caractere
   - valoare intoarsa: sir de caractere scurtat
10. sortArray ($array_de_randuri, $cheie_sortare, $tip_sortare = 'ASC', $cheie_secundara = false, $tip_sortare_secundara = 'ASC') - ordoneaza un array dupa una sau doua chei date.
    - parametrii:
      - $array_de_randuri – array-ul care se ordoneaza
      - $cheie_sortare – cheia principala dupa care se realizeaza ordonarea
      - $tip_sortare – ascendenta sau descendenta
      - $cheie_secundara – cheia secundara dupa care se realizeaza ordonarea, pentru elementele care au aceeasi valoare a cheii principale
      - $tip_sortare_secundara – ascendenta sau descedenta
    - valoare intoarsa: array de randuri
11. splitForHtml ($continut_bucla, $coloane = 2, $tag_separator = '<tr>') - imparte un array de randuri pentru afisare pe un numar exact de coloane, prin adaugarea unei chei suplimentare "split".
    - parametrii:
      - $continut_bucla – array de randuri care va fi impartit pe coloane
      - $coloane – numarul de coloane pe care se imparte array-ului
      - $tag_separator – tag-ul folosit pentru a inchide si incepe o noua linie
    - valoare intoarsa: array de randuri
12. toIds ($array_de_randuri) - intoarce un array simplu continand id-urile de pe fiecare rand din array-ul dat. Id-ul este fie cheia denumita "id" fie prima cheie din array-ul asociativ al unei linii.
    - parametrii:
      - $array_de_randuri – array-ul de randuri continand id-urile

- valoare intoarsa: array simplu de id-uri

13. tomorrow ($data = false) - intoarce data in format unixtime a zilei urmatoare zilei date prin parametrul $data.
    - parametrii:
        - $data – pe "false", se considera data de azi, se poate preciza data in orice format inclusiv unixtime
    - valoare intoarsa: data in format unixtime a zilei urmatoare

15. Functii sistem

1. sys___* - functiile prefixate de "sys___" sunt functii sistem folosite de nucleul phprel si care nu trebuie apelate manual.

6.6. Functii de control

1. alter___(camp) ($val, $d) - functie folosita la transformarea valorilor unui array, in conjunctie cu alter() sau getData(), intoarce valoarea transformata
2. alter___(tabel)___(camp) ($val, $d) - functie folosita la transformarea valorilor unui array, in conjunctie cu alter() sau getData(), intoarce valoarea transformata
3. cache___(resursa) () - defineste o resursa de cache, in web/settings/cache.resources.php
4. cart___add ($product, $fields, $values) - functie apelata la adaugarea unui produs in cos, intoarce valorile finale ale campurilor adaugate
5. cart___checkout ($fields, $values) - functie apelata la incheierea unei comenzi, intoarce valorile finale ale campurilor tabelului de comenzi
6. cart___done () - functie apelata dupa preluarea comenzii si crearea intrarii in tabelul de comenzi.
7. cart___login ($id_user) - functie apelata la autentificarea unui utilizator in sistemul de cos de cumparaturi, intoarce "true" sau "false" dupa cum utilizatorului ii este permis sa se autentifice sau nu
8. cart___logout () - functie apelata la logout in cazul utilizatorilor sistemului de cos de cumparaturi, intoarce "true" sau "false" dupa cum utilizatorului ii este permis sau nu "logout"-ul
9. cart___modify ($product, $fields, $values) - functie apelata la modificarea cantitatii unui produs din cos, intoarce valorile finale ale campurilor modificate
10. cart___refresh ($contents) - functie apelata in momentul autentificarii unui utilizator in sistemul de cos de cumparaturi, intoarce continutul actualizat al cosului de cumparaturi.
11. cart___remove ($product) - functie apelata in momentul stergerii unui produs din cos, intoarce "true" daca stergerea este permisa sau "false" altfel
12. cart___total ($total) - functie apelata pentru a transforma valoarea totalului din cos intr-un format final afisabil pe site, intoarce totalul formatat corespunzator
13. cart___validate ($row, $d) - apelata pentru a valida fiecare linie din cosul de cumparaturi, intoarce "true" daca linia va fi afisata sau "false" altfel
14. class___(lista) ($row, $class, $field, $d) - intoarce clasa css pentru o anumita linie si coloana a unei liste
15. class___(lista)___0 ($class, $field) - intoarce clasa css pentru denumirea unei anumite coloane a unei liste

16. class___(lista)___norecords () - intoarce clasa css a mesajului afisat intr-o lista atunci cand nu exista inregistrari
17. delete___(tabel) ($id) - functie apelata inainte de stergerea unei linii dintr-un tabel, intoarce "true" daca linia poate fi stearsa sau "false" altfel
18. input___(camp) ($field, $val) - functie de transformare a valorii scrise intr-un camp folosind o regula mysql si un formular trimis prin POST, intoarce valoarea finala
19. mail___config ($obiect_mail) - functie de configurare suplimentara a unui obiect de trimitere mail creat de functia mailSend
20. output___(camp) ($val, $d) - functie de transformare a valorii citite dintr-un camp conform unei reguli mysql, intoarce valoarea finala
21. postload___transform___content ($content) - functie apelata in cazul existentei unui eveniment de "postincarcare" a paginii, intoarce continutul html final al paginii
22. rowaction___(lista)___(parametru GET actiune) ($row, $name, $d) - functie de transformare a etichetei/ denumirii concrete a unei actiuni de pe o lista, intoarce denumirea sau imaginea finala folosita pentru actiunea respectiva
23. rowcheck___(lista)___(parametru GET actiune) ($row, $d) - functie de confirmare a adaugarii unei actiuni pentru o anumita linie dintr-o lista, intoarce "true" daca actiunea va fi adaugata sau "false" altfel
24. rowconfirm___(lista)___(parametru GET actiune) ($row, $confirm, $d) - functie de transformare a mesajului de confirmare afisat in momentul folosirii unei actiuni de pe o linie a unei liste (care necesita confirmare), intoarce mesajul de confirmare final
25. rowdetails___(lista) ($row, $d) - functie de confirmare a existentei zonei de detalii pentru o anumita linie dintr-o lista, intoarce "true" daca zona de detalii poate fi expandata sau "false" altfel
26. rowfooter___(lista) ($row, $d) - intoarce o eventuala zona de "footer" a unei linii dintr-o lista sub forma de continut html
27. rowheader___(lista) ($row, $d) - intoarce o eventuala zona de "header" a unei linii dintr-o lista sub forma de continut html
28. secureforms___failed () - functie apelata daca revalidarea automata in php a unui formular a esuat din cauza faptului ca formularul nu a fost recunoscut si a fost probabil trimis dintr-o alta sursa decat pagina corespunzatoare
29. securelinks___failed () - functie apelata daca validarea url-ului curent (care contine parametrii GET considerati cu risc ridicat) nu a fost realizata, fie pentru ca nu exista cod de validare, fie din cauza faptului ca respectivul cod este incorect
30. std___alter___(tabel)___(camp) ($val, $d) - functie de transformare a valorilor unui array, folosita in conjunctie cu alter() sau getData()
31. std___input ($field, $val) - functie de transformare a anumitor campuri date in cfgs/advanced/core.settings.inc.php la scrierea lor in baza de date, intoarce valoarea finala a campului
32. std___input___(tabel)___(camp) ($val) - functie de transformare a unui camp dintr-un tabel la scrierea in baza de date, intoarce valoarea finala a campului
33. std___output ($field, $val) - functie de transformare a anumitor campuri date in cfgs/advanced/core.settings.inc.php la citirea din baza de date, intoarce valoarea finala a respectivului camp

34. std___output___(tabel)___(camp) ($val, $d) - functie de transformare a unui camp dintr-un tabel la citirea din baza de date, intoarce valoarea finala a respectivului camp
35. std___transform___(tabel)___(camp) ($val, $d) - functie de transformare a unui camp dintr-un tabel la afisarea intr-o lista, intoarce valoarea finala a respectivului camp
36. style___(lista) ($row, $style, $field, $d) - intoarce atributele css "style" ale unei linii si coloane dintr-o lista
37. style___(lista)___0 ($style, $field) - intoarce atributele css "style" ale denumirii unei coloane dintr-o lista
38. style___(lista)___norecords () - intoarce atributele css "style" ale mesajului afisat intr-o lista atunci cand nu exista inregistrari
39. transform___(lista)___any ($field, $val, $d) - functie apelata pentru transformarea oricarei valori dintr-o lista, dupa aplicarea oricaror alte functii de transformare, intoarce valoarea finala afisata in lista
40. transform___(lista)___unknown ($field, $val, $d) - functie apelata pentru transformarea unei valori dintr-o lista atunci cand nu exista alte functii de transformare, intoarce valoarea transformata
41. transform___(tabel)___(camp) ($val, $d) - functie de transformare a unui camp dintr-un tabel la afisarea intr-o lista, intoarce valoarea finala a respectivului camp
42. validate___(camp) () - intoarce mesajul personalizat de validare a unui camp dat
43. validate___(formular)___(camp) – intoarce mesajul personalizat de validare a unui camp dat

6.7. Directoare si fisiere

1. **/**
    1. .htaccess – folosit pentru a seta regulile de rescriere pentru apache
    2. functions.front.php – fisier de functii create de dumneavoastra, inclus automat de nucleul phprel
    3. functions.back.php – fisier de functii create de dumneavoastra, inclus automat de nucleul phprel
    4. index.php – radacina aplicatiei, care include nucleul phprel
2. **cfgs/**
    1. config.inc.php – configurarea de baza a aplicatiei, incluzand conexiunea la baza de date
    2. smartlocate.inc.php – configurarea functiei smartLocate de includere a fisierelor din alte directoare
3. **cfgs/advanced/**
    1. assistant.inc.php – configurarea asistentului phprel
    2. core.settings.inc.php – configurarea diferitelor module ale nucleului phprel
    3. devel.config.inc.php – configurarea de baza a aplicatiei, in modul de dezvoltare
    4. engine.settings.inc.php – configurarea variantei de phprel si a cache-ului
    5. more.dbs.inc.php – configurarea bazelor de date aditionale
    6. optimizers.inc.php – activarea sau dezactivarea diverselor module si functii phprel
4. **core/**
    1. * - contine nucleul phprel
5. **core/phprel/**

    1. * - contine interfata de management a aplicatiei, Web Assistant, scrisa in phprel
6. **include/calendar/**
    1. * - contine calendarul dhtml, aplicatie third-party inclusa in phprel
7. **include/javascript/**
    1. async.js – extensia javascript a nucleului phprel, care se ocupa de cererile asincrone la client
    2. jquery.js – biblioteca javascript third-party inclusa in phprel
8. **include/phpmailer/**
    1. * - contine aplicatia third-party phpmailer, inclusa in phprel
9. **include/tiny_mce/**
    1. * - contine editorul html, aplicatie third-party inclusa in phprel
10. **uploads/**
    1. * - contine fisierele urcate pe server de aplicatia realizata cu phprel
11. **web/css/**
    1. default.css – css-ul inclus standard de phprel
    2. phprel.css – css-ul folosit de Web Assistant
12. web/handlers/
    1. * - contine "handler"-e cererilor asincrone
13. **web/images/**
    1. * - contine imaginile folosite de aplicatia scrisa cu phprel
14. **web/otherincs/**
    1. * - contine alte include-uri folosite de dumneavoastra in aplicatie
15. **web/pages/**
    1. * - contine php-urile corespunzatoare paginilor aplicatiei
16. **web/rules/**
    1. data.access.php – contine functii personalizate folosite in conjunctie cu getData()
    2. forms.mysql.rules.php – contine regulile mysql definite de dumneavoastra pe diferite pagini
    3. forms.process.php – contine validarile php, procesarile de formulare si redirectionarile definite de dumneavoastra
    4. io.functions.php – contine functiile IO personalizate pe pagina create de dumneavoastra
17. **web/rules/tables/**
    1. * - contine descrierile de tabele
18. **web/settings/**
    1. cache.resources.php – contine resursele de cache
    2. constants.inc.php – contine constantele aplicatiei
    3. default.frame.php – contine cadrul implicit incarcat pentru liste
    4. extended.rewrite.rules.php – contine operatiile extinse executate la rescrierea url-ului
    5. form.settings.inc.php – contine pattern-urile de validare si template-urile de tag de formular
    6. pages.settings.inc.php – contine configurarile privind paginile popup, template-urile imprumutate si paginile virtuale
    7. phprel.frame.php – contine cadrul folosit de listele din Web Assistant
    8. rewrite.rules.php – contine regulile de rescriere a url-urilor
19. **web/templates/**
    1. * - contine template-urile corespunzatoare paginilor aplicatiei
20. **webdevel/**

1. * - contine fisierele incarcate ca substitut al fisierelor reale in modul de dezvoltare, aceeasi structura de directoare ca si web/

7. Concluzii

Framework-ul phprel se adreseaza producatorilor de aplicatii web mijlocii si mici in primul rand, iar prin tehnologia avansata de caching poate fi folosit cu succes si in cazul aplicatiilor mari. Obiectivul sau este accelerarea procesului de dezvoltare web, imbunatatind totodata securitatea, ergonomia si flexibilitatea aplicatiilor. Prin realizarea automata (fara interventia dumneavoastra) sau asistata (cu un numar minim de linii de cod scrise) a anumitor sarcini tipice, phprel va permite sa scrieti site-uri web mai repede si mai eficient, reusind sa micsoreze frecvent timpul de implementare la o treime din timpul alocat in mod obisnuit.

Paradigma de programare este naturala si concisa, bazata pe relevanta structurii si pe reflexivitate (reflection-oriented programming) si permite o comunicare constructiva cu framework-ul care sporeste productivitatea. Fiecare proces si fiecare conventie phprel urmareste indeaproape structura web a aplicatiei si modul natural de intelegere a respectivului concept, framework-ul devenind astfel usor de inteles si de deprins, iar stilul de lucru fiind unul placut si intuitiv.

Echipa noastra isi propune ca phprel sa fie primul pas catre o redefinire a conceptelor de "rapiditate in dezvoltare" si "interactiune om-calculator" in realizarea proiectelor software, si continua sa depuna eforturi pentru crearea de noi unelte si sisteme capabile sa va ofere un ajutor real in scrierea aplicatiilor web.

Cu certitudinea ca phprel va ofera un cadru de dezvoltare placut si rapid pentru aplicatiile dumneavoastra web, va multumim pentru alegerea facuta si va asteptam sa ne adresati intrebarile, opiniile si sugestiile pe adresa noastra, parteneri@phprel.ro.