Masser - příručka programátora

sepsal Jakub Špičák 1.0 ($Revision: 1.2 $), $Date: 2003/10/07 08:07:27 $

Masser je prostředí pro tvorbu webovských aplikací založená na jazyce perl a technologii fast-cgi (viz www.fastcgi.com). Je připravena pro připojení k databázi a vytvořena tak, aby umožnila co nejefektivnější vývoj a opakované použití maximálního množství kódu.

Contents

1 Instalace

1.1 Požadavky

Požadavky na systém:

- operační systém typu UNIX (doporučeno je Linux, Solaris, HPUX)
- Perl 5.005 a vyšší
- web server s podporou fastcgi (například apache 1.3.26)

Požadavky na moduly do perlu:

- Delta a DeltaX 3.06+
- FCGI 0.65+

1.2 Instalace pro server apache

Předpokladem je, že je nainstalovaný a funkční modul mod_fastcgi. Nyní tedy:

- nakopírujte soubor masser.fcgi do adresáře, kde bude viditelný pro server apache (například DOCUMENT_ROOT/masser)
- nakopírujte adresář img/ a soubory v něm obsažené také na místo, kde budou viditelné pro server (doporučuji DOCUMENT_ROOT/masser/img)
- nastavte možnost spouštění CGI v adresáři, kde je soubor masser.fcgi, (Options +ExecCGI) a v obou adresářích z bezpečnostních důvodů odstraňte možnost výpisu obsahu adresáře
- soubor masser.conf nakopírujte na jedno z těchto míst:
  - adresář, kde je masser.fcgi (nedoporučuji)
  - /etc
  - /usr/etc

  a upravte tyto volby:

  - path - podle toho, kde budete mít umístěné aplikace a jejich konfigurace (obvykle například /usr/local/lib/masser)
  - img - podle toho, kde jste umístili adresář img/, v tomto příkladě to bude /masser/img
  - app_path - podle umístění masser.fcgi, v tomto příkladě jméno_serveru/masser
  - ostatní dle potřeby (viz dále v dokumentaci)
- pokud chcete masser jako statickou aplikaci (doporučuji), přidejte do konfigurace apache tento řádek:

```
FastCgiServer /cesta/k/masseru/masser.fcgi -processes 1
```

(počet procesů dle potřeby)

1.3 Instalace aplikace

Aplikace bude obvykle zřejmě obsahovat tyto adresáře:

- etc/ s konfiguračním souborem aplikace - ten nakopírujte do některého z adresářů podle path v masser.conf nebo do podadresáře config/ v některém z těchto podadresářů
- app/ se samotnou aplikací - ten nakopírujte nejlépe do některého z adresářů určeného v path v masser.conf - do podadresáře podle jména aplikace (například /usr/local/lib/masser/APLIKACE)
- img/ s obrázky specifickými pro aplikaci - ten patří do adresáře, kde jsou obrázky masseru, podadresáře podle jména aplikace (například /var/www/html/masser/img/APLIKACE)
- ostatní je závislé na aplikaci (sql/ adresář se skripty pro vytvoření databáze a podobně)
- upravte konfigurační soubor aplikace a spusťte:

```
http://server/cesta/masser.fcgi/APLIKACE
```

2 Jak to funguje?

Hlavní aplikace je v souboru masser, který musí být nainstalován tak, aby byl přístupný pro web server (například apache). Aplikace samotné mohou být uloženy v podstatě kdekoliv jinde, doporučuji je umístit mimo dosah web serveru, aby nebyly ani případnou chybou dostupné protokolem http.

Jde o fast-cgi aplikaci, hlavní program je tedy spuštěn web serverem (dle konfigurace buď při startu nebo při prvním požadavku).

2.1 Úvodní inicializace

Po startu program načte konfiguraci:

- pokud je nastavena proměnná prostředí `MASSER_CONFIG`, načte ji ze souboru podle jejího obsahu
- jinak čte soubor `masser.conf`

- pokud soubor neobsahuje absolutní cestu, masser hledá postupně v těchto adresářích:
  - aktuální adresář
  - adresář `/usr/etc`
  - adresář `/etc`

2.2 Zpracování požadavku

Po příchodu požadavku systém zpracuje parametry, nejdříve z URL požadavku, poté proměnné požadavku (GET nebo POST požadavek).

1. za částí URL vedoucí ke skriptu *masser* jsou zpracovány jedna či dvě části (v **debug módu** mohou být tři:
   - pokud je první argument `_source`, program zobrazí přeložený zdrojový kód stránky
   - první je považován za kód aplikace
   - druhý je považován za (seznam) stránek k zobrazení

   Příklady:

   `/cesta_k_aplikaci/masser/TEST`
   : zobrazí (default) stránku aplikace TEST

   `/cesta_k_aplikaci/masser/_source/TEST/page1`
   : zobrazí zdrojový kód stránky page1 aplikace TEST

2. dále jsou zpracovány tyto proměnné (pokud už nejsou načteny z předchozího bodu:

   APP
   : jméno aplikace (doporučuji velkými písmeny), pokud není vyplněno, doplní se *APP*

   PG
   : seznam stránek, které mají být zobrazeny, oddělený čárkami, pokud není vyplněno, doplní se *default*

   OP
   : seznam operací, které mají být provedeny, oddělený čárkami, pokud není vyplněno, je prázdný

   LANG
   : kód jazyka, pokud není vyplněno, doplní se *CZ*

   SCH
   : kód schématu, pro použití aplikací

   SID
   : Session Identification, pro použití aplikací

Poté je inicializována aplikace, pokud ještě nebyla použita:

- načtení konfigurace z adresáře podle konfigurace masseru, jméno konfiguračního souboru aplikace je předpokládáno ve tvaru *jméno aplikace převedené na malá písmena s příponou .conf*

- inicializace databázových připojení podle konfigurace
- načtení souborů s jazykovými variacemi, masser předpokládá, že se tyto soubory jmenují <app>_lang.<lang_code> (app vždy malými písmeny) a vyhledává je pro všechny cesty uvedené v masser.conf i <app>.conf takto:
  - adresář / aplikace / `lang` / soubor
  - adresář / aplikace / soubor
  - adresář / `lang` / soubor
  - adresář / soubor
- inicializace aplikace: program se pokouší najít soubor <app>.init (app převedeno na malá písmena) takto:
  - adresář / aplikace / `init` / soubor
  - adresář / aplikace / soubor
  - adresář / `init` / soubor
  - adresář / soubor

  Pokud jej nalezne, zkompiluje jej a provede *pouze jednou* pro každou instanci masseru.[1] Inicializace je určena k provedení akcí, které stačí realizovat pouze jednou za život aplikace (například načtení specifické konfigurace, nastavení speciálních parametrů databáze a podobně). Lze použít persistentní proměnnou *%p* - podrobnosti dále v příslušné kapitole.

Nakonec jsou provedeny všechny požadované oprace a zpracovány všechny požadované stránky.

Provedení operace nebo stránky probíhá takto:

- kompilace, pokud ještě nebyla stránka kompilována[2]
- kontrola databázových připojení vyžadovaných aplikací
- příprava SQL dotazů, pokud již nejsou připraveny[3]
- nastavení správného obsahu *%txt* dle vybraného jazyka (podle proměnné **LANG**)
- zápis zdrojového kódu, pokud je nastaveno *write_code_dir*
- nastavení persistentního hashe *%p*
- provedení kódu
- kontrola ukončení všech započatých transakcí v aktivních databázových připojeních, vypsání chyby pokud tomu tak není a případné ukončení transakce (dle *pending_transaction_rollback*)

[1] Toto neplatí pro variantu debug se zapnutým patřičným parametrem, v tomto případě se provádí při každém požadavku.

[2] v debug režimu vždy

[3] v debug režimu vždy

3 Konfigurace

3.1 masser.conf

`max_cycles`

počet cyklů, po nichž je instance aplikace restartována, pro provoz je doporučena hodnota 1000 - 5000

`path`

seznam adresářů, v nichž probíhá hledání souborů, pokud nejsou nalezeny v adresářích aplikace, oddělený dvojtečkami

`img`

adresář s obrázky aplikace masser (relativně ke kořeni web serveru)

`log_file`

cesta a jméno logu masseru

`log_error`

kam zapisovat chyby:

- file - do souboru určeného `log_file`
- stderr - na stderr
- kombinace obojího - file,stderr

`log_warn`

kam zapisovat varování

`log_info`

kam zapisovat informace

`log_debug`

kam zapisovat ladící informace

`debug`

pokud je nastaveno na 1, masser běží v ladícím režimu (pouze debug varianta programu)

`content-type`

nastavení content-type v hlavičkách stránek generovaných masserem (platí pro stránky, kde bylo použito `page_header`)

`app_path`

nastavení URL pro přístup k masseru bez úvodní specifikace protokolu (doplněno automaticky), například tedy `ip_nebo_adresa_serveru/cesta/masser.fcgi`

`access_log`

celá cesta k souboru, kde budou zapisovány informace o požadavcích na masser, soubor musí být zapisovatelný pro uživatele, pod nímž je masser spuštěn. Pokud není uvedeno, log není vytvářen.

`access_log_format`

formát záznamu v access_log, kromě libovolného textu lze použít tyto proměnné, které budou nahrazeny skutečností:

- *%app* - ID aplikace
- *%pg* - jméno stránky
- *%op* - jméno akce
- *%ip* - IP adresa, z níž přišel požadavek
- *%date* - datum a čas příchodu požadavku

`performance_log`

celá cesta k souboru, kde budou zapisovány informace pro zjištění výkonnosti systému masser, soubor musí být zapisovatelný pro uživatele, pod nímž je masser spuštěn. Pokud není uvedeno, log není vytvářen.

`performance_log_format`

formát záznamu v performance_log, kromě libovolného textu lze použít tyto proměnné, které budou nahrazeny skutečností:

- *%app* - ID aplikace
- *%pg* - jméno stránky
- *%op* - jméno akce
- *%ip* - IP adresa, z níž přišel požadavek
- *%date* - datum a čas příchodu požadavku
- *%dtime* - čas v sekundách nutný k vyřízení požadavku
- *%dmem* - počet bytů, o něž se zvětšila alokovaná paměť masseru v průběhu vyřizování požadavku
- *%mem1* - počet bytů alokovaných masserem před vyřízením požadavku
- *%mem2* - počet bytů alokovaných masserem po vyřízení požadavku

`max_memory_size`

maximální velikost paměti obsazená masserem, pokud je překročena, masser je ukončen, aby byla paměť uvolněna. Kontrola se provádí vždy po dokončení požadavku.

`pending_transaction_rollback`

masser provádí kontrolu, zda po dokončení stránky či operace byly ukončeny všechny započaté transakce v aktivních databázových připojeních. Pokud tomu tak není, vypíše hlášení do logu aplikace a pokud je tato proměnná nastavena na 1, provede rollback.

3.2 <app>.conf

Konfigurační soubory aplikací.

`path`

doplněk `path` z `masser.conf` pouze pro aktuální aplikaci, ostatní aplikace tímto nastavení nejsou dotčeny

`languages`

seznam jazyků, které budou pro aplikaci k dispozici, oddělený čárkami, doporučuji velká písmena

`log_file`

cesta a jméno logu aplikace

`log_error`

kam zapisovat chyby:

- file - do souboru určeného `log_file`
- stderr - na stderr
- kombinace obojího - file,stderr

`log_warn`

kam zapisovat varování

`log_info`

kam zapisovat informace

`log_debug`

kam zapisovat ladící informace

`dbX_*`

nastavení jednotlivých připojení k databázi, X je číslo 1 - 5:

- dbX_driver - databázový driver
- dbX_name - jméno databáze
- dbX_user - DB uživatel
- dbX_passw - DB heslo
- dbX_datestyle - formát datumu a času pro databázi
- dbX_trace - úroveň trasování (viz DeltaX::Database)
- dbX_stat - nastavení statistik, viz db_stat
- dbX_statfile - soubor pto yápis statistik, viz db_statfile

`write_code_dir`

pokud je uvedeno určuje adresář, do něhož budou ukládány zkompilované stránky a akce před jejich spuštěním - jméno je vždy <app>_<type>_<name>.code, kde <app> je jméno aplikace, <type> je typ (pg - stránka, ac - akce) a <name> je jméno objektu

`db_trace`

default nastavení úrovně trasování pro DeltaX::Database, toto nastavení je použito v případě, že není nastaveno odpovídající dbX_trace. Default je 0 (žádné trasování).

`db_stat`

nastavení statistik, toto nastavení je použito, pokud není uvedeno odpovídající dbX_stat. Nastavení je ve formátu `db_stat = type[,max_high[,max_all]]`, kde:

- type je nastavebí typu statistiky:
  - none - žádné statistiky (default)
  - sums - pouze počty provedených příkazů a celkový čas
  - high - sums a časově nejnáročnější příkazy
  - all - high a výpis všech příkazů
- max_high - počet časově nejnáročnějších příkazů, které budou vypsány (high a all), default je 5
- max_all - maximální celkový počet příkazů, které budou vypsány (all), default je 100

`db_statfile`

soubor, do kterého budou vypisovány statistiky, toto nastavení je použito, pokud není uvedeno odpovídající dbX_statfile.

Konfigurace aplikace může obsahovat libovolné další volby, které používá aplikace samotná. Na chování masseru nemají vliv.

4 Syntaxe

4.1 Stránky (.pg)

Jsou hlavní součástí aplikace a většina ostatních souborů se vztahuje právě k těmto souborům. Podle nastavení proměnné `PG` masser vyhledá patřičný soubor, který vykoná, podle nastavení `path` z konfigurace masseru a aplikace. Pokouší se najít soubor podle obsahu `PG` s příponou .pg nejdříve podle konfigurace aplikace a poté masseru samotného takto:

- adresář / aplikace / `pages` / soubor
- adresář / aplikace / soubor
- adresář / `pages` / soubor
- adresář / soubor

Základní složkou syntaxe těchto souborů jsou řídící tagy <? a ?>. Informace umístěné mezi těmito tagy program dále při kompilaci zpracovává, zbytek souboru vytiskne beze změny na výstup.

Dále je možné použít varianty těchto tagů, všechny jsou zmíněny v následujících podsekcích.

Kód vložený do stránky je lokální v tom smyslu, že například funkce zapsaná ve stránce je neviditelná z jiných stránek v téže nebo jiné aplikaci, to samé platí pro proměnné (kromě globálních).

Příklad - aplikace APP, soubor page1.pg:

```
...
    <?
        sub my_func { print 'page1'; }

        my_func();
    ?>
...
```

Příklad - aplikace APP, soubor page2.pg:

```
...
    <?
        sub my_func { print 'page2'; }

        my_func();
    ?>
...
```

Ve stránce page1 vytiskne "page1", ve stránce page "page2".

4.1.1 Kód <? ... ?>

Text mezi těmito dvěma tagy je považován za kód perlu a prakticky beze změny je proveden. Programátor může počítat s těmito globálními proměnnými:

- **%g** - globální hash k volnému použití
- **$app** - aplikace
- **$lang** - aktuální jazyk
- **$sch** - schéma
- **$sid** - Session ID
- **$query** - objekt CGI[4]

Dále je možné použít tyto zkratky, které budou nahrazeny odpovídajícím textem:

- `PAR(id)` - hodnota z parametrizace aktuální aplikace
- `TXT(id)` - text podle aktuálního jazyka
- `G(id)` - je nahrazeno výrazem `$g{'id'}`

4.1.2 Vložení hodnoty <?= ... ?>

Tato varianta se používá pro vložení hodnoty na dané místo stránky v případě, že se neprovádí žádná další akce. Text <?=$variable?> je ekvivalentní <? print $variable; ?>.

Příklad:

```
...
    <h1> <?='TXT(app_title)'?> </h1>
...
```

[4]Ve většině případů není nutné tento objekt používat, protože k dispozici jsou všechny funkce modulu CGI přímo, viz `use CGI qw(:all)`.

4.1.3 Poznámka <?!---- ... ----?>

Používá se pro označení poznámky. Text mezi těmito dvěma tagy nebude zpracován ani vytištěn do výsledného souboru.[5]

4.1.4 Vložení kódu <?!include ...?>

Syntaxe: <?!include filename [def1,[def2,...]]?>

Program se soubor daného jména pokouší najít obdobně jako v případě stránky (.pg) takto:

- adresář / aplikace / `include` / soubor
- adresář / aplikace / soubor
- adresář / `include` / soubor
- adresář / soubor

Soubor je vložen přímo do kódu, takže syntaxe souboru je stejná jako u stránek, je možné použít všechny speciální příkazy.[6] Definice defX lze použít pro řízení skutečně zahrnutých částí obsahu .inc pomocí if, else a end konstrukce.

Příklad:

```
Soubor test.pg:
  <?!include table print?>
Soubor table.inc:
  <?!use table?>
  <?!javascript table?>
  <?:if print?>
    <?!css table_print?>
  <?:else?>
    <?!css table_screen?>
  <?:end?>
```

4.1.5 Balíček <?!package ...?>

Syntaxe: <?!package filename [def1,[def2,...]]?>

Program se soubor jména [filename].pkg pokouší najít obdobně jako v případě stránky (.pg) takto:

- adresář / aplikace / `package` / soubor
- adresář / aplikace / soubor
- adresář / `package` / soubor

[5] Narozdíl od poznámky v HTML (<!---- ... ---->), která je do výsledné stránky přenesena.

[6] Systém masser v současné verzi nekontroluje rekurzivní vkládání pomocí direktivy `include`, takové použití vede k zacyklení programu!

- adresář / soubor

Má prakticky stejnou funkci, jako *include*, ale předpokládá se spíše využití pro "zabalení" více modulů do jednoho funkčního celku (například JavaScript, CSS a kód dohromady).

Příklad:

```
<?!include button.inc?>
<?!javascript button?>
<?!css button?>
```

Definice defX lze použít pro řízení skutečně zahrnutých částí obsahu .pkg pomocí if, else a end konstrukce.

Příklad:

```
Soubor test.pg:
  <?!package table print?>
Soubor table.pkg:
  <?!use table?>
  <?!javascript table?>
  <?:if print?>
    <?!css table_print?>
  <?:else?>
    <?!css table_screen?>
  <?:end?>
```

4.1.6 Modul <?!use ...?>

Syntaxe: <?!use filename

Program se soubor jména [filename].pm pokouší najít obdobně jako v případě stránky (.pg) takto:

- adresář / aplikace / `modules` / soubor
- adresář / aplikace / soubor
- adresář / `modules` / soubor
- adresář / soubor

Tato direktiva slouží k vložení instrukce pro zahrnutí modulu do kódu stránky. Může být umístěna kdekoliv, je zpracována při kompilaci. Modul pro využití v masseru se prakticky nijak neliší od modulu v jazyce perl, ale lze v něm využít dalších možností - podrobnosti jsou dále v příslušné kapitole.

4.1.7 Začátek stránky <?!page_start ...?>

Syntaxe: <?!page_start [key1=>value1, key2=>value2, ...]?>

Vytiskne hlavičku stránky (HTTP hlavičku a začátek stránky po tag BODY včetně). Jako parametry lze použít jakékoliv parametry přípustné ve funkci modulu CGI `start_html()` a kromě nich také:

- **title** - titulek stránky (default "Page Title")
- **do_form** - pokud je nastaveno a je nastaveno i do_start, pak vytiskne začátek formuláře s odpovídajícími skrytými proměnnými masseru (default je nastaveno)
- **do_start** - pokud je nastaveno, tiskne začátek stránky (pomocí start_html()) (default je nastaveno)
- **onload** - pokud je uvedeno, je použito jako onload v sekci BODY dokumentu (default je nic)
- **content** - content-type dokumentu, default je content-type z parametrizace masseru
- **cookie** - nastavení cookies, default je nic, nemá význam, pokud není nastaveno do_start

[7]

Příklad:

```
...
    <?!page_start title=>'TXT(app_title)', bgcolor=>'#ffffff'?>
...
```

Tato funkce navíc zaručuje vložení odpovídajícího JavaScript kódu a definice CSS, pokud ji nepoužijete, tagy `javascript` a `css` nepracují správně. Funkce navíc vytiskne začátek formuláře a povinné skryté proměnné - APP, PG, OP, LANG, SCH a SID.

4.1.8 Konec stránky <?!page_end?>

Syntaxe: <?!page_end?>

Příkaz vytiskne konec stránky - ukončí formulář, BODY a HTML tagy. Měl by být použit na konci stránky.

4.1.9 Vložení JavaScriptu <?!javascript ...?>

Syntaxe: <?!javascript [ext:]filename?>

Umožňuje vložení JavaScript kódu do hlavičky stránky (mezi začátek a konec HEAD tagu). Pokud použijete předponu `ext:`, program vloží pouze odkaz na daný soubor, jinak se pokouší najít soubor jména **[filename].js** takto:

- adresář / aplikace / `javascript` / soubor
- adresář / aplikace / soubor
- adresář / `javascript` / soubor
- adresář / soubor

Jeho obsah přečte a vloží do stránky.

V souborech s JavaScriptem je možno použít tyto konstrukce:[8]

[7] Tento příkaz musí být první výstup ve stránce, protože tiskne hlavičku HTTP protokolu, v jiném případě je výsledkem 500 Internal Error a v logu HTTP serveru hlášení typu "malformed headers".

[8] Nelze použít v souborech vložených s předponou `ext:` - nejsou zpracovávány.

- `PAR([id])` je nahrazeno obsahem $conf{'id'} (konfigurace)
- `TXT([id])` je nahrazeno obsahem $txt{'id'} (text)
- `GLB([id])` je nahrazeno obsahem $g{'id'} (globální hash)

Příklad - soubor example1.js:

```
alert('TXT(error_required)');
var num_items = PAR(max_results);
for (var i=0; i<G(nusers); i++) { alert(i); }
```

Příklad - stránka:

```
<?!javascript example1?>
```

Tuto direktivu je možné použít kdekoliv ve stránce nebo vložených souborech, je zpracovávána při kompilaci.

4.1.10 Vložení CSS <?!css ...?>

Syntaxe: <?!css [ext:]filename?>

Umožňuje vložení CSS kódu do hlavičky stránky (mezi začátek a konec HEAD tagu). Pokud použijete předponu `ext:`[9] , program vloží pouze odkaz na daný soubor, jinak se pokouší najít soubor jména **[filename].css** takto (zde je drobná změna oproti vyhledávání jinýc souborů aplikací masser - bere se v úvahu proměnná $schema kvůli tomu, aby bylo možné aplikacím měnit vzhled):

- adresář / aplikace / `css` / `param('SCH')` / soubor
- adresář / aplikace / `css` / soubor
- adresář / aplikace / soubor
- adresář / `css` / `param('SCH')` / soubor
- adresář / `css` / soubor
- adresář / soubor

Jeho obsah přečte a vloží do stránky.

V souborech s CSS je možno použít tyto konstrukce:[10]

- `PAR([id])` je nahrazeno obsahem $conf{'id'} (konfigurace)
- `TXT([id])` je nahrazeno obsahem $txt{'id'} (text)
- `GLB([id])` je nahrazeno obsahem $g{'id'} (globální hash)

[9] Vzhledem k omezení modulu CGI, který program využívá, není možné vložit více než jeden soubor s příponou `ext:` do stránky. Pokud jich použijete více, bude vložen první z nich.

[10] Nelze použít v souborech vložených s předponou `ext:` - nejsou zpracovávány.

Příklad - soubor example2.css:

```
BODY { background-image: url(GLB(img_dir)/backg1.gif); }
```

Příklad - stránka:

```
<?!css example2?>
```

Tuto direktivu je možné použít kdekoliv ve stránce nebo vložených souborech, je zpracovávána při kompilaci.

4.1.11 Příprava SQL příkazu <?!statement ...?>

Syntaxe: <?!statement [dbnum:]filename?>

Připraví SQL příkaz v databázi (prepare_statement). Pomocí předpony dbnum (číslo) lze určit, ve kterém připojení to má být, pokud program využívá více než jedno připojení k databázi. Pokud dbnum není uvedeno, program předpokládá připojení číslo 1.

Program hledá soubor [**filename**]**.sql** takto:

- adresář / aplikace / `statement` / soubor
- adresář / aplikace / soubor
- adresář / `statement` / soubor
- adresář / soubor

Lze použít tyto konstrukce pro databázově závislé zpracování:

- _DATE_ vloží konstrukci pro datum
- _DATETIME_ vloží konstrukci pro datum a čas

Příklad - soubor MY_STAT.sql:

```
SELECT * FROM cats_users WHERE pernr = ? AND last_chgp = _DATETIME_
```

Příklad - stránka:

```
    <?!statement MY_STAT?>
    <?!statement 2:OTHER_STAT?>
...
    <?
        my $result = $db->perform_statement('MY_STAT');
    ?>
```

Připravené SQL dotazy jsou globální pro celou aplikaci, ale neovlivňují ostatní aplikace. Pokud není masser v debug módu, je každý příkaz připraven pouze jednou, i když je vložen na více stránkách. Obdobně jako JavaScript a CSS je možné příkaz zapsat na jakémkoliv místě stránky nebo vloženého souboru, informace je zpracována při kompilaci.

4.1.12 Volání funkce <?!do ...?>

Syntaxe: <?!do filename [parameters]?>

Na daném místě provede - na rozdíl od include nevloží celý kód - funkci, jejíž kód je v souboru **filename**, který masser hledá takto:

- adresář / aplikace / `function` / soubor
- adresář / aplikace / soubor
- adresář / `function` / soubor
- adresář / soubor

Obsahem souboru musí být perlovský kód, lze použít stejné konstrukce, jako ve stránce mezi tagy <? a ?> - tedy PAR, TXT a G a stejné globální proměnné.

Funkce je globální pro celou aplikaci - mezi aplikacemi je neviditelná.

Příklad - soubor func.pl:

```
my (undef,$par1,$par2) = @_;
# POZOR! První parametr funkce je jméno aplikace!

if ($par1 > $par2) {
    G(to_print) = 'TXT(is_greater)';
} elsif ($par1 < par2) {
    G(to_print) = 'TXT(is_lesser)';
} else {
    G(to_print) = 'TXT(is_equal)';
}
```

Příklad - stránka:

```
...
    <?!do func.pl 5,8?>
    <?=$g{'to_print'}?>
...
```

4.2 Akce (.ac)

Akce jsou vyvolávány na základě obsahu proměnné `OP`, kde může být prázdný řetězec (žádná akce), identifikátor akce, nebo seznam identifikátorů akce oddělený čárkami.

Akce jsou prováděny ***před*** stránkami.

Program hledá soubor podle jména v `OP` s příponou .ac takto:

- adresář / aplikace / `actions` / soubor
- adresář / aplikace / soubor

- adresář / `actions` / soubor
- adresář / soubor

Syntaxe těchto souborů je stejná, jako u stránek, praktický význam ale má pouze perlovský kód, a direktivy `include` a `statement`. Nic vám ale nebrání použít cokoliv ze zbývajících možností.

Lze použít stejné konstrukce (TXT, PAR, G) a počítat se stejnými globálními proměnnými.

5 Jak psát...

5.1 ...stránky

Ve stránkách (.pg) je možné použít řídící tagy uvedené v předchozí kapitole. Kromě nich je možné v kódu použít tyto proměnné:

%g

Globální hash, využívá se především pro přenos údajů mezi akcí a stránkou. Pozor, jeho platnost je omezena jedním cyklem, tedy těsně před započetím vyřizování jednoho požadavku (zaslaného jako požadavek z prohlížeče) je jeho obsah prázdný a po dokončení požadavku je vyčištěn.

Nelze jej tedy použít pro předání údajů mezi dvěma požadavky, k tomu je možné použít pouze prostředků HTML a/nebo HTTP, tedy skrytých proměnných ve formulářích nebo cookies! Vyjímkou je hash %p popsaný v následujícím bodě

Pro zápis je také možné použít formu G(item), tedy `print G(name);` a `print $g{'name'};` jsou ekvivalentní.

%p

Globální hash, jehož podstatnou odlišností od %g je to, že jeho obsah zůstává zachován i mezi požadavky. Využití je především v nastavení jeho položek v akci *init* nebo pro předání hodnoty mezi dvěma cykly aplikace (dvěma různými požadavky).

Pozor, tato vlastnost je ovšem výrazně omezena v případě, že v systému je spuštěn více než jeden masser, což technologie FastCGI umožňuje. Nelze pak zaručit, že druhý požadavek je předán stejné instanci masseru. ***Hash je persistentní pouze v rámci jedné instance programu masser!***

Pozor, používejte tuto proměnnou s rozvahou, není vhodná pro ukládání většího množství dat.

Nelze použít pro předání informací mezi dvěma aplikacemi ani v rámci jedné instance masseru, každá aplikace má vlastní instanci této proměnné.

%txt

Proměnná obsahující načtené texty ze souboru s jazykovými variantami dle aktuálně vybraného jazyka (proměnná LANG). *Pouze pro čtení.*

%conf

Proměnná obsahující konfiguraci aplikace získanou z <app>.conf. *Pouze pro čtení.*

$app

Proměnná obsahující aktuální aplikaci (stejně jako param('APP')).

$pg

Proměnná obsahující aktuálně prováděnou stránku (stejně jako param('PG')).

$lang

Proměnná obsahující aktuálně vybraný jazyk (stejně jako param('LANG')).

$db, $db1, $db2, $db3, $db4, $db5

Proměnné obsahující odkaz na objekt typu DeltaX::Database číslované dle konfigurace aplikace. Poznámka: $db = $db1.

V kódu je také možné použít tyto funkce:

`error, warn, info, debug, trace`

Funkce modulu DeltaX::Trace. Vypisují informace do logu aplikace.

`break_page`

Funkce umožňující změnit stránku, kterou masser vykoná (tedy obsah $pg a param('PG'), které toto řídí, prostým přepisem obsahu některé z těchto proměnných toho nelze docílit).

Jediným parametrem této funkce je nová stránka (řetězec). Masser se pak zachová takto:

- pokud je nově nastavená stránka shodná s aktuální, nic se neděje
- pokud není, je změněn obsah proměnných řídících zobrazenou stránku
- *po dokončení aktuální stránky* je provedena nastavená stránka

Obvykle se používá například pro předání řízení jiné stránce (vypisující chybu) v případě, že selže kontrola oprávnění a podobně.

Pamatujte na to, že to, co již bylo vytištěno (například pomocí page_start), tímto již nelze vzít zpět.

Další "speciality" využitelné v kódu stránky:

- podmíněné vložení kódu je možné takto:

```
...
<? if ($neco) { ?>
    <?!include file1.inc>
<? } else { ?>
    <?!include file2.inc>
<? } ?>
```

Ale pozor, tímto postupem je při kompilaci vložen kód obou souborů. Řízení kompilace obsahem proměnné zjistitelným až při běhu není bohužel možné...

- předčasné ukončení stránky je možné příkazy **return**, **die** (bude vypsáno do logu aplikace jako chyba), nebo pomocí skoku na konec kódu **goto END_THIS_PAGE;**. Ano, finální kód, do něhož masser překládá, je anonymní funkce ;-).

5.2 ...akce

Pro akce (.ac) platí vše, co bylo řečeno v předcházející kapitole pro stránky. Zde bych jen chtěl varovat před použitím jakýchkoliv tiskových výstupů v akcích. Rozdíl mezi akcemi a stránkami je spíše logický a to proto, aby bylo možné logicky oddělit něco vykonávaného na popud uživatele a samotný grafický výstup. Ne vždy je to samozřejmě možné, ale čím více to bude odděleno, tím lépe.

5.3 ...moduly

Moduly pro masser se v podstatě nijak neliší od modulů pro jazyk perl, až na dvě vyjímky: Jednak je možné je umístit mimo adresáře, kde by je sám perl mohl nalézt automaticky (tedy do adresářů aplikace), a dále je možné v nich využít některé proměnné poskytované masserem (jde o *%conf, %txt, %g, %p, $app, $pg, $lang, $db, $dbX*) takto:

```
package xy;
...
use vars qw/%txt %conf $app/;
import main;
...
```

5.4 ...JavaScript a CSS

Zde jen k možnosti použít variantu GLB(...) pro vložení informací z globální proměnné %g: Je třeba si uvědomit, že toto nahrazování je provedeno ve funkci **page_start**, takže případná nastavené v této funkci je nutné provést před jejím provedením (tedy patřičné volání funkce, include, modulu nebo provedení kódu, který to zařídí, musí v kódu stránky předcházet tagu s page_start).