Šablonovací systém htmltmpl

Představení šablonovacích systémů

Každý, kdo se zabývá tvorbou webových aplikací dříve nebo později zjistí, že je vhodné oddělovat obsah aplikace od jejího vzhledu. Pokud je webová aplikace napsána jen v jazyce HTML, vystačí si autor s kaskádovými styly. Tento postup je pro jednodušší aplikace dostačující. Jestliže však plánujete programovat složitější projekt, jehož stránky nejsou pouze statické, ale jejich obsah se může měnit třeba podle obsahu databáze, přistupuje se často k naprogramování projektu v jazyce PHP.

Práce s PHP vypadá tak, že generujete HTML kód spolu s tagy kaskádových stylů. Přestože máte obsah neustále oddělen od vzhledu, nemusí to být přehledné a ani pohodlné vracet se ke zdrojovému kódu a něco v něm měnit, případně dělat jiný design. Bez znalosti PHP jazyka se zde neobejdete. Navíc, pokud na projektu pracuje více lidí, je celá situace ještě složitější. Jedním z řešení, jak si práci usnadnit a zpřehlednit, je použít některý z šablonovacích systémů.

Nyní je správný okamžik, říci si, co to vlastně šablonovací systém je. Jedná se vrstvu, která se při programování vloží mezi PHP a HTML. V praxi to vypadá tak, že místo, abyste v PHP generovali přímo HTML kód, napíšete si pomocí tagů šablonovacího systému šablonu, která bude generovat HTML. Data do šablony dostanete tak, že prostřednictvím PHP předáte požadovaná data šablonovacímu systému ve formě proměnných. Až si tímto způsobem připravíte všechna data, řeknete šablonovacímu systému, aby zhotovil ze šablony HTML kód. Ve vývojovém týmu pak situace vypadá tak, že programátor napíše onen PHP skript a designér obstará šablonu, aniž by se musel zatěžovat tím, co dělá programátor v PHP. Pokud je třeba změnit vzhled nad stejnými daty, stačí měnit pouze šablonu a PHP skript může zůstat beze změn. Jedinou nevýhodou je trocha práce navíc při tvorbě projektu. Ta je ovšem více než kompenzována přehledností a ušetřením práce při různých předělávkách a zásazích.

Šablonovacích systémů existuje mnoho a vybrat si mezi nimi ten správný nemusí být úplně jednoduché. Některé systémy jsou opravdu mocnými nástroji, ale jsou poměrně složité a mají svou syntaxi, kterou se musíte naučit. Rád bych představil šablonovací systém htmltmpl. Není tak mocný, jako někteří jeho konkurenti, ale pro většinu větších (i velkých) projektů s ním určitě vystačíte. Navíc je jeho syntaxe podobná jazyku HTML, což značně usnadňuje naučit se s ním pracovat.

Seznámení s htmltmpl

Šablonovací systém htmltmpl je šířen pod licenci GNU General Public Licence. Byl napsán v roce 2001 Tomášem Stýblem, který se o něj staral až do roku 2007, kdy jej pod svá křídla převzal Jakub Vrána. Myslím si, že právě postava Jakuba Vrána je určitou zárukou kvality tohoto produktu. Původně byl systém htmltmpl určen jen pro jazyk PYTHON. V roce 2004 byl však upraven také pro jazyk PHP.

Jak již bylo zmíněno v úvodu, představuje šablonovací systém nástroj pro oddělení programátorské části webové aplikace od její prezentační části. Právě zde vidím největší výhodu htmltmpl. Ten poskytuje nástroje dostatečně silné, aby mohl designér pomocí šablony vytvořit vzhled, ale zároveň nic víc. Nemá možnosti, jak ovlivňovat chod PHP skriptů programátora, čímž by mohl při nepochopení myšlenek programátora napáchat nějakou škodu. Zkrátka poskytuje přesně to, co je nezbytně nutné, ale nic víc. Přistupme nyní k samotnému popisu tohoto šablonovacího systému.

Syntaxe šablon htmltmpl

Příkazy

Příkazy šablonovacího systému htmltmpl vypadají podobně jako značky jazyka HTML. Lze je zapisovat ve dvou podobách:

- `<TMPL_VAR>`
- `<!-- TMPL_VAR -->`

Pokud se rozhodnete používat delší formu zápisu, je nutné striktně psát za otevírací `<!--` a před uzavírací `-->` mezeru. Dále budeme z důvodu úspory místa používat kratší typ zápisu.

Všechny příkazy kromě `TMPL_VAR` musí začínat na novém řádku.

Pokud se mezi začátkem řádku a příkazem nacházejí pouze bíle znaky, tj. mezery a tabulátory, jsou odstraněny. Obdobně systém při generování HTML kódu odstraňuje prázdné řádky. Pokud si přejete prázdné řádky zachovat, třeba kvůli formátování, stačí na řádek udělat mezeru nebo tabulátor.

Příkazy nemusí dodržovat pravidla HTML, aby bylo HTML validní. Například následující použití příkazu TMPL_VAR vytvoří validní HTML kód:

```
<img src="<TMPL_VAR obrazek>" />
```

Nerozpoznané příkazy `TMPL_*` způsobí při kompilaci šablony chybu.

Psaní příkazů velkými písmeny zlepšuje čitelnost šablon. Lze tak na první pohled odlišit značky šablonovacího systému od značek XHTML, které musí být napsány písmeny malými.

Parametry příkazů

Stejně jako příkazy, lze i parametry zapisovat dvěma způsoby:

- S uvozovkami: `<TMPL_VAR moje_promenna ESCAPE="HTML">`
- Bez uvozovek: `<TMPL_VAR moje_promenna ESCAPE=HTML>`

Před ani za znakem = nesmí být mezera.

Celý parametr musí být velkými písmeny. Předdefinované hodnoty parametrů je nutné psát také velkými písmeny, jak vidíme u parametru `ESCAPE` a jeho hodnoty `HTML`.

Jména parametrů a jejich hodnot mohou obsahovat pouze znaky anglické abecedy a některé další znaky: pomlčku, tečku, podtržítko, dvojtečku, lomítko a zpětné lomítko. Nesmí však obsahovat mezery.

Jména

Rozlišujeme tři druhy jmen:

- Jména proměnných
- Jména cyklů
- Jména souborů a vkládaných šablon

Jména cyklů a proměnných mohou obsahovat pouze znaky anglické abecedy, podtržítka a pomlčky. Kromě toho musí dodržovat následující pravidla:

- Písmena použitá ve jméně proměnné musí být malá: `moje_promenna`
- Jméno cyklu musí začínat velkým písmenem a ostatní písmena být malá: `Muj_cyklus`

Pro názvy souborů v příkaze `TMPL_INCLUDE` platí stejná pravidla jako pro hodnoty parametrů. Mohou tedy obsahovat pouze znaky anglické abecedy a některé další znaky: pomlčku, tečku, podtržítko, dvojtečku, lomítko a zpětné lomítko. Nesmí však obsahovat mezery.

Minimální délka jména proměnné nebo cyklu je jeden znak. Jména proměnných a cyklů lze definovat dvěma způsoby:

- Jako slovo: `<TMPL_VAR moje_promenna>`
- Pomocí parametru `NAME`: `<TMPL_VAR NAME="moje_promenna">`

Přehled příkazů a jejich parametrů

Příkaz	Možné parametry
TMPL_INCLUDE	NAME
TMPL_VAR	NAME, ESCAPE, GLOBAL
TMPL_IF	NAME, GLOBAL
TMPL_UNLESS	NAME, GLOBAL
TMPL_ELSE	
/TMPL_IF	
/TMPL_UNLESS	
TMPL_LOOP	NAME
/TMPL_LOOP	
TMPL_BOUNDARY	
TMPL_STATIC	NAME, ESCAPE

Komentáře v šabloně

Komentáře se píší ve tvaru: `### nejaky komentar`

Všechno, co se nachází za „`###` “ je odstraněno při kompilaci šablony. Použití mezery za třetím hash křížem je nezbytné. Používání komentářů lze zakázat parametrem `comments` (viz. dále).

Příklady komentářů:

```
<TMPL_VAR jmeno>    ### komentar jedna
<TMPL_VAR prijmeni> ### komentar dva
```

Příkazy

Vkládání šablon

Příkaz `<TMPL_INCLUDE>` vloží šablonu do aktuální šablony přesně na místo, kde se příkaz nachází. Obsah vložené šablony je používán úplně stejně, jako by byl napsán přímo v aktuální šabloně.

Všechny vkládané šablony se musí nacházet v adresáři pojmenovaném *inc*, který se nachází ve stejném adresáři jako hlavní šablona. Při vkládání šablony je nutné použít jen její jméno bez cesty.

Příklady:

Opět máme dvě možnosti použití příkazu. V uvedeném příkladu musí být šablona *hlavicka.tmpl* umístěna v adresáři *inc*, který je podadresářem adresáře, ve které je uložena šablona, v níž se nachází uvedený příkaz.

```
<TMPL_INCLUDE hlavicka.tmpl>
<TMPL_INCLUDE NAME="hlavicka.tmpl">
```

Proměnné

Pro vkládání proměnných slouží příkaz `<TMPL_VAR>`

Escapování proměnných

Všechny proměnné jsou automaticky HTML escapovány. Toto chování lze zakázat nastavením parametru `html_escape` konstruktoru `TemplateProcessor` na `false` (viz. dále). Escapování lze také nastavovat libovolně pro každou proměnnou pomocí parametru `ESCAPE` nezávisle na automatickém nastavení. Parametr `ESCAPE` může nabývat čtyř hodnot:

- `HTML` zapíná HTML escapování

- `URL` zapíná URL + HTML escapování (nejprve se escapuje URL a pak HTML)
- `WAP` zapíná WML escapování (pro WAPové proměnné: „$“ => „$$“)
- `NONE` vypíná escapování

Parametr `ESCAPE` je možné používat u příkazů `<TMPL_VAR>` a `<TMPL_STATIC>`. U obou dvou je implicitně zapnuto HTML escapování (jak je uvedeno výše).

Globální hledání proměnných

Všechny proměnné, které se nacházejí uvnitř cyklů, jsou lokální v rámci tohoto cyklu. Pokud tedy chcete použít uvnitř cyklu proměnnou „globální“ (ležící mimo cyklus), musíte buď nastavit parametr `global_vars` konstruktoru `TemplateProcessor` na `true` nebo použít u dané proměnné parametr `GLOBAL`, který má vyšší prioritu než `global_vars`.

Příklady:

```
<TMPL_VAR jmeno>
<TMPL_VAR NAME="mesto">
<TMPL_VAR NAME="text1" ESCAPE="HTML">
<TMPL_VAR NAME="text2" ESCAPE="NONE" GLOBAL="0">
<TMPL_VAR adresa GLOBAL="1">
<!-- TMPL_VAR test ESCAPE=URL -->
```

Podmínky

S podmínkami se pracuje pomocí následujících příkazů:

- `TMPL_IF` značí začátek IF bloku
- `/TMPL_IF` značí konec IF bloku resp. ELSE blok
- `TMPL_ELSE` značí začátek ELSE bloku
- `TMPL_UNLESS` značí začátek UNLESS bloku
- `/TMPL_UNLESS` značí konec UNLESS bloku resp. ELSE bloku

Značky začínají a ukončují blok, který je vložen do šablony, pokud je splněna podmínka bloku. V podmínkách lze používat pouze jména proměnných nebo jména bloků. Podmínkové bloky mohou obsahovat další hnízděné podmínkové bloky.

Pokud je v podmínce použito jméno cyklu, pak je podmínka splněna, pokud je obsah cyklu takový, že cyklus proběhne alespoň jednou.

Příklady:

```
<TMPL_IF podminka1>
    ### Tento blok bude vložen do výstupní šablony, pokud je podminka1 true.
<TMPL_ELSE>
```

```
    ### Tento blok bude vložen do výstupní šablony, pokud je podminka1 false.
</TMPL_IF>

<TMPL_UNLESS podminka2>
    ### Tento blok bude vložen do výstupní šablony, pokud je podminka2 false.
<TMPL_ELSE>
    ### Tento blok bude vložen do výstupní šablony, pokud je podminka2 true.
</TMPL_UNLESS>
```

Cykly

S cykly se pracuje pomocí dvou příkazů:

- `TMPL_LOOP` značí začátek cyklu
- `/TMPL_LOOP` značí konec cyklu

Cykly mohou obsahovat hnízděné cykly. Každý cyklus má svůj vlastní jmenný prostor pro proměnné. Za normálních podmínek nemohou proměnné uvnitř cyklu odkazovat na proměnné vně cyklu. Takového chování lze docílit nastavením parametru `global_vars` konstruktoru `TemplateProcessor` na `true` nebo použitím parametru `GLOBAL` v příkaze `TMPL_VAR`.

Pokud použijete jméno cyklu v příkaze `TMPL_VAR`, získáte celkový počet průchodů cyklu.

Příklady:

```
<TMPL_LOOP Muj_cyklus>
    ### Obsah tohoto bloku je vložen do aktuální šablony při každém průchodu cyklem
    <TMPL_VAR nejaka_promenna>    ### Lokální proměnná v cyklu
</TMPL_LOOP>
```

Vestavěné proměnné cyklu

Šablonovací systém pro každý cyklus automaticky vytváří několik užitečných proměnných. Ty mohou být používány uvnitř cyklu jako běžné proměnné. Jejich jména začínají a končí dvěma podtržítky. Své vlastní proměnné nelze pojmenovávat tímto způsobem, jinak kompilace šablony skončí chybou. Hodnoty těchto proměnných jsou vždy integery (případně true=1 a false=0).

Seznam vestavěných cyklových proměnných:

Název proměnné	Popis
`__FIRST__`	Proměnné má hodnotu `true`, pokud je aktuální průchod cyklu první.
`__LAST__`	Proměnné má hodnotu `true`, pokud je aktuální průchod cyklu poslední.
`__INNER__`	Proměnné má hodnotu `true`, pokud není aktuální průchod cyklu první ani poslední.
`__ODD__`	Proměnná má hodnotu `true`, pokud je aktuální průchod cyklu lichý (tj. 1, 3, 5, …).
`__PASS__`	Hodnotou proměnné je pořadové číslo aktuálního cyklu. Číslování začíná od jedné (tedy první průchod má číslo 1, druhý 2, …).
`__PASSTOTAL__`	Hodnotou proměnné je celkový počet průchodů cyklu.
`__EVERY__x`	`x` musí být integer. Hodnota této proměnné je `true`, pokud je `x` modulo číslo aktuálního cyklu rovno nule. V prvním a posledním průchodu cyklu její hodnota není nikdy `true`.

Vícedílné šablony

Lze je vytvořit použitím příkazu `TMPL_BOUNDARY`. Tento příkaz nemá žádné parametry.

Vícedílné šablony se používají, pokud potřebujete překompilovat část šablony dříve, než zbytek šablony. To se typicky hodí ve webových aplikacích, kde tak můžete poslat prohlížeči třeba hlavičku dokumentu ještě dříve, než obdržíte například data z databáze.

Každá část vícedílné šablony musí být validní šablona. Nelze tedy vkládat hranice uvnitř cyklů nebo podmínkových bloků.

Rozdělení šablony na jednotlivé části je provedeno po dokončení vkládání šablon do aktuální šablony. Nedoporučuje se vkládat hranice do vkládaných šablon.

Statické proměnné

Statické proměnné jsou reprezentovány příkazem `TMPL_STATIC`. Jsou podobné jako normální proměnné `TMPL_VAR`, až na to, že se nikdy nemění. Hodí se, pokud se některé řetězce vyskytují v šabloně několikrát nebo v různých variantách šablony.

Částečně se podobají příkazu `TMPL_INCLUDE`, ale jsou vykonávány mnohem rychleji a používají se typicky pro krátké řetězce.

Statické proměnné jsou do šablony doplněny ještě před samotnou kompilací šablony a při jejich změně se šablona znovu překompiluje.

Příklady:

```
<TMPL_STATIC nejaka_url_adresa ESCAPE="URL">
<TMPL_STATIC NAME="nejaka_url_adresa">
```

Jádro šablonovacího systému htmltmpl

Samotný šablonovací systém se skládá z několika tříd. Před popisem jednotlivých tříd a jejich metod se podívejme na stručný přehled:

- `TemplateCompiler` - Předzpracování, parsování, tokenizace a kompilace šablony.
  - `TemplateCompiler` Konstruktor třídy.
  - `compile` Zkompiluje šablonu ze souboru.
  - `compile_string` Zkompiluje šablonu z řetězce.
- `Template` – Tato třída představuje zkompilovanou šablonu.
  - `is_uptodate` Zkontroluje, zda je kompilace šablony aktuální.
- `TemplateError` – Reprezentuje kritické výjimky - chybná syntaxe nebo běhová chyba.
- `TemplateManager` – Třída řídí kompilaci a předkompilaci šablon.
  - `TemplateManager` Konstruktor třídy.
  - `prepare` Předzpracování, parsování, tokenizace a kompilace šablony.
  - `static_data` Definice statických proměnných šablony.
  - `update` Znovu kompiluje už zkompilovanou šablonu.
  - `watch_file` Kontroluje předložený soubor, zda nebyl změněn.
- `TemplateProcessor` – Naplní šablonu daty a vykoná ji.
  - `TemplateProcessor` Konstruktor třídy.
  - `process` Vykoná zkompilovanou šablonu. Výsledek vrací jako řetězec.
  - `set` Nastavuje hodnotu proměnné nebo cyklu.
  - `reset` Vymaže všechna data šablony.

Třída: `TemplateCompiler`

Předzpracování, parsování, tokenizace a kompilace šablony.

Tato třída parsuje šablonu provádí její kompilaci. Zkompilovaná forma šablony je instancí třídy `Template` a je vstupem pro `TemplateProcessor`, který předanou šablonu naplní daty a vykoná ji.

Tuto třídu lze použít přímo, pokud potřebujete zkompilovat šablonu z textového řetězce. Pokud

však máte šablonu uloženu v souboru je lepší použít třídu `TemplateManager`, která představuje nástavbu této třídy a umožňuje uložení zkompilované šablony na disk.

Metoda: `TemplateCompiler()`

Konstruktor třídy `TemplateCompiler`.

Parametry:

```
TemplateCompiler(include=1,max_include=5,comments=1,gettext=0,debug=0)
```

- `include` Povoluje/Zakazuje vkládání šablon.
- `max_include` Maximální hloubka hnízděných vkládání šablon.
- `comments` Povoluje/Zakazuje používání komentářů v šablonách.
- `gettext` Povoluje/Zakazuje podporu gettextu.
- `debug` Povoluje/Zakazuje vypisování ladících zpráv

Metoda: `compile()`

Zkompiluje šablonu ze souboru.

Návratová hodnota: Zkompilovaná šablona. Je instancí třídy `Template`.

Parametry:

```
compile(file)
```

- `file` Jméno souboru se šablonou. Parametr je podrobněji popsán u metody `prepare()` třídy `TemplateManager`.

Metoda: `compile_string()`

Zkompiluje šablonu z řetězce.

Tato metoda zkompiluje šablonu z řetězce. Šablona nemůže být vkládána do jiných šablon (příkaz `TMPL_INCLUDE` vrací varování).

Návratová hodnota: Zkompilovaná šablona. Je instancí třídy `Template`.

Parametry:

```
compile(data)
```

- `data` Textový řetězec obsahující data šablony.

Třída: `Template`

Tato třída představuje zkompilovanou šablonu.

Třída `Template` poskytuje úložiště a metody pro zkompilované šablony a asociovaná metadata. Nikdy by jste neměli vytvářet instance této třídy přímo. K vytváření instancí vždy používejte třídu `TemplateManager` nebo `TemplateCompiler`.

Metoda: `is_uptodate()`

Zkontroluje, zda je kompilace šablony aktuální.

Pokud je kompilace šablony aktuální, tak metoda vrací `true`. Pokud byl zdrojový kód šablony od předchozí kompilace změněn, vrací metoda `false`. Vyhodnocení probíhá porovnáním časů poslední změny. Do porovnání jsou zahrnuty také vložené šablony.

Návratová hodnota: Pokud je šablona aktuální `true`, jinak `false`.

Parametry:

```
is_uptodate(compile_params=None)
```

- `compile_params` Pouze pro interní použití. Nepoužívejte tento parametr.

Třída: `TemplateError`

Reprezentuje kritické výjimky - chybná syntaxe nebo běhovou chybou.

Tyto výjimky jsou vyvolány, pokud dojde k běhové chybě programu nebo pokud se vyskytne v šabloně špatná syntaxe. Má jeden řetězcový parametr, ve kterém je vždy popis vzniklé chyby.

Jsou odchytávány všechny potenciální I/O chyby a vyvolány výjimky `TemplateError`. To znamená, že můžete odchytávat výjimky `TemplateError`, když je možné, že soubor šablony nebude dostupný.

Tato výjimka může být vyvolána konstruktorem nebo metodou libovolné třídy.

Pokud je výjimka vyvolána, nelze již s instancí šablony dále pracovat.

Třída: `TemplateManager`

Třída řídí kompilaci a předkompilaci šablon.

Tuto třídu můžete použít vždy, pokud pracujete se šablonami, které jsou uloženy jako soubory. Třída umí vytvořit zkompilovanou šablonu a řídit její kompilaci. Sama si také hlídá, aby byly kompilace šablon aktuální.

Metoda: `TemplateManager()`

Konstruktor třídy.

Parametry:

```
TemplateManager(include=1, max_include=5, precompile=1, comments=1,
gettext=0, debug=0)
```

- `include` Povoluje/Zakazuje vkládání šablon (příkaz `TMPL_INCLUDE`). Zakázání může trochu zvýšit rychlost. Implicitně je povoleno.
- `max_include` Maximální hloubka vkládání hnízděných šablon (příkaz `TMPL_INCLUDE`). Implicitní hodnota je 5. Tento parametr zabraňuje nekonečným rekurzivním vkládáním šablon.

- `precompile` Zapíná/Vypíná předkompilaci šablon a používání předkompilovaných šablon. Implicitně je zapnuta. Předkompilované šablony se ukládají do stejného adresáře, jako je uložena uvažovaná šablona. Je třeba mít právo zapisovat do tohoto adresáře. Předkompilace poskytuje podstatné zrychlení, protože se ušetří kompilování šablon, které se od poslední předkompilace nezměnily. Zrychlení je znatelné hlavně při vkládání šablon. Při kompilaci je vždy provedena kontrola, které předkompilace nejsou aktuální a jsou pak znovu zkompilovány. Kontroluje se také aktuálnost vkládaných šablon. K překompilování všech šablon dojde automaticky při nahrání nové verze htmltmpl. Pokud se nepodaří předkompilovanou šablonu uložit, je vyvolána výjimka `TemplateError`.
- `comments` Povoluje/Zakazuje používání komentářů v šablonách. Implicitně jsou komentáře povoleny. Zakázání může přinést mírné zrychlení.
- `gettext` Zapíná/Vypíná podporu gettextu.
- `debug` Zapíná/Vypíná vypisování ladících výpisů. Implicitně jsou výpisy vypnuty. Pokud jsou zapnuty, probíhá jejich vypisování na standardní chybový výstup.

Metoda: `prepare()`

Předzpracování, parsování, tokenizace a kompilace šablony.

Pokud je povoleno předkompilování, pokusí se tato metodo načíst předkompilované tvary šablon z adresáře, ve kterém jsou uloženy zdrojové soubory šablon. Pokud se jí to podaří, porovná časy poslední změny předkompilovaných a zdrojových souborů šablon, včetně šablon vložených pomocí příkazu `TMPL_INCLUDE`. Pokud časy nevyhovují, tak jsou šablony znovu zkompilovány.

Pokud není předkompilace šablon povolena, pak tato metoda parsuje a kompiluje šablony.

Návratová hodnota: Zkompilovaná šablona. Je to instance třídy `Template`.

Parametry:

```
prepare(file, force_precompiled=0)
```

- `file` Cesta k souboru, který má být zpracován. Pokud je cesta relativní, hledá metoda soubor v aktuálním adresáři. všechny vkládané soubory šablon musí být umístěny v podadresáři inc aktuálního adresáře.
- `force_precompiled` Použít pouze předkompilované šablony. Tento parametr se hodí, pokud jsou všechny šablony předkompilovány a jsou umístěny v adresáře, který má nastaveno pouze čtecí právo. Pokud nejsou zkompilované šablony nalezeny, je vyvolána výjimka. Implicitně se nekontroluje, zda jsou kompilace šablon aktuální.

Metoda: `static_data()`

Definice statických proměnných šablony.

První parametr je asociativní pole, kde jsou jména proměnných klíči a jejich hodnoty jsou hodnotami pole.

Návratová hodnota: Žádná

Parametry:

```
static_data(static)
```

- `static` Pole párů jméno – hodnota.

Metoda: `update()`

Znovu kompiluje už zkompilovanou šablonu.

Tato metoda znovu zkompiluje soubor šablony. Pokud je povolena předkompilace, je zkompilovaná forma šablony na disku aktualizována.

Návratová hodnota: Překompilovaná aktuální šablona.

Parametry:

```
update(template)
```

- `template` Zkompilovaná šablona. Instance třídy `Template`, musí představovat šablonu zkompilovanou ze souboru na disku.

Metoda: `watch_file()`

Kontroluje předložený soubor, zda nebyl změněn.

Tato metoda může být použita pro kontrolu souborů, jestli nebyly změněny. Pokud došlo ke změně, šablony budou automaticky překompilovány.

Je velmi užitečná, pokud se používají statické proměnné (`TMPL_STATIC`) a jejich hodnoty jsou udržovány v samostatném souboru.

Návratová hodnota: Žádná

Parametry:

```
watch_file(files)
```

- `files` Pole jmen kontrolovaných souborů.

Třída: `TemplateProcessor`

Naplní šablonu daty a vykoná ji.

Tato třída provede zpracování zkompilované šablony. Používá se pro nastavování proměnných a cyklů. Po vykonání šablony je vrácen výsledek zpracování (tj. výsledný HTML kód).

Metoda: `TemplateProcessor()`

Konstruktor třídy.

Parametry:

```
TemplateProcessor(html_escape=1, magic_vars=1, global_vars=0, debug=0)
```

- `html_escape` Zapíná/Vypíná HTML escapování proměnných. Lze jím měnit implicitní nastavení escapování. Při HTML escapování jsou HTML závorky, ampersandy a dvojtečky nahrazeny HTML entitami.
- `magic_vars` Zapíná/Vypíná používání vestavěných proměnných cyklů. Implicitně jsou tyto vestavěné proměnné zapnuty.

- `global_vars` Zapíná globální hledání proměnných. Pokud není proměnné nalezena mezi lokáními proměnnými bloku, hledá se mezi globálními proměnnými šablony. Implicitně je globální hledání proměnných vypnuto. Pro danou proměnnou jej lze zapnout parametrem GLOBAL příkazu TMPL_VAR.
- `debug` Zapíná/Vypíná vypisování ladících výpisů. Implicitně jsou výpisy vypnuty. Pokud jsou zapnuty, probíhá jejich vypisování na standardní chybový výstup.

Metoda: `process()`

Vykoná zkompilovanou šablonu. Výsledek vrací jako řetězec.

Návratová hodnota: Výsledek vykonání šablony (HTML kód) v textovém řetězci.

Parametry:

```
process(template, part=None)
```

- `template` Zkompilovaná šablona, tj. instance třídy Template, která byla vytvořena buď třídou TemplateManager nebo TemplateCompiler.
- `part` Část vícedílné šablony, která má být vykonána (viz. `TMPL_BOUNDARY`). Tento parametr lze používat pouze s vícedílnými šablonami. Určuje číslo části šablony, která má být vykonána. Hodnotou parametru musí být kladné číslo. Části musí být vykonány ve správném pořadí. To znamená, že nemůžete vykonat část, která předchází jiné části, která již vykonána byla. Pokud není tento parametr uveden, je vykonána celá šablona.

Metoda: `set()`

Nastavuje hodnotu proměnné nebo cyklu.

Pokud chceme přiřadit proměnné hodnotu, předáme metodě jako první parametr název proměnné. Druhý parametr je par jednorozměrná hodnota (tj. textový řetězec, číslo, hodnota PHP proměnné, …).

Pokud chceme přiřadit hodnotu cyklu, postup je obdobný. Prvním parametrem je jméno cyklu a druhým parametrem pole, jehož obsahem jsou pole, která představují jednotlivé průchody cyklu. Tato pole jsou klíčována názvy proměnných resp. podcyklů a hodny polí představují hodnoty těchto proměnných resp. cyklů.

Metodu lze volat s jedním nebo dvěma parametry. Volání se dvěma parametry odpovídá přesně situaci popsané výše. Pokud potřebujete přiřadit hodnoty více proměnným a cyklům, lze metodu zavolat jen s jedním parametrem, který je pole klíčované názvy proměnných resp. cyklů a hodnoty pole odpovídají hodnotám příslušných proměnných resp. cyklů.

Návratová hodnota: Žádná

Parametry:

```
set(var, value)
```

- `var` Jméno proměnné nebo cyklu šablony.
- `value` Hodnota, která bude přiřazena proměnné nebo cyklu

Metoda: `reset()`

Vymaže všechna data šablony.

Tato metoda vymaže data nastavená v dané instanci třídy `TemplateProcessor`. Instance může být použita k vykonání libovolného počtu šablon, ale po vykonání jedné instance musí být zavolána tato metoda. Tím se vymažou všechna data a instance je použita pro další použití.

Návratová hodnota: Žádná

Parametry:

```
reset(keep_data=0)
```

- `keep_data` Povoluje/Zakazuje vymazání dat. Implicitně je vymazání povoleno.

Zdroje:

http://htmltmpl.sourceforge.net/

http://php.vrana.cz/sablony.php

Příklad použití htmltmpl

Pro lepší představu, jak celý systém htmltmpl funguje, si ukážeme jednoduchý příklad. Bude se skládat ze tří částí.

V první části si připravíme dvě šablony. Jedna šablona bude obsahovat pouze hlavičku HTML dokumentu. Ukážeme si, jak snadné je její vložení do druhé šablony, která bude obsahovat ukázky práce s jednotlivými příkazy šablonovacího systému.

Ve druhé části si ukážeme, jak pracovat v PHP skriptu se šablonovacím systémem. Předvedeme si vytvoření potřebných instancí tříd šablonovacího systému, připravení proměnných pro šablony a nakonec provedení šablony.

V poslední třetí části se podíváme na výstup šablonovacího systému, kterým bude validní XHTML dokument.

Část 1.: Příprava šablon

Šablona hlavicka.tmpl

Tato šablona nám poslouží pouze pro ukázku, jak ji vložit do hlavní šablony *stranka.tmpl*.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="cs" lang="cs">


<head>
  <title><TMPL_VAR nadpis></title>    ### Použití globální proměnné nadpis
  <meta http-equiv="content-language" content="cs" />
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <meta name="authors" content="Michal Vajbar" />
</head>
```

Hlavní šablona stranka.tmpl

V této šabloně je ukázáno použití všech příkazů šablonovacího systému htmltmpl.

```
### Vložení hlavičky dokumentu
<TMPL_INCLUDE hlavicka.tmpl>


### Vložíme hranici a rozdělíme tak šablonu na dvě části
<TMPL_BOUNDARY>
```

```
<body>
  <h1><TMPL_VAR nadpis></h1>
  <table border=1>
    <TMPL_LOOP Adresar>
      <TMPL_IF __FIRST__> ### První průchod cyklem
      <tr>
        <th></th>
        <th><TMPL_VAR p_prijmeni GLOBAL="1"></th>
        <th><TMPL_VAR p_jmeno    GLOBAL="1"></th>
        <th><TMPL_VAR p_pohlavi  GLOBAL="1"></th>
        <th><TMPL_VAR p_telefon  GLOBAL="1"></th>
      </tr>
      </TMPL_IF>
      <tr>
        ### Číslo průchodu / Počet všech průchodů
        <td><TMPL_VAR __PASS__>/<TMPL_VAR __PASSTOTAL__></td>
        <td><TMPL_VAR prijmeni></td>
        <td><TMPL_VAR jmeno></td>
        <td>
          <TMPL_IF pohlavi>  ### Rozhodne se, kterou globální proměnnou vypsat
            <TMPL_VAR muz GLOBAL="1">
          <TMPL_ELSE>
            <TMPL_VAR zena GLOBAL="1">
          </TMPL_IF>
        </td>
        <td>
          <TMPL_UNLESS Telefon>
            ---                  ### Pokud nemá telefon
          <TMPL_ELSE>
            <TMPL_LOOP Telefon> ### Vnořený cyklus telefonů
              <TMPL_VAR cislo><TMPL_UNLESS __LAST__>; </TMPL_UNLESS>
            </TMPL_LOOP>
          </TMPL_UNLESS>
        </td>
      </tr>
    </TMPL_LOOP>
  </table>
</body>
</html>
```

Část 2.: Příprava PHP skriptu

```
/* Vložíme šablonovací systém htmltmpl. */
require('./htmltmpl/htmltmpl.php');

/* Vytvoříme instanci třídy TemplateManager. */
$manager  = new TemplateManager();
/* Zkompilujeme šablonu. */
$template =& $manager->prepare('./stranka.tmpl');
/* Vytvoříme instanci třídy TemplateProcessor. */
$tproc    = new TemplateProcessor();

/* Připravíme si statické proměnné. Pro jednoduchost je nemáme v extra souboru. */
$tproc->set("nadpis", "Kontakty");

/* Připravíme si globální proměnné. */
$global = array(
  "p_prijmeni" => "Příjmení",
  "p_jmeno"    => "Jméno",
  "p_pohlavi"  => "Pohlaví",
  "p_telefon"  => "Telefon",
  "muz"       => "Muž",
  "zena"      => "Žena"
);
/* Předáme globální proměnné šabloně. */
$tproc->set($global);

/* Připravíme si všechny kontakty. Každá položka pole $adresar přestavuje jeden kontakt, tedy jeden průchod
cyklem. Všimneme si položky "Telefon", která představuje vnořený cyklus. U čtvrtého kontaktu předáme místo pole
hodnotu "false", která říká, že vnořený cyklus neexistuje. Položka pohlaví je true, pokud jdu o muže a false, jde-li o
ženu. */
  $adresar = array(
    array(
      "prijmeni" => "Dlouhý",
      "jmeno"    => "Václav",
      "pohlavi"  => true,
      "Telefon"  => array(
                      array("cislo" => "773 123 456")) ),
    array(
      "prijmeni" => "Krátká",
```

```
      "jmeno"    => "Ludmila",
      "pohlavi"  => false,
      "Telefon"  => array(
                      array("cislo" => "602 987 456"),
                      array("cislo" => "608 654 321")) ),
    array(
      "prijmeni" => "Tmavá",
      "jmeno"    => "Karolína",
      "pohlavi"  => 0,
      "Telefon"  => array(
                      array("cislo" => "721 231 465")) ),
    array(
      "prijmeni" => "Černý",
      "jmeno"    => "Petr",
      "pohlavi"  => 1,
      "Telefon"  => false)
  );
  /* Předání cyklu. První písmeno názvu musí být velké. */
  $tproc->set("Adresar", $adresar);

  /* Vykonáme šablonu a výsledný HTML dokument vypíšeme do PHP skriptu. */
  echo $tproc->process($template,1);   // Nejprve část první.
  echo $tproc->process($template,2);   // Potom část druhou.
```

Část 3.: Výsledný validní XHTML dokument

Na závěr se podívejme, jak vypadá výsledný HTML dokument.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="cs" lang="cs">

<head>
  <title>Kontakty</title>
  <meta http-equiv="content-language" content="cs" />
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <meta name="authors" content="Michal Vajbar" />
</head>
```

```
<body>
  <h1>Kontakty</h1>
  <table border=1>
      <tr>
        <th></th>          <th>Příjmení</th>   <th>Jméno</th>
        <th>Pohlaví</th>   <th>Telefon</th>
      </tr>
      <tr>
        <td>1/4</td>
        <td>Dlouhý</td>
        <td>Václav</td>
        <td>Muž</td>
        <td>773 123 456</td>
      </tr>
      <tr>
        <td>2/4</td>
        <td>Krátká</td>
        <td>Ludmila</td>
        <td>Žena</td>
        <td>602 987 456; 608 654 321</td>
      </tr>
      <tr>
        <td>3/4</td>
        <td>Tmavá</td>
        <td>Karolína</td>
        <td>Žena</td>
        <td>721 231 465</td>
      </tr>
      <tr>
        <td>4/4</td>
        <td>Černý</td>
        <td>Petr</td>
        <td>Muž</td>
        <td>---</td>
      </tr>
  </table>
</body>
</html>
```