

# Php - 3. část

Ing. Martin Dostal, Ph.D.

[madostal@kiv.zcu.cz](mailto:madostal@kiv.zcu.cz)  
[github.com/madostal](https://github.com/madostal)

# Obsah

- MVC
- Šablony - SMARTY a Twig

# Opakování

- Jak rychle vypíšete data z asociativního pole. S pomocí jakého cyklu?
  - Na co je třeba dát pozor?
- PDO - co to je? K čemu je to dobré?

# Opakování - řešení

- Jak rychle vypíšete data z asociativního pole. S pomocí jakého cyklu? **Foreach**
  - Na co je třeba dát pozor? **Test if not null.**

# MVC (MPO) - úvod

Často aplikovaný model při programování webu

- MPO – Model, Pohled, Ovladač
- MVC – Model, View, Controller
  - Model – komunikace s DB, konverze a kontrola dat
  - View – část zajišťující formátování výstupu
  - Controller – zpracovává vstup a předává ho modelu, řídí aplikaci
- Oddělení aplikační logiky od zobrazování je vhodné
- Aplikace je pružnější, snadno modifikovatelná - např. snadná záměna databázové vrstvy
- Kód je přehlednější
- Roste opětovná použitelnost částí systému

# MVC prakticky

## index.php

- hlavní soubor aplikace, includeuje všechny ostatní objekty
- dle konfigurace vytvoří spojení s databází
- dle parametru nebo URL určí typ obsahu
- např. ze souboru obsah.inc.php se načte požadovaný obsah. S pomocí php wrapperu se uloží do proměnné
- všechny texty se odešlou do šablony k zobrazení

# MVC - zásady 1

- mnoho různých implementací
- méně je někdy více, vždy je třeba sledovat cíl a nároky aplikace
- příkladem špatného MVC je např. plugin do Joomla - jednoduchou věc v rozsahu 5h je možné do pluginu naprogramovat za týden
- vytvoření jednoduchého pluginu do Joomla je popsáno v celé asi 350 stránkové knize

# MVC - zásady 2

- je třeba dělat věci pořádně, ale není vhodné se “vyžívat” ve složité struktuře aplikace
- občas kostra aplikace složená ze 3 souborů může být naprosto dostatečná:
  - index.php
  - obsah.inc.php
  - sablona.tpl
  - + css apod.



# Šablony

- Implementace MVC na webu i pomocí šablon
- HTML a logika zobrazení obsažena v šabloně
- Aplikační kód neobsahuje žádnou logiku zobrazení, pouze zpracuje požadavek, provede potřebné úkony a předá data šabloně
- Populární šablonové systémy: Twig, Smarty
  - pro SP doporučuji Twig z důvodu snadného zprovoznění na `students.kiv.zcu.cz`

# Šablonovací systémy - přehled

1. TWIG: <http://twig.sensiolabs.org/doc/intro.html>
2. Dwoo: <http://dwoo.org/>
  - zajímavá je část basic:  
<https://github.com/emulienfou/dwoo/wiki/Introduction>
3. Plates: <http://platesphp.com/>
4. Smarty templates: <http://www.smarty.net/>
  - ukládají horu zbytečností na disk
  - raději nepoužívat Smarty

# Smarty - hodně používaný z historických důvodů

- Používá speciální značky v souborech šablon
- Šablony jsou kompilovány do cachovaného PHP skriptu
- Instalace
  - Vytvořit adresář pod PHP, zkopírovat do něj stažené knihovny (smarty.php.net)
  - Do include cesty v php.ini přidat tento adresář
  - Vytvořit adresář, kde může smarty načítat svoji konfiguraci a soubory šablon: \templates a \smarty\_config

# Smarty - cachování

- Smarty dovoluje dvě úrovně cachování
- Když je poprvé šablona použita, smarty ji zkompiluje do čistého PHP a uloží do templates
- Značky smarty poté nemusí být znovu zpracovávány
- Adresář templates\_c
  - (Cachování aktuálně zobrazeného obsahu)

# První smarty

- Šablony umístit do adresáře templates
- Koncovka .tpl
- Smarty značky vnořené v {}
- Příklad hello.tpl
- Stránka hello.php tuto šablonu používá takto: příklad hello.php
- Předání dat šabloně: `$smarty->assign('name', $name);`
- Zobrazení stránky: metoda `display()`
- Smarty zjistí, že neexistuje zkompilovaná verze šablony
- Projde šablonu a zkonvertuje smarty značky na odpovídající PHP kód
- Výsledek uloží do `templates_c`

# Řídící struktury smarty

- Podmínky:
  - `{if $name=="cizince"} ... {/if}`
- Použití – odkaz na stránku pro přihlášení
- Plná podmíněná syntaxe if/elseif/else
- Cykly – foreach:
  - `<ul>`
  - `{foreach from=$myArray item=foo}`
  - `<li>{$foo}</li>`
  - `{/foreach}`
  - `</ul>`

# Proměnná smarty

- Asociativní pole
- Dovoluje přistupovat ke globálním proměnným php
- `$_COOKIE['name'] -> $smarty.cookies.name`

# Funkce systému smarty

- Lze volat vestavěné nebo uživatelsky definované funkce
  - Vestavěná funkce include: `{include file="header.tpl"}`
- Lze vložit jednu šablonu do jiné
- Přes atributy lze předat parametry
- Vlastní funkce lze registrovat: `register_function()`
- Očekává na vstupu pole `$params`
- `function create_table($params)`
- V šabloně: `{create_table data=$file_array}`
- V PHP - `$smarty->register_function('create_table', 'create_table')`
- Naplnění proměnné `$data` a vložení do šablony:
  - Předání do šablony `$smarty->assign('file_array', $data)`



# Modifikátory proměnných

- Funkce které modifikují zobrazení proměnných
- Např. volání nl2br() pro smarty proměnnou \$text:  
{ \$text|nl2br }
- Registrace modifikátoru:
  - `$smarty->register_modifier('encode', 'urlencode');`
- Seznam funkcí a modifikátorů:
  - <http://smarty.php.net/manual/en>
- Funkce, které chcete používat ve více šablonách registrujte v konstruktoru Smarty

# Smarty – další funkce

- Povolení cachování:

```
$smarty-> caching = true;
```

```
{assign var='prom' value='0'}
```

- Při použití matematiky musí být hodnota value v `...`

```
{assign var='prom' value='0'}
```

# Twig (<http://twig.sensiolabs.org/>)

- Základní použití - načtení šablony:
  - `$template = $twig->loadTemplate('index.html');`
- Vložení proměnných a zobrazení šablony:
  - `echo $template->render(  
 array('the' => 'variables', 'go' => 'here')  
);`

# Twig - šablona

- Výpis proměnné:
  - přímo v php: `<?php echo $var ?>`
  - ve Twigu: `{{ var }}` nebo `{{ var | raw }}` pro html
- Výpis pole:
  - `{% for user in users %}`
  - `* {{ user.name }}`
  - `{% else %}`
  - `No users have been found.`
  - `{% endfor %}`

# Twig - dědičnost

- Dědičnost:
  - `{% extends "layout.html" %}`
  - `{% block content %}`
  - Obsah bloku ...
  - `{% endblock %}`
- v tomto případě potomek přepíše block content u svého rodiče
- vše ostatní se použije od rodiče

# Twig - get started

```
require_once '/path/to/vendor/autoload.php';
```

```
$loader = new Twig_Loader_Array(array(  
    'index' => 'Hello {{ name }}!',  
));
```

```
$twig = new Twig_Environment($loader);
```

```
echo $twig->render('index', array('name' => 'Martin'));
```

# Twig - příklad šablony - kostra

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Titulek</title>  
  </head>  
  <body>  
    ...  
  </body>  
</html>
```

# Twig - proměnné do šablony

```
<body>
  <ul id="navigation">
    {% for item in navigation %}
      <li><a href="{{ item.href }}">{{ item.caption }}</a></li>
    {% endfor %}
  </ul>
  <h1>My Webpage</h1>
  {{ a_variable }}
</body>
```



# Twig - vložení html kódu

```
<html>
<head>
</head>
<body>
    <h1>{{ nadpis1 }}</h1>

    <div>
        {{ obsah | raw }}
    </div>
</body>
</html>
```

# MVC - pokračování

MVC dotažené do extrémů je popsané v samostatném projektu na Githubu:

- <https://github.com/panique/mini>

V rámci Vaší SP můžete použít tento přístup, nebo můžete použít jednodušší verzi prezentovanou na cvičení.

# Rekapitulace - co bych měl umět?

- MVC - co to je, k čemu je to dobré. Teoreticky i prakticky.
- Šablony - jaké jsou nejznámější šablonovací systémy?
- Šablony - praktické použití v rámci semestrálky.