

---

# Oddělení aplikační a prezentační logiky

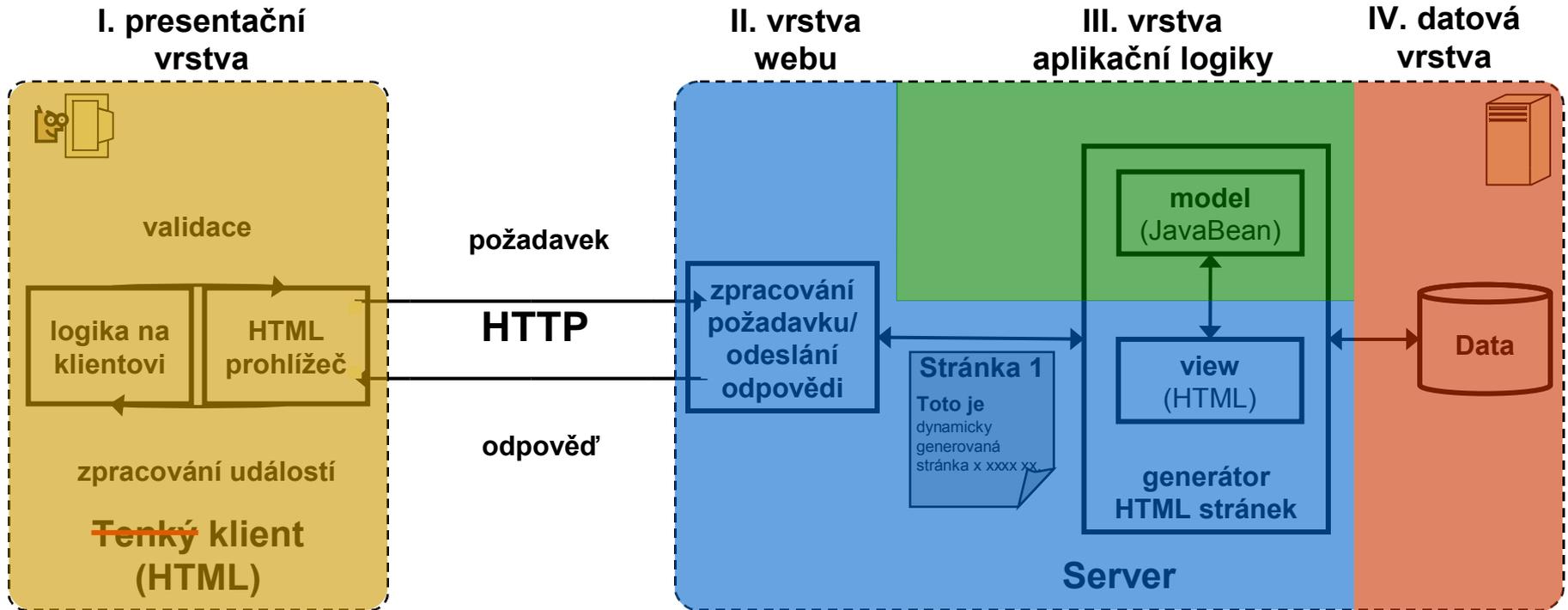
Martin Klíma



Computer Graphics Group

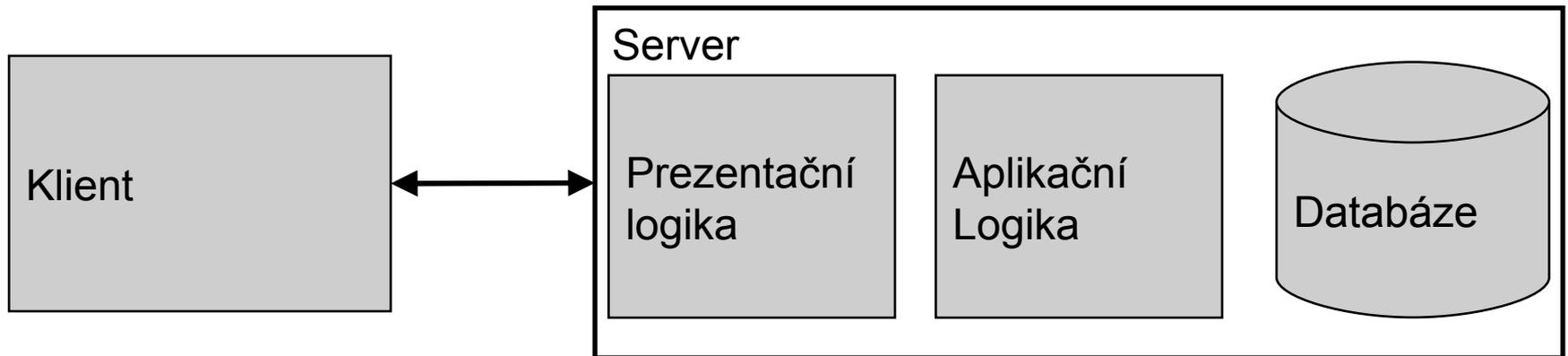


# Architektura web aplikace: dynamický web



# Čtyřvrstvá architektura

---



# Jak na to v PHP

---

Použití šablon (template)



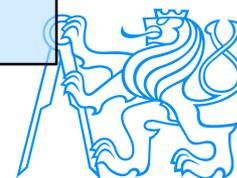
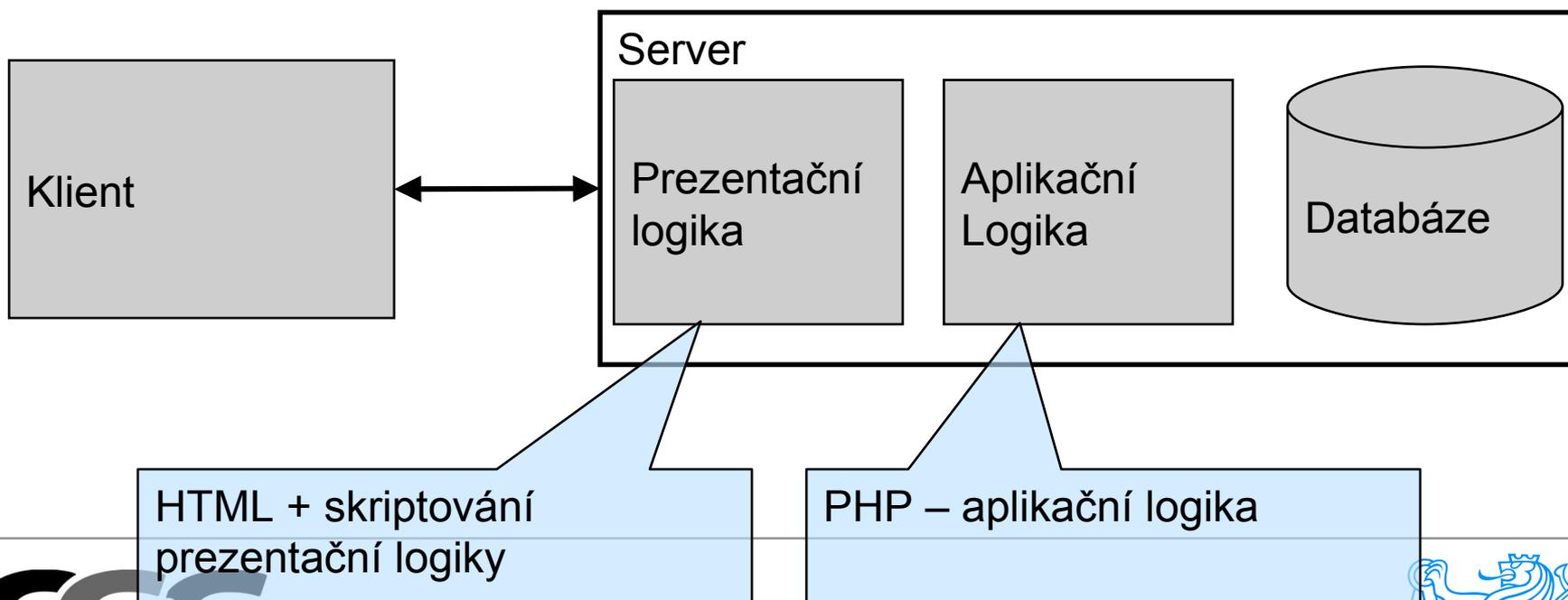
Computer Graphics Group



# Co šablony dělají

Oddělují aplikační kód a prezentační kód

Neoddělují jakýkoli kód a html,  
pouze se snaží tyto dvě logiky separovat.



# Výhody a nevýhody

---

## ■ Výhody

- Jednodušší vývoj
- Možnost dělby práce
- Průhledný a dobře spravovatelný kód
- Může být značně rychlé (cache)

## ■ Nevýhody

- Složitý projekt, nutno integrovat více souborů
- Aditivní výpočetní výkon
- Nutnost parsovat šablony
- Nutnost naučit se další skriptovací jazyk



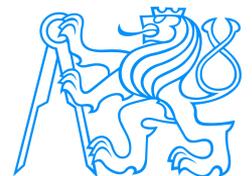
# Existující enginy

---

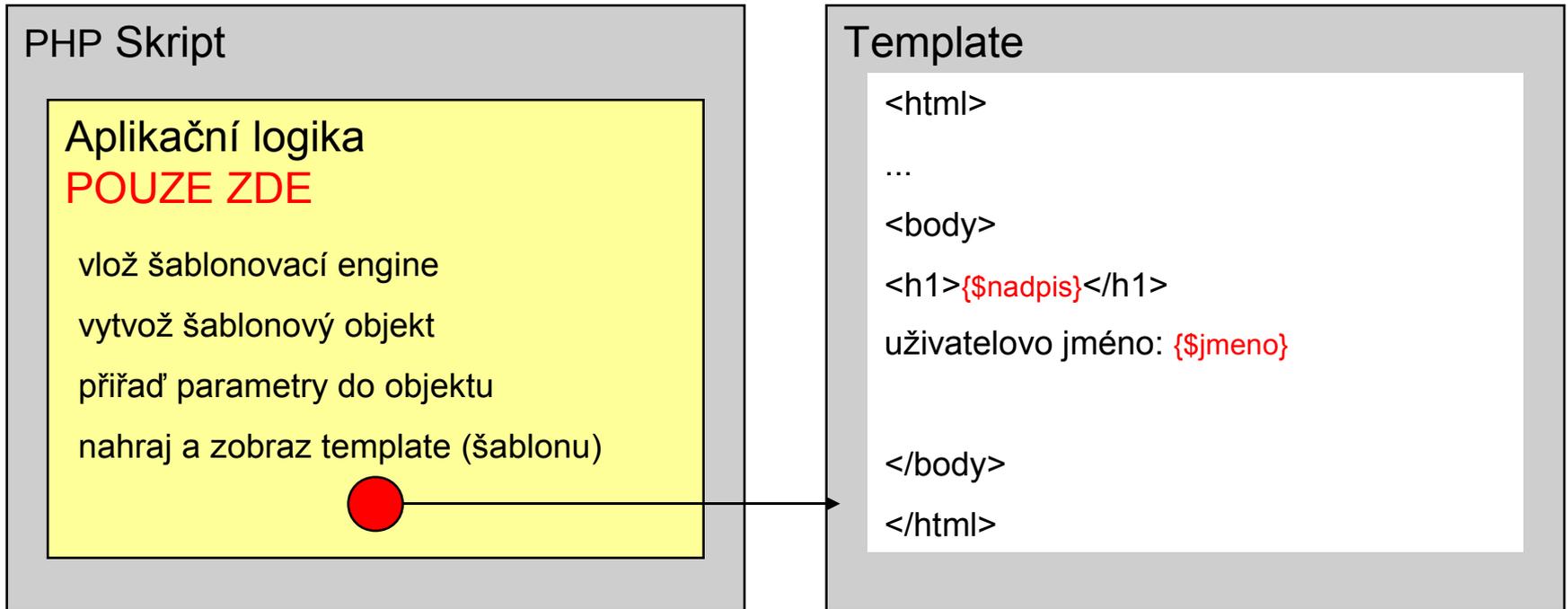
- × Smarty
- × FastTemplate
- × PHPLib
- × ...
- × Můj vlastní?

## Čím se liší

- Schopností rychle parsovat
- Schopností provádět vlastní skriptovací kód
- Cache
  
- Náš dnešní favorit:  
Smarty



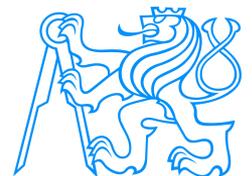
# Jak to vypadá



# Jak to funguje

---

1. Proved' PHP skript – aplikační logika
2. Vytvoř se Templatovací objekt
3. Tomuto objektu se přiřad' datové struktury z ap. logiky
4. Zavolej metodu display
  1. je zapnuta cache? ano-jdi na bod 8
  2. ne – existuje přeložená šablona? ano – jdi na bod 5
  3. proved' lexikální analýzu šablony
  4. expanduj příkazy pseudoskriptu do podoby PHP kódu
  5. přeloženou šablonu ulož
  6. proved' příkazy pseudoskriptu
  7. výsledek ulož do cache (pokud je zapnuta)
  8. výsledek zobraz na standardní výstup



# Jednoduchý projekt se šablonou smarty

## PHP Skript

```
<?php
// inicializace smarty engine
require_once("init_smarty.php");

// vyrobim si data, v tomto pripade info o
datumu
$datum = date("d.m.Y");

// vytvor sablonu
$templatovaci_objekt = new T_Template();

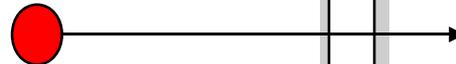
//prirad data do sablony
$templatovaci_objekt->assign("datum",
$datum);

// nech to zobrazit
$templatovaci_objekt-
>display('hello_world.html');

?>
```

## Template

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
<html>
  <head>
    <title>
      Hello world
    </title>
  </head>
  <body>
    Dnes je: {$datum}
  </body>
</html>
```



# Základní metody třídy Smarty (ale je jich více)

---

dokumentace je na:

<http://smarty.php.net/>

## Přiřazení

assign()

assign\_by\_ref()

register\_object()

## Výstup

display()

fetch()

## Rozšíření

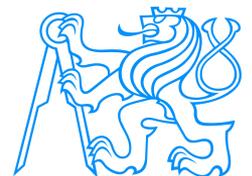
register\_function()

register\_modifier()

## Cache

is\_cached()

clear\_cache()



# Přiřazení

```
<?php
// inicializace smarty engine
require_once("init_smarty.php");

// vyrobim si data, v tomto pripade info o datumu
$datum = date("d.m.Y");

// vyrobim slozita data
$pole_hodnot = array("Jmeno"=>"Martin", "Prijmeni"=>"Klima");

// vytvor sablonu
$templatovaci_objekt = new T_Template();

//prirad data do sablony
$templatovaci_objekt->assign("datum", $datum);
$templatovaci_objekt->assign_by_ref("clovek", $pole_hodnot);

// nech to zobrazit
$templatovaci_objekt->display('prirazeni.html');
?>
```

Přiřazení jednoduché hodnoty

Přiřazení složité hodnoty



# Použití v šabloně

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head>
    <title>
      Hello world
    </title>
  </head>
  <body>
    Dnes je: {$datum} <br/>
    Jmeno: {$clovek.Jmeno}<br/>
    Prijmeni: {$clovek.Prijmeni}
  </body>
</html>
```

Asociativní pole jsou  
adresována pomocí tečkové  
notace



# Výstupy display & fetch

---

- display vrací výstup na standardní výstup
- fetch vrací výstup do proměnné
  
- použití: například mail se šablonou



# Přiřazení

```
<?php
// inicializace smarty engine
require_once("init_smarty.php");

// vyrobim si data, v tomto pripade info o datumu
$datum = date ("d.m.Y");

// vyrobim slozita data
$pole_hodnot = array("Jmeno"=>"František", "Prijmeni"=>"Vomáčka");

// vytvor sablonu
$templatovaci_objekt = new T_Template();

//prirad data do sablony
$templatovaci_objekt->assign("datum", $datum);
$templatovaci_objekt->assign_by_ref("jmeno", $pole_hodnot);

// ziskej sloucený mail
$text_dopisu = $templatovaci_objekt->fetch("mail.tpl");

// odesli ho
mail("vomacka@post.cz", "Automaticky mail", $text_dopisu);

$templatovaci_objekt->assign("dopis",$text_dopisu );
// nech to zobrazit
$templatovaci_objekt->display('fetch.html');
?>
```

Získání výsledku mailu

Výstup html

# Šablony

## fetch.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head>
    <title>
      Ukazka sestaveni dopisu
    </title>
  </head>
  <body>
    Ahoj, prave Ti byl zaslan tento mail:<br/>
    <pre>
      {$dopis}
    </pre>
  </body>
</html>
```

## mail.tpl

```
Vážený pane/paní {$jmeno.Jmeno} {$jmeno.Prijmeni}
blablabla
blablalba
blablabla

S pozdravem
  Martin Klíma
```



# Přiřazení – register object

```
class Trida {  
    public function metodaJedna ($params, &$smarty) {  
        echo "toto je metoda 1";  
    }  
  
    public function metodaDva($params, &$smarty) {  
        echo "tot je metoda 1";  
    }  
}  
  
// inicializace smarty engine  
require_once("init_smarty.php");  
  
// instance Tridy  
$objekt = new Trida();  
  
// vytvor sablonu  
$templatovaci_objekt = new T_Template();  
  
// registruj novou funkci  
$templatovaci_objekt->register_object('obj',  
$objekt, array("metodaJedna"));  
// nech to zobrazit  
$templatovaci_objekt->display('register_object.html');
```

Registrace tridy a definice viditelných metod



# Přiřazení – register object šablona

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head>
    <title>
      Registrace objektu
    </title>
  </head>
  <body>
    {obj->metodaJedna} <br/>
    {*obj->metodaDva*} <br/>
  </body>
</html>
```

Chyba, proto je to zakomentováno



# Rozšiřování funkčnosti – register function

```
<?php
```

```
function zjistiCenu($params, &$smarty) {  
    $cena = 100;  
    switch ($params['mena']) {  
        case "CZK": return $cena." CZK";  
        case "EUR": return ($cena/26)." EUR";  
        case "USD": return ($cena/18)." USD";  
    }  
}
```

```
// inicializace smarty engine  
require_once("init_smarty.php");
```

```
// vytvor sablonu  
$templatovaci_objekt = new T_Template();
```

```
// registruj novou funkci  
$templatovaci_objekt->register_function('cena', 'zjistiCenu');
```

```
// nech to zobrazit  
$templatovaci_objekt->display('register_function.html');  
?>
```

Registruji novou funkci pod názvem "cena" ukazující na funkci "zjistiCenu"

kuk: register\_function.php

# Použití rozšíření v šabloně

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head>
    <title>
      Registrace funkce
    </title>
  </head>
  <body>
    Cena v CZK: {cena mena="CZK"} <br/>
    Cena v EUR: {cena mena="EUR"} <br/>
    Cena v USD: {cena mena="USD"} <br/>
  </body>
</html>
```

kuk: register\_function.html



# Úkol: zobraz seznam klientů

---

- Předpoklad je, že máme databázi uživatelů.
- Zobraz jenom jejich seznam
- Aplikační logika
  - vytvoří datovou strukturu – pole uživatelů
  - vytvoří objekt šablony
  - přiřadí pole
  - nechá zobrazit
- Prezentační logika
  - definuje, jak toto pole zobrazit



# Jak vypadá funkce pro načtení pole

```
define("UZIVATEL", "session_uzivatel");
define("DB_HOST", "localhost");
define("DB_UZIVATEL", "root");
define("DB_HESLO", "root");
define("DB_JMENO", "test");
// vraci vysledek sql dotazu nad db
function dotaz ($sql) {
    mysql_connect(DB_HOST, DB_UZIVATEL, DB_HESLO);
    mysql_select_db(DB_JMENO);
    $vysledek = mysql_query($sql);
    return $vysledek;
}
function poleUzivatelu () {
    $vysledek = array();
    // sestav dotaz
    $dotaz = "SELECT * FROM uzivatel";
    // nacti vysledek
    $sql_vysledek = dotaz($dotaz);
    // osetreni chyby pri komunikaci s db
    if (!$sql_vysledek) die("Nepodarilo se spojit s databazi");
    // iteruj radky v db
    while ($radek = mysql_fetch_assoc($sql_vysledek))
        $vysledek[] = $radek;
    }
    // vrat pole radku
    return $vysledek;
}
```

Načteme tabulku z  
MySQL databáze



# Jak vypadá aplikační logika

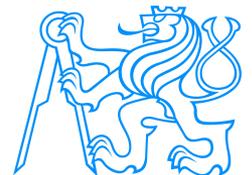
```
<?  
require_once("funkce.inc");  
require_once("init_smarty.php");  
  
// nacti pole uzivatelu  
$pole_uzivatelu = poleUzivatelu();  
  
// vytvor sablonu  
$templatovaci_objekt = new T_Template();  
  
//prirad data do sablony  
$templatovaci_objekt->assign_by_ref("uzivatele", $pole_uzivatelu);  
  
// nech to zobrazit  
$templatovaci_objekt->display('index.tpl');  
  
?>
```

načtení pole z DB

nový objekt Smarty

přiřazení datových struktur

spuštění procesu zobrazení



# Jak vypadá template

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=windows-
1250">
<title>Pokusna sablona</title>
</head>
<body>
<table summary="Seznam uzivatelu db">
{section name=i loop=$uzivatele}
    <tr>
        <td>{$uzivatele[i].Jmeno}</td>
        <td>{$uzivatele[i].Prijmeni}</td>
    </tr>
{/section}
</table>
</body>
</html>
```

Chyba!!!



# Jak vypadá template - správně

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=windows-
1250">
<title>Pokusna sablona</title>
</head>
<body>
<table summary="Seznam uzivatelu db">
{section name=i loop=$uzivatele}
  <tr>
    <td>{$uzivatele[i].Jmeno|escape:"html"}</td>
    <td>{$uzivatele[i].Prijmeni|escape:"html"}</td>
  </tr>
{/section}
</table>
</body>
</html>
```

Ochrana proti  
HTML neplatnému  
výstupu



# Nový požadavek – text ve dvou sloupcích

---

- PHP kód zůstává beze změny
- Prezentační logika se mění



# Modifikovaná prezentační logika

```
<table summary="Seznam uzivateleu db">
```

```
{section name=i loop=$uzivatele}
```

```
{if ($smarty.section.i.iteration mod 2) == 1}
```

```
<tr>
```

```
{/if}
```

```
<td>{$uzivatele[i].Jmeno|escape:"htmlall"}</td>
```

```
<td>{$uzivatele[i].Prijmeni|escape:"htmlall"}</td>
```

```
{if ($smarty.section.i.iteration mod 2) == 0}
```

```
</tr>
```

```
{/if}
```

```
{/section}
```

```
{if $smarty.section.i.rownum mod 2 == 1}
```

```
<td> --- </td>
```

```
<td> --- </td>
```

```
</tr>
```

```
{/if}
```

```
</table>
```

Lichá položka?

Ano: vlož značku <tr>

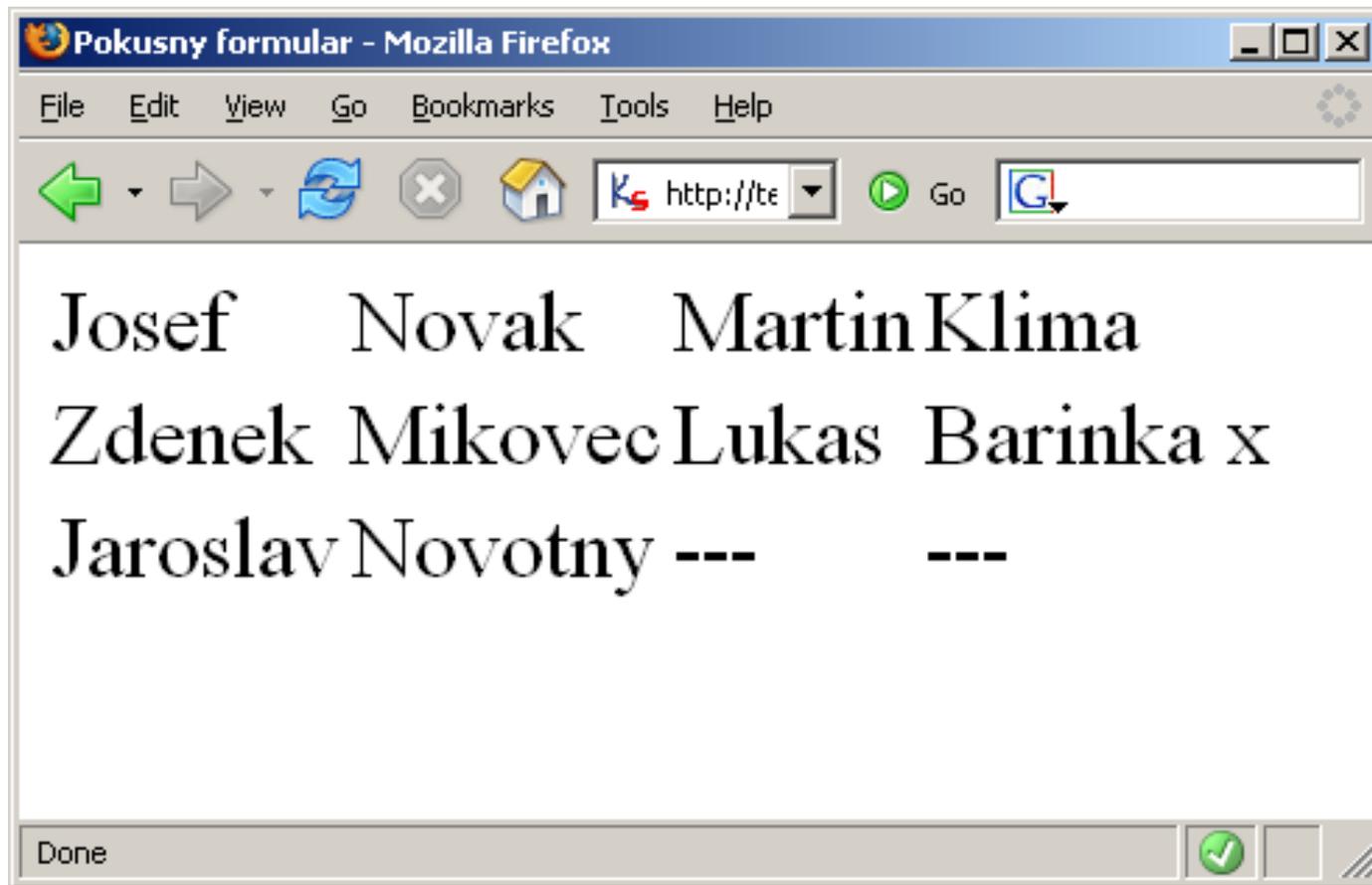
Sudá položka?

Ano: ukonči značku

</tr>

Co se stane, když je lichý počet položek

# Výsledek

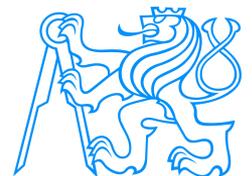


# Další požadavek: střídání řádků

---

Řešení:

- opět čistě jen prezentační logika



# Prezentační logika – střídání řádků

```
<table summary="Seznam uzivatele db">
{section name=i loop=$uzivatele}
{if ($smarty.section.i.iteration mod 2) == 1}
    <tr class="{cycle values="lichy, sudy"}">
{/if}
    <td>{$uzivatele[i].Jmeno|escape:"htmlall"}</td>
    <td>{$uzivatele[i].Prijmeni|escape:"htmlall"}</td>
{if ($smarty.section.i.iteration mod 2) == 0}
    </tr>
{/if}
{/section}
{if $smarty.section.i.rownum mod 2 == 1}
    <td> --- </td>
    <td> --- </td>
</tr>
{/if}
</table>
```

Cyklicky střídá  
parametry oddělené  
čárkou



# Střídání řádků – vložení CSS stylu

```
<title>Pokusny formular</title>
```

```
<style type="text/css">
```

```
tr.lichy { background-color: grey; }
```

```
tr.sudy { background-color: white; }
```

```
</style>
```

```
</head>
```

```
<table summary="Seznam uzivatelu db">
```

```
{section name=i loop=$uzivatele}
```

```
{if ($smarty.section.i.iteration mod 2) == 1}
```

```
<tr class="{cycle values="lichy, sudy"}">
```

```
{/if}
```

```
<td>{$uzivatele[i].Jmeno|escape:"htmlall"}</td>
```

```
<td>{$uzivatele[i].Prijmeni|escape:"htmlall"}</td>
```

```
{if ($smarty.section.i.iteration mod 2) == 0}
```

```
</tr>
```

```
{/if}
```

```
{/section}
```

```
{if $smarty.section.i.rownum mod 2 == 1}
```

```
<td> --- </td>
```

```
<td> --- </td>
```

```
</tr>
```

```
{/if}
```

```
</table>
```

Pozor!

Konflikt značek { a }

Smarty parser

interpretuje tyto značky



# Střídání řádků – vložení CSS stylu

```
<title>Pokusny formular</title>
```

```
{literal}
```

```
<style type="text/css">
```

```
tr.lichy { background-color: grey; }
```

```
tr.sudy { background-color: white; }
```

```
</style>
```

```
{/literal}
```

```
</head>
```

```
<table summary="Seznam uzivatelu db">
```

```
{section name=i loop=$uzivatele}
```

```
{if ($smarty.section.i.iteration mod 2) == 1}
```

```
<tr class="{cycle values="lichy, sudy"}">
```

```
{/if}
```

```
<td>{$uzivatele[i].Jmeno|escape:"htmlall"}</td>
```

```
<td>{$uzivatele[i].Prijmeni|escape:"htmlall"}</td>
```

```
{if ($smarty.section.i.iteration mod 2) == 0}
```

```
</tr>
```

```
{/if}
```

```
{/section}
```

```
{if $smarty.section.i.rownum mod 2 == 1}
```

```
<td> --- </td>
```

```
<td> --- </td>
```

```
</tr>
```

```
{/if}
```

```
</table>
```

Neinterpretuj jako  
Smarty kód

Toto už není konflikt



# Vkládání šablon

---

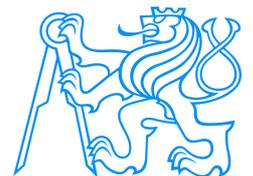
- Šablony lze libovolně vnořovat
- Výhoda:
  - Řešení elementárních problémů
  - Elegance
  - Znovupoužitelnost
- Nevýhoda:
  - Více souborů
  - Problém s cache (může být výhoda)



# Příklad

---

- Vytvořte seznam uživatelů
- Seznam bude obecný
- Každá položka v seznamu bude zobrazovat detail uživatele



# Řešení: aplikační logika zůstává

---

```
require_once ("funkce.inc");
require_once ("init_smarty.php");

// nacti pole uzivatelu
$pole_uzivatelu = poleUzivatelu();

// vytvor sablonu
$templatovaci_objekt = & new T_Template();

//prirad data do sablony
$templatovaci_objekt->assign_by_ref("uzivatele", $pole_uzivatelu);

// nech to zobrazit
$templatovaci_objekt->display('index2.tpl');
```



# Šablona první úrovně – rozložení seznamu

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=windows-1250">
<title>Pokusny formular</title>
</head>
<body>
{foreach from=$uzivatele item=aktualni_uzivatel}
{include file="detail_uzivatele.tpl"}
{/foreach}
</body>
</html>
```

Iteruj přes pole \$uzivatele  
aktualni položku prirad do promenne \$aktualni\_uzivatel

Vlož definici detailu



# Šablona druhé úrovně – detail

Smarty komentář. Je odstraněn při překladu šablony

{\* v této šabloně předpokládám, že informace o uživateli jsou v poli v proměnné \$aktualni\_uzivatel\*}

<hr>

<div>Jméno: {\$aktualni\_uzivatel.Jmeno}</div>

<div>Příjmení: {\$aktualni\_uzivatel.Prijmeni}</div>

<div>Logovací jméno: {\$aktualni\_uzivatel.LogovaciJmeno}</div>

Vložená šablona dědí definované proměnné



# Šablona druhé úrovně – detail

Smarty komentář. Je odstraněn při překladu šablony

{\* v této šabloně předpokládám, že informace o uživateli jsou v poli v proměnné \$aktualni\_uzivatel\*}

<hr>

<div>Jméno: {\$aktualni\_uzivatel.Jmeno|escape:"html"}</div>

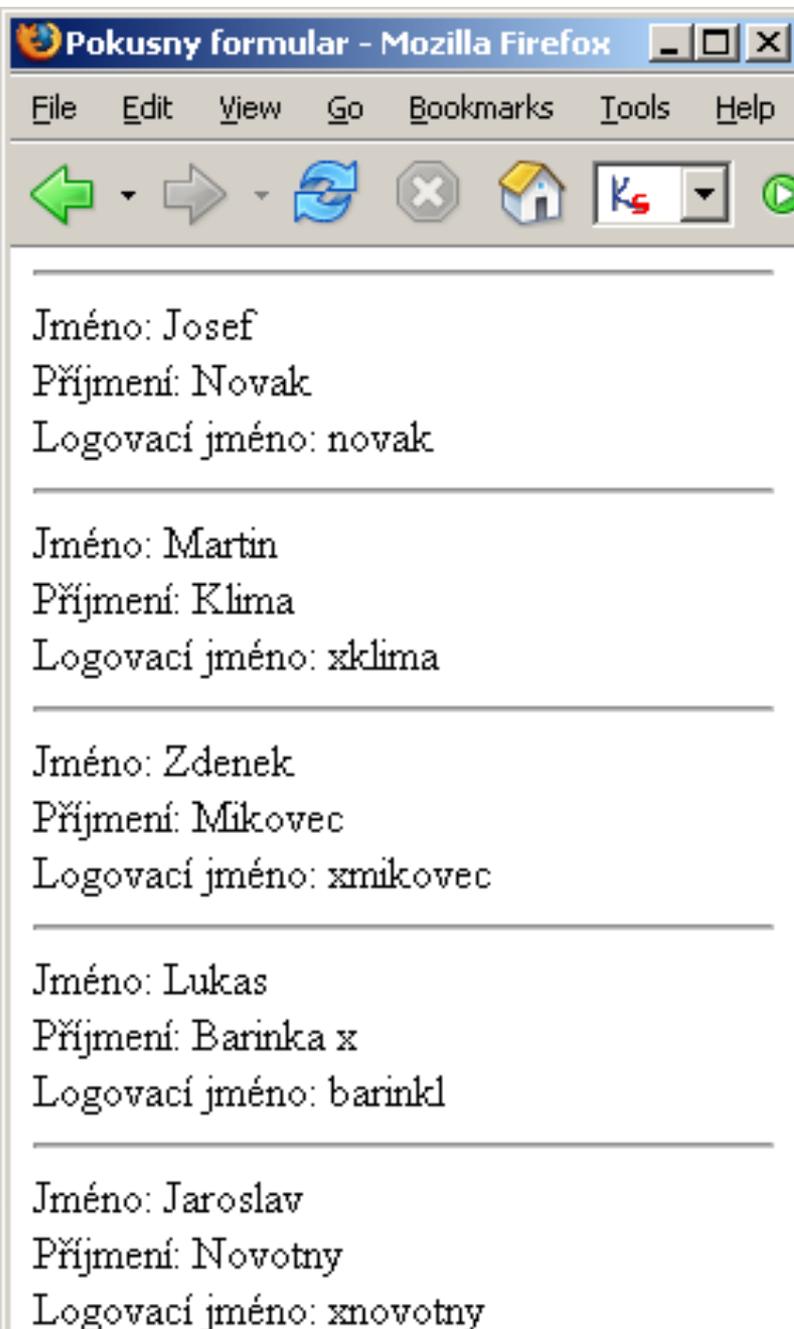
<div>Příjmení: {\$aktualni\_uzivatel.Prijmeni|escape:"html"}</div>

<div>Logovací jméno: {\$aktualni\_uzivatel.LogovaciJmeno|escape:"html"}</div>

Vložená šablona dědí definované proměnné



# Výsledek



Pokusny formular - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

← → ↻ × 🏠 Ks ▶

---

Jméno: Josef  
Příjmení: Novak  
Logovací jméno: novak

---

Jméno: Martin  
Příjmení: Klima  
Logovací jméno: xklima

---

Jméno: Zdenek  
Příjmení: Mikovec  
Logovací jméno: xmikovec

---

Jméno: Lukas  
Příjmení: Barinka x  
Logovací jméno: barinkl

---

Jméno: Jaroslav  
Příjmení: Novotny  
Logovací jméno: xnovotny



# Cache

---

- Jednou vygenerovaný výsledek může být uložen a použit znovu
- Výhoda:
  - Výrazné zrychlení odezvy
  - Méně dotazů do db
- Nevýhoda:
  - Zabírá prostor na disku
  - Aditivní kód
  - Uživatel nedostává aktuální data
  - Delší zpracování stránek, které nejsou v cache



# Příklad – jen aplikační logika

```
<?  
require_once("funkce.inc");  
require_once("init_smarty.php");  
  
// nacti pole uzivatelu  
$pole_uzivatelu = poleUzivatelu();  
  
// vytvor sablonu  
$templatovaci_objekt = & new T_Template();  
$templatovaci_objekt-> caching = true;  
  
//prirad data do sablony  
$templatovaci_objekt->assign_by_ref("uzivatele", $pole_uzivatelu);  
  
// nech to zobrazit  
$templatovaci_objekt->display('index.tpl');
```

Zde **NE**šetříme čas

Zapni cache

Zde šetříme čas

kuk smarty3-caching/index.php

```
?>
```

# Příklad – cache - oprava

```
<?  
require_once("funkce.inc");  
require_once("init_smarty.php");  
  
// vytvor sablonu  
$templatovaci_objekt = & new T_Template();  
$templatovaci_objekt->キャッシング = true;  
  
if ( ! $templatovaci_objekt->is_cached("index.tpl") ) {  
    // nacti pole uzivatelu  
    $pole_uzivatelu = poleUzivatelu();  
    //prirad data do sablony  
    $templatovaci_objekt->assign_by_ref("uzivatele",  
$pole_uzivatelu);  
}  
  
// nech to zobrazit  
$templatovaci_objekt->display('index.tpl');  
  
?>
```

Zapni cache

Zde šetříme čas  
když to lze

Zde šetříme čas,  
když to lze

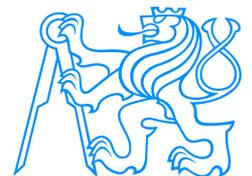
kuk smarty3-caching/index2.php



# Debug - ladění

---

Umožňuje zobrazit parametry přiřazené k template objektu.  
Do těla template stačí napsat značku **{debug}**



# Výsledek

```
http://testserver - Smarty Debug Console - Mozilla Firefox

Smarty Debug Console
included templates & config files (load time in seconds):
no templates included
assigned template variables:
{$SCRIPT_NAME}      /smarty2/index2.php
{$app_name}         Test
{$uzivatele}       Array (5)
                    \r0 => Array (6)
                    \r  ID => 1
                    \r  LogovaciJmeno => novak
                    \r  Jmeno => Josef
                    \r  Prijmeni => Novak
                    \r  Heslo => nov
                    \r  Povolen => 1
                    \r1 => Array (6)
                    \r  ID => 2
                    \r  LogovaciJmeno => xklima
                    \r  Jmeno => Martin
                    \r  Prijmeni => Klima
                    \r  Heslo => klim
                    \r  Povolen => 1
                    \r2 => Array (6)
                    \r  ID => 3
                    \r  LogovaciJmeno => xmikovec
                    \r  Jmeno => Zdenek
                    \r  Prijmeni => Mikovec
                    \r  Heslo => mik
                    \r  Povolen => 1
                    \r3 => Array (6)
                    \r  ID => 4
                    \r  LogovaciJmeno => barinkl
                    \r  Jmeno => Lukas
                    \r  Prijmeni => Barinka x
                    \r  Heslo => bar
                    \r  Povolen => 1
                    \r4 => Array (6)
                    \r  ID => 8
                    \r  LogovaciJmeno => xnovotny
                    \r  Jmeno => Jaroslav
                    \r  Prijmeni => Novotny
                    \r  Heslo => nov
```



# Filtry

---

- Životní cyklus šablony
  - Prochází řadou filtrů
  - Před překladem
  - Po překladu
  - Po vykonání



# Prefilter

---

- Textový filter, kterým je prohnán template ještě předtím, než je zkompilován
- Hodí se k odstranění uživatelských komentářů, preprocessing obecně



# Postfilter

---

- Textový filter, kterým je prohnán template poté, co byl zkompilován
- Hodí se např. k přidání nějaké spec. informace



# Output filter

---

- Textový filter, kterým je prohnán template poté, co byl zkompilován
- Pracuje tedy nad kompletním výstupem
- Hodí se např. k zvýraznění některých slov, jejich potlačení, ochrana před vulgaritami atd.



# Jak se filtry používají

```
require_once("funkce.inc");
require_once("init_smarty.php");
function muj_output_filter($tpl_output, &$smarty)
{
    // provede nejakou textovou manipulaci a vrati tento modifikovany
    text
    // budeme nahrazovat tyto vyrazy
    $vzor[] = '/Trabant/';
    $vzor[] = '/Tatra 613/';
    $vzor[] = '/Lada/';
    $nahrazeni[] = 'splášený vysavač';
    $nahrazeni[] = 'papalášfáro';
    $nahrazeni[] = 'Žigulík';
    // proved nahrazeni a vrat vysledek
    return preg_replace($vzor, $nahrazeni, $tpl_output);
}

// vytvor sablonu
$templatovaci_objekt = & new T_Template();

// registrace výstupní funkce
$templatovaci_objekt->register_outputfilter("muj_output_filter");

// nech to zobrazit
$templatovaci_objekt->display('smarty_filter1.tpl');
```

# Jak se filtry používají – pokr.

---

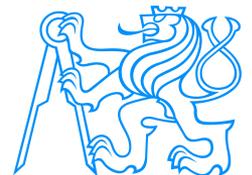
```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=windows-1250">
<title>Pokus s filtrem</title>
</head>
<body>
<h1>Filtr</h1>
<p>Na této stránce se dějí zajímavé věci s filtry.</p>
<p>Franta říkal, že nejlepší auto je Trabant.</p>
<p>Pepa ale říkal, že on si koupí jediné Tatra 613.</p>
<p>Já si ale myslím, že nejlepší je Lada.</a>
</body>
</html>
```



# Výsledek

---

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=windows-1250">
<title>Pokus s filtrem</title>
</head>
<body>
<h1>Filtr</h1>
<p>Na této stránce se dějí zajímavé věci s filtry.</p>
<p>Franta říkal, že nejlepší auto je splašený vysavač.</p>
<p>Pepa ale říkal, že on si koupí jedině papalášfáro.</p>
<p>Já si ale myslím, že nejlepší je Žigulík.</a>
</body>
</html>
```



---

**Používejte Smarty, je to dobré 😊**

Děkuji za pozornost,  
Martin Klíma

