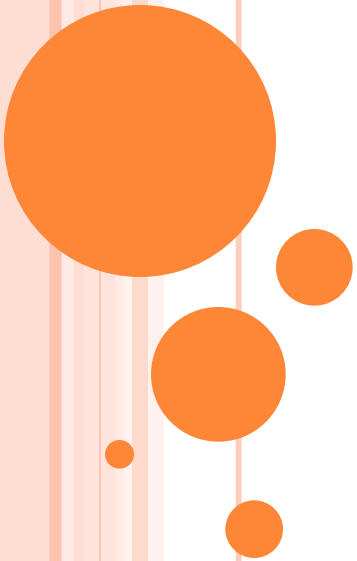

PHP – OOP, ŠABLONY

WEB 2013-2014

OOP (1)

- Funkce a data svázány do objektů
- Objekt je definován třídou
- Třída definuje atributy a metody
- Objekt = instance třídy
- Vytvoření instance – volání konstruktoru
  - V PHP nazýván `__construct()`
- Dědičnost/zapouzdření/polymorfismus
- Ukázka – User.php

OOP (2)

- Názvy tříd
  - První písmeno velké (rozlišení názvu třídy od názvu objektu
  - K simulaci zanoření jmenných prostorů lze použít podtržítko
  - Víceslovné názvy zřetězeny + první písmeno každého slova velké
  - `class XML_RSS{}, class Text_PrettyPrinter`
- Parametr `$this` je automaticky vytvořen uvnitř metody třídy a reprezentuje samotný objekt
- Pro přístup k metodám/vlastnostem použijte ->

Dědičnost

- Dědičný vztah je definován klíčovým slovem `extends`
- Třída dědí vlastnosti/metody nadřazené třídy, lze je přepisovat/přidat další
- `class AdminUser extends User {}`
- Musí se ručně volat konstruktor rodičovské třídy:

```
parent::__construct($name,
$birthday)
```

Zapouzdření

- Od PHP5 lze rozdělit viditelnost dat/metod na veřejnou, chráněnou a soukromou
- public – přístupná odkudkoliv
- protected – není zvenku přístupná, ale je přístupná v podtřídě
- private – přístupná pouze uvnitř třídy, ve které je definována

Statické atributy/metody

- `class TestClass {public static $counter}`
- Vázány přímo na třídu, ne na objekt
- Volány syntaxí: `ClassName::method()` a `ClassName::property`
- Použití `$this` není možné
- Změna statické vlastnosti se projeví ve všech instancích dané třídy
- `self/parent` – odkaz na třídu,resp. rodiče
- Přístup ke statickým proměnným uvnitř třídy:

```
class TestClass {
  public static $counter=0;
  public $id;
  public function __construct() {
    $this->id = self::$counter++;
  }
}
```

Speciální metody

- Konstruktor: `__construct()`
- `destruct()` – metoda volaná při rušení objektu (uvolnění zdrojů)
- V PHP4 jsou objekty předávány hodnotou:
  `$obj = new TestClass(); $copy = $obj;`
  (3 kopie)
- V PHP5 přiřazení vrací handle na objekt (1 kopie)
- `__clone` pro vytvoření kopie
  - `$this->id = self::$counter++;`
    (pro příklad výše)
- `__get($varname), __set($varname, $value)`
  - Volány při získávání hodnoty proměnné, resp. přiřazování hodnoty proměnné
- `__call($funcname, $args)`
  - Spuštěna, když voláme metodu, která neexistuje

Návrhové vzory (1)

- Zevšeobecněná řešení tříd problémů, se kterými se programátoři setkávají nejčastěji
- Př. práce s DB – třída, která vytvoří spojení, vybere db, provede zadaný dotaz a vrátí výsledek
- Adaptér vzoru – poskytuje přístup k objektu prostřednictvím specifického rozhraní (rozhraní k sadě procedur)

```
$dbh = new DB_WEB(přístup k db);
$smtp = $dbh->execute($query);
// projít výsledek
```

Návrhové vzory (2)

- Šablona vzoru
  - Třída, která modifikuje logiku podtřídy, čímž ji rozšiřuje
  - Např. skrytí parametrů připojení
  - 1 třída pro testovací DB a druhá pro ostrou

- Polymorfismus

```
function show_entry($entry_id, $dbh) {
  …
  $dbh->execute(…)
  …
}
```

Delegace

- Objekt má jako atribut jiný objekt, který používá k provádění úloh

```
class DB_WEB {
  protected $dbh;
  public function setDB($dbh) {
    $this->dbh = $dbh;
  }
  // funkce pracující s $dbh
}
```

Rozhraní

- Při delegaci musí být zaručené, že předaná `$dbh` implementuje potřebné metody
- Rozhraní je vlastně kostra třídy, definuje metody třídy, bez kódu

```
interface DB_Connection {
     public execute($query);
}
```

- `class DB_WEB implements DB_Connection {…}`
- Třída může implementovat více rozhraní -> lze obejít to, že nelze dědit od více tříd
- Abstraktní třída – může obsahovat jak hotové metody, tak i abstraktní metody, které musí být definovány potomkem

Kontrola typu

- `if (!is_a($dbh, "DB_Connection"))`

  … chyba
- Od PHP5 – možnost kontroly typu:

```
function setDB(DB_Connection $dbh) {…}
```

Vzor factory

- změna z jedné DB na druhou = nahrazení všech výskytů třídy

  z `$dbh = new DB_MySQL();`

  na `$dbh = new DB_Oracle();`

- Řešení – funkce typu factory

```
function DB_Connection_Factory() {
  return new DB_MySQL();
}
$dbh = DB_Connection_Factory();
```

MPO (MVC)

- Často aplikovaný model při programování webu
  - MPO – Model, Pohled, Ovladač
  - MVC – Model, View, Controller
- Model – provádí obchodní logiku (práce s DB)
- Pohled – část zajišťující formátování výstupu systému
- Ovladač – zpracovává vstup a předává ho modelu, řídí aplikaci
- Oddělení aplikační logiky od zobrazování je vhodné
  - Aplikace je pružnější, snadno modifikovatelná
  - Kód vypadá přehlednější
  - Roste opětovná použitelnost částí systému

Šablony

- Implementace MPO na webu 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
- Jeden z nejpopulárnějších šablonových systémů – Smarty

Smarty

- Používá speciální značky s 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

    (knihovnu Smarty lze kopírovat i s aplikací)

  - 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

- Rozšíření základní třídy pro každou aplikaci (příklad)
- 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()
- Zavolání 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 – příklad

Proměnná smarty

- Asociativní pole
- Dovoluje přistupovat ke globálním proměnným php
- $_COOKIE[‘name’] -> $smarty.cookies.name
- Příklad

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 $data
    - 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'}`