Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta, katedra informatiky

Obor: Výpočetní technika a informatika

PUBLIKAČNÍ SYSTÉM PRO WWW

Praktická bakalářská práce

Autor: Libor Suchý

Vedoucí práce: RNDr. Hana Havelková

České Budějovice, 2006

Anotace

Cílem mé absolventské práce bylo vytvoření publikačního systému pro WWW. Od systému jsem očekával dva hlavní přínosy. Prvním bylo vytvoření jednotného a znovupoužitelného prostředí, ve kterém by majitelé internetových stránek mohli spravovat jejich obsah. Druhým cílem bylo zavedení objektového přístupu do tvorby dynamických stránek v PHP a s tím související zvýšení efektivity programování.

Publikační systém, dále v textu označovaný jménem *Publisher*, byl postaven na platformě PHP. Staví ale i na dalších technologiích, které jsou běžně používané při vývoji webových systémů. Uživatelské rozhraní systému je vytvořeno v XHML za podpory kaskádových stylů CSS a skriptování v jazyce JavaScript. Samozřejmě je podporována práce s databází pomocí speciálně vytvořené třídy.

Tento dokument si neklade za cíl vysvětlit či detailně rozebrat zmíněné technologie. Má sloužit k vysvětlení principu vytvořeného publikačního systému a podpořit programátory v případné tvorbě dodatečných modulů. Text předpokládá alespoň základní znalosti v oblasti tvorby dynamických stránek v PHP a znalost objektově orientovaného programování. Pro porozumění procesu konfigurace modulu Catalog je rovněž nezbytná znalost XML.

Abstract

The main goal of my dissertation work was to create a content management system for web publishing. I expected two main benefits. The first one was creation of uniform and reusable environment, in which the web site owner would be able to manage its content. The other goal was introducing object oriented approach to web design in PHP and increasing the effectiveness of programming.

The content management system, further in this text called as *Publisher,* was built on the PHP platform. But it's based upon other technologies commonly used in web system development. User interface of the system is designed with XHTML, cascading style sheets (CSS) and scripting in JavaScript. Of course there is a database access support in a form of special set of classes.

This document is not meant as explanation or detailed reference for mentioned technologies. It's dedicated to explain the fundamentals of created system and it should support programmers in eventual creation of extra modules. The text requires at least basic knowledge in a field of PHP web site design and object oriented programming. In order to understand a configuration of Catalog module it's necessary to understand XML.

Poděkování

Děkuji doktorce Haně Havelkové za projevenou důvěru a poskytnutou volnost při tvorbě této bakalářské práce. Dále bych chtěl poděkovat mému kolegovi Josefu Kyrianovi za testování systému a odhalení řady chyb.

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně, a že jsem veškeré použité zdroje uvedl v Seznamu použitých zdrojů.

.....

Libor Suchý

Obsah

1 ÚVOD 9

1.1 TÉMA PRÁCE 9
1.2 DEFINICE POJMŮ 9
1.3 MOTIVACE 10

2 PUBLISHER 13

2.1 DVA POHLEDY 13
2.2 SYSTÉM 13
2.3 KONFIGURAČNÍ SOUBORY 14
2.4 TŘÍDY 14
2.4.1 PUBLISHER 14
2.4.2 MODULE 14
2.4.3 DATABASEHANDLER 15
2.4.4 QUERYRESULT 16
2.4.5 TREEVIEW 16
2.4.6 TREEVIEWNODE 17
2.5 VÝJIMKY 17
2.5.1 ZPRACOVÁNÍ NEZACHYCENÝCH VÝJIMEK 18
2.5.2 PUBLISHEREXCEPTION 19
2.5.3 SYSTEMERROREXCEPTION 19
2.5.4 INVALIDMETHODCALLEXCEPTION 19
2.5.5 INVALIDARGEXCEPTION 19
2.5.6 DATABASECONNECTIONFAILEDEXCEPTION 20
2.5.7 QUERYFAILEDEXCEPTION 20
2.5.8 RECORDNOTFOUNDEXCEPTION 20
2.5.9 INVALIDSETTINGSFILEEXCEPTION 20
2.5.10 INVALIDSETTINGVALUEEXCEPTION 20
2.5.11 SETTINGNOTFOUNDEXCEPTION 20

2.5.12 FILENOTFOUNDEXCEPTION 20
2.5.13 INDEXOUTOFBOUNDSEXCEPTION 20
2.6 ZPŮSOB DOKUMENTACE TŘÍD 20

3 MODULY PUBLISHERU 23

3.1 MODUL WEBPARTS 25
3.1.1 POPIS MODULU 25
3.1.2 TŘÍDA WEBPARTS 25
3.1.3 TŘÍDA WEBPART 25
3.2 MODUL CATALOG 26
3.2.1 POPIS MODULU 26
3.2.2 STATICKÝ PŘÍSTUP 26
3.2.2.1 Definice katalogu 27
3.2.2.2 Definice atributu položky katalogu 28
3.2.2.3 Definice typu obrázků 29
3.2.3 DYNAMICKÝ PŘÍSTUP 31
3.2.4 TŘÍDY MODULU CATALOG 31
3.2.4.1 Catalogs 31
3.2.4.2 Catalog 31
3.2.4.3 CatalogItem 31
3.2.4.4 CatalogItems 32
3.2.4.5 CatalogItemAttribute 32
3.2.4.6 CatalogItemAttributes 32
3.2.4.7 CatalogItemAttributeOption 32
3.2.4.8 CatalogItemAttributeOptions 32
3.2.4.9 CatalogItemImage 32
3.2.4.10 CatalogItemImages 32
3.2.4.11 CatalogItemImagesType 32
3.2.4.12 CatalogItemImagesTypes 33

4 POUŽITÍ PUBLISHERU PŘI TVORBĚ WWW STRÁNEK 34

4.1 VYTVOŘENÍ XHTML + CSS ZÁKLADU 34
4.2 INSTALACE A NASTAVENÍ PUBLISHERU 35
4.2.1 NASTAVENÍ MODULŮ 35
4.3 NAPROGRAMOVÁNÍ WEBU 41
4.3.1 ZAVEDENÍ PROSTŘEDÍ PUBLISHERU 41
4.3.2 VYGENEROVÁNÍ MENU 41
4.3.3 VÝPIS OBSAHU 43
4.3.4 VÝPIS KATALOGU 44
4.4 SHRNUTÍ NÁROČNOSTI 45

1 Úvod

1.1 Téma práce

Publikační systém pro WWW je software, nejčastěji webová aplikace, pomocí které lze spravovat obsah internetových stránek. V angličtině se tento druh software označuje pojmem „Content Management System" (zkráceně CMS). Výhody použití takového systému jsou zřejmé. Tvůrce stránek je zbaven povinnosti vytvářet pro každou zakázku rozhraní pro správu obsahu. Toto rozhraní je již vytvořeno v rámci publikačního systému a úkolem tvůrce stránek je pouze vhodným způsobem jej napojit na obsah vytvářených internetových stránek. Publikační systémy také většinou disponují nástroji, které dále usnadňují tvůrcům jejich práci.

V dnešní době je k dispozici celá řada publikačních systémů, z nichž některé jsou vyvíjeny Open Source. Většina firem zabývajících se tvorbou internetových stránek však vyvíjí a používá vlastní publikační systémy.

Mnou vytvořený publikační systém, dále v textu označovaný jménem *Publisher*, byl postaven na platformě PHP. Staví ale i na dalších technologiích, které jsou běžně používané při vývoji webových aplikací. Uživatelské rozhraní systému je vytvořeno v XHML za podpory kaskádových stylů CSS a skriptování v jazyce JavaScript. Samozřejmě je podporována práce s databází pomocí speciálně vytvořené třídy.

1.2 Definice pojmů

Než se pustím do vysvětlování systému a popisu jednotlivých částí, je potřeba vysvětlit několik termínů, které budu v textu používat.

- *Tvůrce webu*
  Člověk, který se zabývá programováním internetových stránek
- *Programátor modulů*
  Člověk, který vyrábí dodatečné moduly pro Publisher.
- *Administrační rozhraní*
  Grafické uživatelské rozhraní Publisheru, ve kterém majitel internetových stránek provádí správu obsahu.
- *Prostředí Publisheru*
  Prostor v PHP kódu, kde lze využívat funkcí Publisheru.

1.3 Motivace

Již několik let se intenzivně zabývám tvorbou internetových stránek. Je to nejen můj koníček, ale i jeden z hlavních zdrojů mých příjmů. Základy HTML jsem se naučil během jedné probděné noci a první web byl na světě. Nedlouho poté mě spolužák zasvětil do skriptovacího jazyka PHP a databáze MySQL. Zpočátku jsem se držel statického přístupu a PHP jsem příliš nedůvěřoval, ale to se záhy změnilo. Po několika lekcích jsem si už na statické HTML a rámy ani nevzpomněl.

Poměrně brzy jsem začal tvořit internetové stránky komerčně, jelikož stále přetrvávající internetový boom umožnil získat zakázky na tvorbu stránek i poměrně nezkušeným tvůrcům. Během studia na vysoké škole jsem již celkem pravidelně dostával zprostředkované zakázky od různých lidí.

V zásadě se dá říci, že práce tvůrce internetových stránek je v mnoha směrech jednotvárná. Zjednodušeně by se dal proces tvorby stránek rozdělit do těchto fází:

1) *Vypracování grafického návrhu*
   Design stránek by měl být vždy v rukou profesionálního grafika. Bohužel velké množství programátorů má tendenci dosazovat do této role sebe samé. Většinou nejde ani tak o zlý úmysl, jako spíše o finanční stránku věci.

2) *Nakódování šablon v XHTML a CSS*
   Za posledních pár let se přístup k vytváření šablon velmi změnil. Úzkostlivě se dbá na validitu kódu, striktní oddělení obsahu a formy a s tím související přístupnost (web accessibility). Ačkoliv se XHTML a CSS mohou jevit jednoduše, skutečně profesionální úrovně dosahuje málokdo.

3) *Naprogramování webu v prostředí PHP*
   Programátor se většinou drží svého stylu. Mě se osvědčila metoda umístění společného (neměnného) obsahu do souboru `index.php` a separace sekcí webu do samostatných souborů.

4) *Vytvoření administračního rozhraní pro správu obsahu.*
   Bezesporu nejzdlouhavější a nejúmornější část celého procesu. Nechuť k práci bývá umocněna tím, že si programátor uvědomuje, že už něco velmi podobného několikrát dělal.

Velice brzy jsem si všiml, že určité fragmenty z bodu tři se velice často u různých webů opakují. Bod čtyři je pak téměř pokaždé o tom samém. Byl tedy logický krok začít se zabývat myšlenkou tvorby systému, který by výrazně zefektivnil tvorbu stránek tím, že by zavedl jednotné a znovupoužitelné rozhraní pro administraci obsahu a navíc poskytl tvůrci webu sadu tříd, které by podporovaly tvorbu nejběžnějších součástí webových stránek. Jinými slovy, začal jsem uvažovat o tvorbě vlastního publikačního systému pro WWW.

Z původního záměru se mi zatím nepodařilo splnit všechna předsevzetí. Velice brzy po zahájení prací se totiž ukázalo, že nepůjde jen o znalost problematiky tvorby webových stránek, ale především o znalost návrhu systémů. Bohužel dobrý návrhář se z člověka nestane tím, že si přečte pár knížek. Souhlasím s názorem, že člověk se musí jako návrhář systémů narodit a postupně se v této disciplíně praxí zdokonalovat. Nevím jestli splňuji první podmínku, ale rozhodně mi práce na tomto projektu dala hodně zkušeností a nápadů pro další práci.

Předem mi bylo jasné, že podobný systém by zasluhoval vývojový tým, nikoliv jednoho programátora, ať již je nadšený a odhodlaný sebevíc. Z časového hlediska nebylo možné dokončit více než dva moduly a oku pozorného kritika pravděpodobně neujdou určité nedostatky, za což se předem omlouvám. Systém jsem se ale snažil budovat flexibilně s možností dalšího rozšiřování a práce na něm s odevzdáním absolventské práce rozhodně nekončí.

2 Publisher

2.1 Dva pohledy

Pro pochopení vytvořeného publikačního systému je nutné na něj nahlížet ze dvou pohledů. Z pohledu majitele internetových stránek jde o grafické rozhraní fungující v internetovém prohlížeči, pomocí kterého může spravovat obsah svých stránek.

Z pohledu tvůrce webu jde především o systém, který mu výrazně zjednodušuje práci tím, že vnáší do tvorby internetových stránek v PHP objektové pojetí a téměř zcela ho zbavuje práce při vytváření rozhraní pro správu obsahu.

V tomto dokumentu se zabývám výhradně druhým pohledem. Uživatelská příručka ke grafickému rozhraní bude vytvořena později. Prostředí je ale dostatečně intuitivní, aby bylo možné s ním bez problémů pracovat i bez manuálu.

2.2 Systém

Základním rysem systému je jeho modularita. Samotné jádro systému po programové stránce sestává pouze ze základní sady skriptů a tříd, které zajišťují jeho běh. Jedná se zejména o předdefinované výjimky a třídy pro práci s databází. Podrobněji budou vysvětleny v další kapitole.

Publisher také poskytuje základní kostru grafického rozhraní pro správu obsahu. V horní liště jsou na základě automatického načtení vypsány jednotlivé moduly a jejich ikony. Další bloky jsou prázdné a jejich obsah je v režii aktuálního modulu.

2.3 Konfigurační soubory

Jak samotný Publisher, tak i jednotlivé moduly, používají k uchování konfiguračních parametrů inicializační soubory `configuration.php` ve formátu INI. Kvůli bezpečnosti je použita přípona `.php`, jelikož soubor s příponou `.ini` by byl lehce přístupný a představoval by velké bezpečnostní riziko.

Aby byl inicializační soubor bezpečný a zároveň validní, je na prvním řádku vždy odkomentovaný PHP příkaz `die()`, který zabrání čtení ze souboru přes http server.

Třídy `Publisher` a `Module` disponují metodami pro práci s těmito soubory.

2.4 Třídy

2.4.1 Publisher

Publisher je hlavní třídou celého enginu. Tato třída ovšem není určena pro přímé použití programátorem modulů ani tvůrcem stránek. Při programování se předpokládá existence globálního objektu `$GLOBALS['PUBLISHER']`, který je automaticky vytvořen při inicializaci systému. Žádné další objekty této třídy nesmí být vytvářeny!

2.4.2 Module

Třída Module reprezentuje modul Publisheru. Není určena pro přímé využití programátorem, nicméně její objekt `$GLOBALS['MODULE']` může být použit pro čtení a zápis konfigurace aktuálního modulu.

2.4.3 DatabaseHandler

Všechny přístupy do databáze v prostředí Publisheru se musí realizovat pomocí objektu třídy DatabaseHandler. Tento objekt získá programátor zavoláním metody `getDatabaseHandler()` na globálním objektu Publisheru. Třída `DatabaseHandler` stejně jako `Publisher` není určena k přímému použití programátorem.

`DatabaseHandler` slouží k odstínění programátora modulů a tvůrce webu od specifik databázového serveru. Veškerou komunikaci s databází musí programátor provádět pomocí metod této třídy. Nespornou výhodou tohoto přístupu je možnost okamžitého nasazení jiného typu SQL serveru, například při změně poskytovatele hostingu. Další výhodou je vytvoření jednotné „cesty", přes kterou tečou veškeré databázové požadavky. Programátor se již nemusí starat o připojení k databázi a ošetřování neočekávaných chyb (např. nedostupnost serveru). Aktuálně podporované databázové servery jsou MySQL a PostrgreSQL.

Ukázka použití třídy DatabaseHandler

```
$databaseHandler = $GLOBALS['PUBLISHER']->getDatabaseHandler();
$table = DatabaseHandler::adjustTableName('webparts');

// provede dotaz do databáze
$query = "SELECT id FROM $this->table WHERE type='ROOT'";
$result = $this->databaseHandler->query($query);

// zkontroluje jestli byl nalezen ROOT
if ($result->numRows() == 0) {
        throw new RecordNotFoundException($this->table, "ROOT", "Kořenový záznam
        WebParts nebyl nalezen", 1002);
}
```

2.4.4 QueryResult

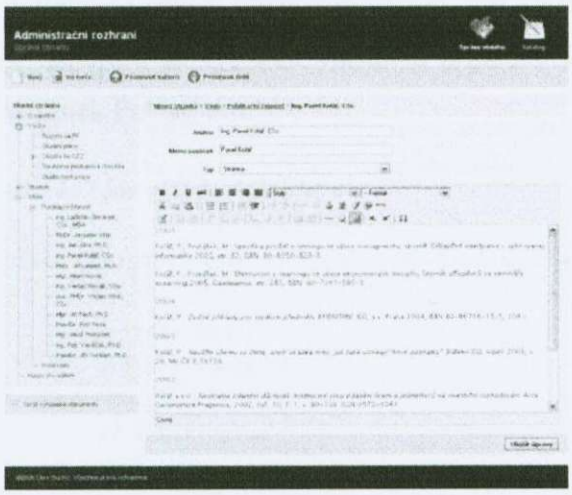
`QueryResult` reprezentuje výsledek SQL dotazu. Objekt této třídy je vracen metodou `query()` třídy `DatabaseHandler`. Tato třída prozatím implementuje pouze základní metody, které ovšem postačují pro většinu běžných databázových operací.

2.4.5 TreeView

Tato třída je pozůstatkem původního záměru vytvořit sadu vizuálních komponent, které by usnadňovaly programátorovi modulů tvorbu uživatelského rozhraní a zároveň sjednocovaly vizuální styl uživatelského prostředí Publisheru. Kvůli časové náročnosti jsem ovšem od tvorby takovéto sady upustil.

Třída `TreeView` slouží k tvorbě a vykreslení (vypsání XHTML kódu) položek uspořádaných do stromové struktury. Konkrétní použití je možné vidět v modulu pro správu obsahu, kde `TreeView` slouží k vytvoření navigačního menu.

Ukázka použití TreeView v modulu pro správu obsahu



2.4.6 TreeViewNode

Tato třída reprezentuje jednu větev ve stromové struktuře TreeView.

2.5 Výjimky

Velkým problémem PHP bylo vždy hledání chyb. Chybová zpráva, kterou PHP vypíše, sice prozradí přesné místo a soubor skriptu, ve kterém došlo k chybě, nicméně u složitějších systémů se nezřídka stalo, že podstata chyby byla zcela jinde, než její následek. Tento nedostatek je ovšem řešitelný.

K ošetření chybových situací v Publisheru slouží výjimky. Práce s výjimkami není pro většinu PHP programátorů zcela běžná, ale při objektovém přístupu jiná metoda práce s chybami prakticky nepřichází v úvahu. Veškeré třídy reprezentující jednotlivé výjimky v Publisheru dědí z třídy `PublisherException`. Ta je postavena na v PHP zabudované třídě `Exception`.

Publisher disponuje několika již vytvořenými výjimkami. Jejich význam bude stručně popsán dále v této kapitole. Pokud bude programátor modulů potřebovat ošetřit chybovou situaci, pro kterou ještě neexistuje výjimka, musí si nadefinovat výjimku vlastní a případně ji nechat zahrnout do základních výjimek Publisheru.

Ukázka kódu výjimky Publisheru

```
class QueryFailedException extends PublisherException
{
        private $query;
        private $dbError;

        // konstruktor
        public function __construct($query, $dbError, $message = "Chyba při SQL
                dotazu", $code = 1001)
        {
```

```
            $this->query = $query;
            $this->dbError = $dbError;
            parent::__construct($message, $code);
        }

        // vrací textovou reprezentaci výjimky
        public function __toString()
        {
            $result = parent::__toString();
            $result .= "SQL dotaz: ".$this->getQuery()."\n";
            $result .= "Chybová zpráva databáze: ".
                    $this->getDbError()."\n";
            return $result;
        }

        public function getQuery()
        {
            return $this->query;
        }

        public function getDbError()
        {
            return $this->dbError;
        }
}
```

2.5.1 Zpracování nezachycených výjimek

Třída `Exception` disponuje velmi užitečnou metodou `getTraceAsString()`, která vypisuje zásobník volání funkcí. Všechny nezachycené výjimky v prostředí Publisheru jsou logovány do adresáře `logs/exceptions/`. To by samo o sobě k vyřešení chybové situace nijak nepřispělo, takže je vždy při výjimce automaticky odeslán e-mail s logem na adresu uvedenou v konfiguračním souboru.

Ukázka logu nezachycené výjimky

```
[2006-04-13, 17:19:07]
Kód: 1001
```

```
Zpráva: Chyba při SQL dotazu
Soubor: C:\www\bakalarka\publisher\classes\DatabaseHandler.php (řádek 180)
#0 C:\www\bakalarka\publisher\modules\catalog\classes\Catalog.php(346):
DatabaseHandler->query('INSERT INTO tea...')
#1 C:\www\bakalarka\publisher\modules\catalog\module.php(60): Catalog-
>addCatalogItem(Object(CatalogItem))
#2 C:\www\bakalarka\publisher\publisher.php(49): require('C:\www\bakalark...')
#3 C:\www\bakalarka\publisher\index.php(2): include_once('C:\www\bakalark...')
#4 {main}

SQL dotaz: INSERT INTO teachers (id, firstName, lastName, titleBefore,
titleAfter, TRIM(CONCAT(titleBefore, ' ', firstName, ' ', lastName,
titleAfter)), room, phone, email, web, consultations, scheduleLink,
showInTeachers, showInConsultations, showInSchedule) VALUES("", "Libor",
"Suchý", "", "", "", "", "", "", "", "", "on", "on", "on")

Chybová zpráva databáze: You have an error in your SQL syntax; check the manual
that corresponds to your MySQL server version for the right syntax to use near
'TRIM(CONCAT(titleBefore, ' ', firstName, ' ', lastName, titleAfter)), room,
phon' at line 1
```

2.5.2 PublisherException

Základní výjimka v publikačním systému. Ostatní výjimky jsou vždy zděděné z této.

2.5.3 SystemErrorException

Chyba při běhu systému.

2.5.4 InvalidMethodCallException

Nepřípustné volání metody. Vyhodí ji kupříkladu metoda `remove()` třídy `WebPart`, pokud dojde k pokusu o vymazání kořenové části.

2.5.5 InvalidArgException

Nepřípustná hodnota argumentu metody. Vstupní argumenty metod by měly vždy procházet alespoň základní kontrolou, aby případné chyby byly snáze odhalitelné.

2.5.6 DatabaseConnectionFailedException

Chyba při připojování k databázi.

2.5.7 QueryFailedException

Chyba při provádění SQL dotazu.

2.5.8 RecordNotFoundException

Databázový záznam nenalezen.

2.5.9 InvalidSettingsFileException

Chyba při parsování konfiguračního souboru.

2.5.10 InvalidSettingValueException

Nepřípustná hodnota nastavení konfiguračního parametru.

2.5.11 SettingNotFoundException

Konfigurační parametr nebyl nalezen

2.5.12 FileNotFoundException

Soubor nebyl nalezen.

2.5.13 IndexOutOfBoundsException

Index je mimo platný rozsah.

2.6 *Způsob dokumentace tříd*

Nedílnou součástí celého systému je dokumentace tříd jednotlivých modulů i samotného Publisheru. Bez kvalitní dokumentace pro tvůrce stránek i programátory modulů by celý projekt ztrácel smysl.

Pro tento účel používá Publisher nástroj phpDocumentor, který je považován za standard v automatické dokumentaci pro jazyk PHP. Tento nástroj dokáže vygenerovat profesionální dokumentaci k třídám čistě na základě speciálních komentářů v kódu. Ideově vychází z javaDocu, což je podobný nástroj pro jazyk Java.

Podrobné informace o použití tohoto nástroje naleznete na stránkách http://www.phpdoc.org/.

Ukázka stylu komentářů pro phpDocumentor

```
/**
 * @author      Libor Suchý libor.suchy@calman.cz
 * @copyright   2006
 * @version     1.0
 * @package     Exceptions
 */

include_once "PublisherException.php";

/**
 * Výjimka indikující selhání při inicializaci modulu
 * @package Exceptions
 */
class ModuleInitFailedException extends PublisherException
{
        /**
         * Identifikátor modulu, jehož inicializace se nezdařila
         * @var string
         */
        private $moduleID;

        /**
         * Konstruktor
         * @param   string   $moduleID   Identifikátor modulu
         * @param   string   $message    Zpráva výjimky
         * @param   int      $code       Číselný kód výjimky
         */
        public function __construct($moduleID, $message = "Inicializace modulu
                se nezdařila", $code = 0)
```

```
        {
                $this->moduleID = $moduleID;
                parent::__construct($message, $code);
        }

        /**
         * Vrací textovou reprezentaci výjimky
         * @return   string   Textová reprezentace výjimky
         */
        public function __toString()
        {
                $result = parent::__toString();
                $result .= "ID modulu: ".$this->getModuleID()."\n";
                return $result;
        }

        /**
         * Vrací identifikátor modulu, jehož inicializace selhala
         * @return   string   Identifikátor modulu
         */
        public function getModuleID()
        {
                return $this->moduleID;
        }
}
```

3 Moduly Publisheru

Jak již bylo zmíněno, Publisher je systém postavený na modulech. V současné verzi existují pouze dva moduly, které řeší základní problémy při vývoji webu, a to správu textového obsahu a správu seznamů (katalogů). Oba moduly budou podrobně vysvětleny v jednotlivých podkapitolách.

Modul Publisheru má pevně určenou strukturu. Každý programátor, který se chystá vyrobit modul, se musí držet následujících pravidel:

1) Veškeré náležitosti modulu musí být umístěny v jednom adresáři. Jméno adresáře je zároveň identifikátorem modulu, který je v systému jedinečný.

2) V adresáři modulu musí být vytvořeny podadresáře:
   classes – Všechny PHP soubory v tomto adresáři budou automaticky načtené jádrem Publisheru.
   scripts – Všechny JavaScripty v tomto adresáři budou automaticky načtené v uživatelském rozhraní.
   styles – Všechny soubory s definicí kaskádových stylů budou automaticky načtené v uživatelském rozhraní.

3) V adresáři modulu musí existovat soubor `configuration.php`, který má standardní, výše popsaný formát. Tento soubor navíc musí obsahovat sekci `ModuleProperties` se systémovými konfiguračními parametry.

Ukázka konfiguračního souboru modulu

```
;<?die()?>
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Povinné systémové parametry ;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
[ModuleProperties]

; Jednoznačný identifikátor modulu
; Musí se shodovat se jménem složky modulu
id = "catalog"

; Název modulu používaný v GUI
name = "Katalog"

; Pořadí modulu v horní liště
order_index = "2"

;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Obecné nastavení modulu ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;
[General]

; ID aktuálního katalogu
currentCatalog = "teachers"
```

4) V adresáři modulu musí existovat soubor `module.php`, který obsahuje hlavní kód.

5) V adresáři modulu musí existovat soubor `panel.content.php`, jehož obsah se vypíše do obsahového bloku v administračním rozhraní.

6) V adresáři modulu může existovat soubor `panel.navigation.php`, jehož obsah se vypíše do levého panelu v administračním rozhraní.

7) V adresáři modulu může existovat soubor `panel.actions.php`, jehož obsah se vypíše do tlačítkové lišty v administračním rozhraní.

8) V adresáři modulu může existovat soubor `panel.heading.php`, jehož obsah se vypíše do lišty pro nadpis v administračním rozhraní.

9) V adresáři modulu může existovat soubor `icon.png`, který bude použit v uživatelském rozhraní v horní liště při výpisu modulů. Standardní velikost obrázku je 48 na 48 pixelů.

10) Dokumentace tříd (viz kapitola 2.4) modulu by měla být umístěna v adresáři documentation.

3.1 Modul WebParts

3.1.1 Popis modulu

Modul WebParts vychází z předpokladu, že strukturu webové stránky lze popsat pomocí stromové struktury. Základem je kořenová položka, která je v systému zabudována a nelze ji odstranit. Ta může obsahovat podřízené položky, které mohou také obsahovat podřízené položky, atd. Stromová struktura je dostatečně obecná, aby vyhovovala široké škále přístupů k organizaci dat na internetových stránkách.

Modul zavádí třídu WebPart, která reprezentuje jednu část (sekci) webu. Základním typem části webu je textový dokument. Druhým možným typem je odkaz.

3.1.2 Třída WebParts

WebParts je hlavní třída, která zaobaluje obsah internetových stránek. Pomocí ní lze získat objekt reprezentující kořenovou položku.

3.1.3 Třída WebPart

Tato třída reprezentuje jednu část webu.

3.2 Modul Catalog

3.2.1 Popis modulu

Modul se snaží v obecné rovině řešit administraci dat v databázových tabulkách. V současné verzi nejsou podporované relace, ale pro většinu případů z praxe je současný přístup dostačující. Na základě XML souboru se vygeneruje administrační rozhraní zahrnující:

1) Seznam položek v tabulce
2) Formulář pro přidání položky
3) Formulář pro editaci položky
4) Prostředí pro správu obrázků přidružených k jednotlivým položkám

Proces generování těchto administračních prvků je plně automatizován a jediné, co se očekává od tvůrce webu, je správná definice jednotlivých katalogů pomocí XML souboru, jehož syntaxe bude popsána v následující kapitole.

3.2.2 Statický přístup

Bylo nutné najít jednoduchý, ale zároveň dostatečně výkonný způsob definice katalogu. Vybral jsem proto formát XML, který má v PHP dobrou podporu v podobě parseru SimpleXML. XML soubor s definicemi katalogů je umístěn v adresáři modulu a nese název `catalogs.xml`. Jednotlivé elementy jsou přímo svázané s třídami modulu, takže přesné vysvětlení lze najít v dokumentaci tříd. V následujících kapitolách vysvětlím strukturu XML souboru pouze zběžně.

3.2.2.1 Definice katalogu

Katalog je definován elementem <catalog> a jeho podřízenými elementy. Atribut id určuje unikátní identifikátor katalogu. Element <name> určuje jméno katalogu. <table> určuje tabulku, se kterou je katalog spojen. Element <condition> nastavuje část SQL dotazu za WHERE, která je použita pro omezení výběru položek z tabulky. Element <orderBy> nastavuje část SQL dotazu za ORDER BY, čímž dochází k seřazení položek katalogu.

Dále v definici následuje několik elementů <imageType> a <atribute>, které souvisejí s popisem položky katalogu a budou vysvětleny dále v textu.

Ukázka definice katalogů XML souborem

```
<?xml version='1.0' standalone='yes'?>
<catalogs>
	<catalog id="teachers">
		<name>Seznam učitelů</name>
		<table>teachers</table>
		<primaryKey>id</primaryKey>
		<condition></condition>
		<orderBy>lastName ASC</orderBy>
		<imagesType id="main">
			...
		</imagesType>
		<imagesType id="others">
			...
		</imagesType>
		<attribute id="firstName">
			...
		</attribute>
		<attribute id="lastName">
			...
		</attribute>
	</catalog>
	...
</catalogs>
```

3.2.2.2 Definice atributu položky katalogu

Atributy položky katalogu úzce souvisejí s jednotlivými sloupci v databázové tabulce. Přiřazují jim ale další informace, pomocí nichž se generuje administrační rozhraní a ošetřují vstupy a výstupy dat. Jednotlivé atributy se v XML souboru definují elementy `<atribute>` podřízenými elementu `<catalog>`.

Každý element reprezentující atribut má několik pod-elementů, které mu určují jeho vlastnosti. Element `<visibility>` může nabývat hodnot `LIST`, `FORM` a `BOTH` a určuje, jestli se daný atribut má zobrazit v seznamu položek katalogu, v editačním formuláři nebo na obou místech. Element `<mandatory>` může nabývat hodnot `true` a `false` a určuje, jestli má být daný atribut položky katalogu považován za povinný, resp. má-li se provádět kontrola nulové délky vstupu v editačním formuláři. Element `<columnName>` určuje jméno sloupce databázové tabulky, se kterým je daný atribut svázán. Element `<caption>` nastavuje popisek atributu v uživatelském rozhraní. Element <type> určuje typ atributu. V současné verzi jsou podporovány typy `TEXT`, `TEXTAREA`, `SELECT`, `BOOLEAN` a `ALIAS`. Bližší informace o jednotlivých typech jsou uvedeny ve vygenerované dokumentaci tříd.

Ukázka definice atributů položky katalogu

```
...
<attribute id="type">
        <visibility>BOTH</visibility>
        <mandatory>true</mandatory>
        <columnName>type</columnName>
        <caption>Typ</caption>
        <type>SELECT</type>
        <options>
                <option id="sale">
                        <databaseValue>sale</databaseValue>
```

```
                <caption>Prodej</caption>
            </option>
            <option id="rent">
                <databaseValue>rent</databaseValue>
                <caption>Pronájem</caption>
            </option>
        </options>
</attribute>

<attribute id="date">
        <visibility>BOTH</visibility>
        <mandatory>true</mandatory>
        <defaultValue type="EVAL">date('Y-m-d');</defaultValue>
        <instructions>Zadejte ve formátu 'RRRR-MM-DD'</instructions>
        <controlMask>([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})</controlMask>
        <columnName>date</columnName>
        <caption>Datum přidání</caption>
        <type>TEXT</type>
        <format>DATE</format>
</attribute>
...
```

3.2.2.3 Definice typu obrázků

Modul Catalog dává možnost přiřazovat k položkám katalogu jeden nebo více obrázků. Je dokonce možné v rámci katalogu nadefinovat neomezené množství různých typů obrázků, u nichž lze nastavit velikost obrázků, velikost náhledů a další vlastnosti.

Uvedu příklad. Firma chce na své stránky umístit katalog triček. U každé položky budou figurovat určité informace, jako například název, cena, velikost, atd. Tuto část řeší již vysvětlené atributy položek katalogu. Dále chce firma u každého trička mít obrázek přední strany, zadní strany a několik fotek modelů či modelek, které dané tričko mají na sobě.

Nadefinujeme tedy tři typy obrázků. První dva budou jedno-obrázkové a třetí bude více-obrázkový. V administračním rozhraní se podle XML definice

automaticky vytvoří editační stránka a žádný další zásah tvůrce webu již není potřeba.

Ukázka definice typů obrázků položek katalogu

```
...
<imagesType id="front">
        <name>Přední strana trička</name>
        <kind>SINGLE</kind>
        <imageDir>../pictures/front</imageDir>
        <imageResizeMethod>FIT_IFLARGER</imageResizeMethod>
        <imageWidth>300</imageWidth>
        <imageHeight>400</imageHeight>
        <thumbnailDir>../pictures/front/thumbnails</thumbnailDir>
        <thumbnailResizeMethod>FIT</thumbnailResizeMethod>
        <thumbnailWidth>90</thumbnailWidth>
        <thumbnailHeight>120</thumbnailHeight>
</imagesType>

<imagesType id="back">
        <name>Zadní strana trička</name>
        <kind>SINGLE</kind>
        <imageDir>../pictures/back</imageDir>
        <imageResizeMethod>FIT_IFLARGER</imageResizeMethod>
        <imageWidth>300</imageWidth>
        <imageHeight>400</imageHeight>
        <thumbnailDir>../pictures/back/thumbnails</thumbnailDir>
        <thumbnailResizeMethod>FIT</thumbnailResizeMethod>
        <thumbnailWidth>90</thumbnailWidth>
        <thumbnailHeight>120</thumbnailHeight>
</imagesType>

<imagesType id="gallery">
        <name>Ukázky</name>
        <kind>COLLECTION</kind>
        <imageDir>../pictures/gallery</imageDir>
        <imageResizeMethod>FIT_IFLARGER</imageResizeMethod>
        <imageWidth>300</imageWidth>
        <imageHeight>400</imageHeight>
        <thumbnailDir>../pictures/gallery/thumbnails</thumbnailDir>
        <thumbnailResizeMethod>FIT</thumbnailResizeMethod>
        <thumbnailWidth>90</thumbnailWidth>
```

```
        <thumbnailHeight>120</thumbnailHeight>
</imagesType>
```

3.2.3 Dynamický přístup

Přístup popsaný v předchozích kapitolách lze považovat za statický, protože tvůrce webu nedefinuje katalogy PHP kódem, ale textově v XML souboru. Pokud je to ale nutné, lze při programování webu použít i dynamický přístup. Struktura XML souboru totiž přesně koresponduje se sadou tříd. Výše popsané definice lze tedy realizovat vytvářením objektů a voláním metod. Ve většině případů lze ale vystačit s XML definicí jak při nastavení administračního prostředí, tak při programování webu.

3.2.4 Třídy modulu Catalog

3.2.4.1 Catalogs

Reprezentuje sadu katalogů nadefinovaných v XML souboru.

3.2.4.2 Catalog

Reprezentuje jeden katalog. V rámci této třídy se realizuje definice katalogu pomocí nastavení atributů a typů obrázků. V případě zavolání konstruktoru s parametrem `$id` se použije k definici katalogu XML soubor. Tato třída také poskytuje přístup k položkám a jejich přidávání do katalogu.

3.2.4.3 CatalogItem

Reprezentuje položku v katalogu. Základními metodami jsou `setData` a `getData`, pomocí kterých se nastavují, respektive čtou hodnoty atributů dané položky. Samotný zápis do databáze se ale realizuje ve třídě `Catalog`. Tato třída také umožňuje manipulaci s obrázky.

3.2.4.4 CatalogItems

Reprezentuje sadu položek katalogu. Implementuje rozhraní `Iterable`.

3.2.4.5 CatalogItemAttribute

Reprezentuje jeden atribut položek katalogu. Třída plní spíše úlohu struktury a její objekty jsou využívány metodami třídy `CatalogItem` a `Catalog` k manipulaci s daty.

3.2.4.6 CatalogItemAttributes

Reprezentuje sadu atributů položek katalogu. Implementuje rozhraní `Iterable`.

3.2.4.7 CatalogItemAttributeOption

Reprezentuje jednu možnou hodnotu atributu typu `SELECT`.

3.2.4.8 CatalogItemAttributeOptions

Reprezentuje sadu hodnot atributu typu `SELECT`.

3.2.4.9 CatalogItemImage

Reprezentuje jeden obrázek položky katalogu. Plní funkci struktury, která nese informace o obrázku, ale neumožňuje přímou manipulaci.

3.2.4.10 CatalogItemImages

Reprezentuje sadu obrázků položky katalogu. Implementuje rozhraní `Iterable`.

3.2.4.11 CatalogItemImagesType

Reprezentuje typ obrázků v rámci katalogu.

3.2.4.12 CatalogItemImagesTypes

Reprezentuje sadu typů obrázků.

4 Použití Publisheru při tvorbě WWW stránek

Abych názorně předvedl praktické využití publikačního systému, rozhodl jsem se vytvořit obdobu stránek katedry informatiky. Originální stránky jsou umístěné na adrese http://www.pf.jcu.cz/stru/katedry/ki. Mnou vytvořenou kopii jsem umístil na adresu http://bakalarka.calman.cz a rozhraní pro administraci obsahu na adresu http://bakalarka.calman.cz/publisher (uživatelské jméno: *user*; heslo: *pass*).

4.1 Vytvoření XHTML + CSS základu

Fázi grafického návrhu v tomto případě opomeneme, jelikož nemá s použitím Publisheru téměř nic společného. Začneme tedy rovnou s vytvářením XHTML šablony. Ta bude v podstatě velice jednoduchá a bude obsahovat pouze několik základních bloků. Prvním blokem je záhlaví stránky, které bude obsahovat logo Jihočeské univerzity a nadpis stránek. Dalším blokem je levý panel, ve kterém bude umístěno menu, případně další navigační prvky. Posledním blokem je samotný obsah. Vytvoření XHTML šablony těchto stránek je velice jednoduché a nezabere více než pár minut.

Dalším krokem je vylepšení vzhledu šablony pomocí CSS stylů. Připomínám důležitost separace obsahu a formy. CSS by vždy mělo přijít na řadu až po dokončení XHTML šablony. Ta by se nikdy neměla podřizovat specifickým požadavkům na vzhled internetových stránek. Dodržení této zásady samo o sobě významně napomáhá k vyhovění kritériím přístupnosti (*web accessibility*).

4.2 Instalace a nastavení Publisheru

Přestože Publisher nedisponuje vlastním instalátorem, je jeho instalace velmi jednoduchá. Nejdříve zkopírujeme celý adresář `publisher` do adresáře webu, respektive tam, kde je umístěn výchozí soubor `index.php`.

Poté otevřeme soubor `publisher/configuration.php` a provedeme nastavení. Všechny parametry jsou v konfiguračním souboru patřičně okomentovány, takže stačí jen v krátkosti zmínit, že jde především o nastavení uživatelského jména a hesla pro přístup do administračního rozhraní a o nastavení přístupových parametrů k databázi.

Dalším krokem je zkopírování modulů, které budou internetové stránky využívat, do adresáře `publisher/modules`. I když v současné době existují pouze dva moduly a oba budou v naší ukázce využity (WebParts a Catalog), považuji za rozumné tento krok zmínit.

4.2.1 Nastavení modulů

Nastavení modulů se provádí obdobně jako nastavení samotného Publisheru, a to pomocí souborů `configuration.php` umístěných v adresářích modulů. V modulu WebParts povolíme editaci kořenové části a povolíme položce s potomky, aby měla vlastní obsah.

Modul catalog žádné důležité konfigurační parametry nemá, ale je potřeba definovat mu soubor katalogů. V našem případě půjde pouze o jeden katalog, a to seznam učitelů. V databázi tedy vytvoříme tabulku pomocí SQL příkazu:

```
CREATE TABLE `teachers` (
  `id` int(10) unsigned NOT NULL auto_increment,
  `firstName` varchar(200) collate utf8_czech_ci NOT NULL default '',
  `lastName` varchar(200) collate utf8_czech_ci NOT NULL default '',
  `titleBefore` varchar(200) collate utf8_czech_ci NOT NULL default '',
```

```
  `titleAfter` varchar(200) collate utf8_czech_ci NOT NULL default '',
  `room` varchar(200) collate utf8_czech_ci NOT NULL default '',
  `phone` varchar(200) collate utf8_czech_ci NOT NULL default '',
  `email` varchar(200) collate utf8_czech_ci NOT NULL default '',
  `web` varchar(200) collate utf8_czech_ci NOT NULL default '',
  `consultations` varchar(200) collate utf8_czech_ci NOT NULL default '',
  `scheduleLink` varchar(200) collate utf8_czech_ci NOT NULL default '',
  `anchorName` varchar(200) collate utf8_czech_ci NOT NULL default '',
  `showInTeachers` tinyint(1) NOT NULL default '1',
  `showInConsultations` tinyint(1) NOT NULL default '1',
  `showInSchedule` tinyint(1) NOT NULL default '1',
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;
```

Konfigurace modulu Catalog pomocí souboru catalogs.xml již byla vysvětlena dříve v textu, takže uvedu pouze výsledek:

```
<?xml version='1.0' standalone='yes'?>
<catalogs>
	<catalog id="teachers">
		<name>Seznam učitelů</name>
		<table>teachers</table>
		<primaryKey>id</primaryKey>
		<condition></condition>
		<orderBy>lastName ASC</orderBy>

		<imagesType id="main">
			<name>Fotka učitele</name>
			<kind>SINGLE</kind>
			<imageDir>../images/teachers</imageDir>
			<imageResizeMethod>FIT_IFLARGER</imageResizeMethod>
			<imageWidth>300</imageWidth>
			<imageHeight>400</imageHeight>
			<thumbnailDir>
				../images/teachers/thumbnails
			</thumbnailDir>
			<thumbnailResizeMethod>FIT</thumbnailResizeMethod>
			<thumbnailWidth>60</thumbnailWidth>
			<thumbnailHeight>80</thumbnailHeight>
		</imagesType>
```

```
                <attribute id="firstName">
                        <visibility>FORM</visibility>
                        <mandatory>true</mandatory>
                        <columnName>firstName</columnName>
                        <caption>Křestní jméno</caption>
                        <type>TEXT</type>
                </attribute>

                <attribute id="lastName">
                        <visibility>FORM</visibility>
                        <mandatory>true</mandatory>
                        <columnName>lastName</columnName>
                        <caption>Příjmení</caption>
                        <type>TEXT</type>
                </attribute>

                <attribute id="titleBefore">
                        <visibility>FORM</visibility>
                        <mandatory>false</mandatory>
                        <columnName>titleBefore</columnName>
                        <caption>Titul před jménem</caption>
                        <type>TEXT</type>
                </attribute>

                <attribute id="titleAfter">
                        <visibility>FORM</visibility>
                        <mandatory>false</mandatory>
                        <columnName>titleAfter</columnName>
                        <caption>Titul za jménem</caption>
                        <type>TEXT</type>
                </attribute>

                <attribute id="fullName">
                        <visibility>LIST</visibility>
                        <mandatory>false</mandatory>
                        <columnName>
TRIM(CONCAT(titleBefore, CHAR(32), firstName, CHAR(32), lastName, titleAfter))
                        </columnName>
                        <caption>Jméno</caption>
                        <type>ALIAS</type>
                </attribute>

                <attribute id="room">
                        <visibility>BOTH</visibility>
```

```
        <mandatory>true</mandatory>
        <columnName>room</columnName>
        <caption>Místnost</caption>
        <type>TEXT</type>
</attribute>

<attribute id="phone">
        <visibility>BOTH</visibility>
        <mandatory>true</mandatory>
        <columnName>phone</columnName>
        <caption>Telefonní číslo</caption>
        <type>TEXT</type>
</attribute>

<attribute id="email">
        <visibility>BOTH</visibility>
        <mandatory>false</mandatory>
        <columnName>email</columnName>
        <caption>E-mail</caption>
        <type>TEXT</type>
</attribute>

<attribute id="web">
        <visibility>FORM</visibility>
        <mandatory>false</mandatory>
        <columnName>web</columnName>
        <caption>WWW stránky</caption>
        <type>TEXT</type>
</attribute>

<attribute id="consultations">
        <visibility>FORM</visibility>
        <mandatory>false</mandatory>
        <columnName>consultations</columnName>
        <caption>Konzultační hodiny</caption>
        <type>TEXT</type>
</attribute>

<attribute id="scheduleLink">
        <visibility>FORM</visibility>
        <mandatory>false</mandatory>
        <columnName>scheduleLink</columnName>
        <caption>Odkaz na rozvrh</caption>
        <type>TEXT</type>
```

```
            </attribute>

            <attribute id="anchorName">
                  <visibility>FORM</visibility>
                  <mandatory>false</mandatory>
                  <columnName>anchorName</columnName>
                  <caption>Jméno XHTML návěstí</caption>
                  <type>TEXT</type>
            </attribute>

            <attribute id="showInTeachers">
                  <visibility>FORM</visibility>
                  <mandatory>false</mandatory>
                  <columnName>showInTeachers</columnName>
                  <caption>Zobrazit v seznamu učitelů</caption>
                  <type>BOOLEAN</type>
                  <defaultValue type="CONST">true</defaultValue>
            </attribute>

            <attribute id="showInConsultations">
                  <visibility>FORM</visibility>
                  <mandatory>false</mandatory>
                  <columnName>showInConsultations</columnName>
                  <caption>Zobrazit v seznamu konzultačních hodin</caption>
                  <type>BOOLEAN</type>
                  <defaultValue type="CONST">false</defaultValue>
            </attribute>

            <attribute id="showInSchedule">
                  <visibility>FORM</visibility>
                  <mandatory>false</mandatory>
                  <columnName>showInSchedule</columnName>
                  <caption>Zobrazit v seznamu rozvrhů</caption>
                  <type>BOOLEAN</type>
                  <defaultValue type="CONST">true</defaultValue>
            </attribute>
      </catalog>
</catalogs>
```

Tímto je konfigurace administračního rozhraní dokončena a po přihlášení je možné se věnovat správě obsahu. V případě těchto stránek jde pouze o

zkopírování struktury a textů z původního webu a přidání jednotlivých učitelů do katalogu. Z demonstračních důvodů jsem oproti původním stránkám přidal možnost nastavit každému pedagogovi v katalogu fotografii. Jelikož jsem neměl k dispozici skutečné fotografie, použil jsem snímky Hollywoodských celebrit.

Modul WebParts (Správa Obsahu)

Modul Catalog (Katalog): Přidání položky

Modul Catalog (Katalog): Výpis položek

Modul Catalog (Katalog): Editace obrázku

4.3 Naprogramování webu

4.3.1 Zavedení prostředí Publisheru

Abychom mohli při programování stránek používat Publisher, je nutné načíst soubor `publisher.php`. To provedeme jednoduše pomocí direktivy `require_once`.

```
require_once 'publisher/publisher.php';
```

Aktuální WebPart budeme jednoduše určovat pomocí GET parametru `dokument`. Pokud tento parametr nebude nastaven, použije se kořenová WebPart. Nesmíme také zapomenout na ošetření možného chybného vstupu (neexistující identifikátor WebPart).

Nastavení aktuální WebPart

```
$webParts = new WebParts();
if (isset($_REQUEST['document'])) {
	try {
		$currentDocument = new WebPart($_REQUEST['document']);
	} catch (Exception $e) {
		$currentDocument = $webParts->getRoot();
	}
} else {
	$currentDocument = $webParts->getRoot();
}
```

4.3.2 Vygenerování menu

Pokud chceme zajistit automatické generování menu bez omezení hloubky, je potřeba použít rekurzivní funkci. V našem případě postačí výpis do třetí úrovně, ale i tak se jeví použití rekurzivní funkce jako nejvhodnější přístup.

Rekurzivní funkce pro výpis menu

```
function writeMenu($root, $level) {
	global $_REQUEST, $currentDocument;
	$currentParent = $currentDocument->getParent();

	// nastavení CSS třídy menu/podmenu
	if ($root->getType() == WebPart::TYPE_ROOT) {
		$menuclass = "mainmenu";
	} else {
		$menuclass = "submenu level".$level;
	}

	// výpis počátečního tagu
	echo '<ul';
	if ($menuclass != "")
		echo ' class="'.$menuclass.'"';
	echo ">\n";

	// výpis jednotlivých položek
	$childCount = $root->getChildCount();
	for ($x=0; $x<$childCount; $x++) {
		$child = $root->getChild($x);
		echo "<li>\n";

		// pokud je typ WebPart LINK, vytvoří se odkaz podle obsahu...
		if ($child->getType() == WebPart::TYPE_LINK) {
			echo '<a href="'.$child->getContent().'">';
			echo $child->getMenuLabel();
			echo "</a>\n";
		}
		// ... jinak se vytvoří odkaz pomocí GET parametru document
		else {
			echo '<a href="index.php?document='.$child->getID().'">';
			echo $child->getMenuLabel();
			echo "</a>\n";
		}

		// trochu krkolomné omezení hloubky výpisu menu
		if ($child->getChildCount() != 0 && (
			$level<1 ||
			$_REQUEST['document'] == $child->getID() ||
			($currentParent != null && $child->getID() ==
				$currentParent->getID())))
```

```
            {
                // rekurzivní volání
                writeMenu($child, $level+1);
            }
            echo "</li>\n";
        }
        echo "</ul>\n";
}
```

Samotný výpis menu pak provedeme takto:

```
$root = $webParts->getRoot();
writeMenu($root, 0);
```

4.3.3 Výpis obsahu

Výpis obsahu bychom mohli jednoduše realizovat výpisem výsledku volání metody `getContent()` na objektu `$currentDocument`. Chceme ale, aby v případě prázdného obsahu došlo k výpisu podřízených částí. I tak bude realizace výpisu velice jednoduchá.

```
$content = $currentDocument->getContent();
$childCount = $currentDocument->getChildCount();

// pokud je prázdný a má podřízené položky
if (trim($content) == "" && $childCount != 0) {?>
<ul>
        <?for ($x=0; $x<$childCount; $x++) {
                $child = $currentDocument->getChild($x);
                if ($child->getType() == WebPart::TYPE_LINK) {?>
        <li><a href="<?=$child->getContent()?>"><?=$child->getName()?></a></li>
                <?} else {?>
        <li><a href="index.php?document=<?=$child->getID()?>">
        <?=$child->getName()?>
        </a></li>
                <?}?>
        <?}?>
</ul>
```

```
<?} else {
        echo $content;
}
```

4.3.4 Výpis katalogu

Modul Catalog použijeme ve třech sekcích. V seznamu vyučujících, v seznamu konzultačních hodin a v odkazech na rozvrhy. Katalogy se budou vypisovat na základě identifikátoru současné WebPart. Proto je nutné označit tři příslušné WebPart v databázi jako zamčené přepnutím parametru `locked`. Toto uzamčení znemožní manipulaci s příslušnými WebPart v administračním rozhraní.

Jelikož se vypisovaný katalog svými atributy v ničem neliší od katalogu definovaného v souboru `catalogs.xml`, můžeme použít dříve zmíněný statický přístup.

Ukázka kódu vypisujícího katalog

```
$catalog = new Catalog("teachers");
$attributes = $catalog->getItemAttributes();
$imagesTypes = $catalog->getImagesTypes();
$mainImage = $imagesTypes->getItemByID("main");
$mainImage->setImageDir("images/teachers");
$mainImage->setThumbnailDir("images/teachers/thumbnails");
$items = $catalog->getCatalogItems();

foreach($items as $teacher) {
        $showInTeachers = $teacher->getData($attributes->
                getItemByID("showInTeachers"));
        if ($showInTeachers == false)
                continue;
        $fullName = $teacher->getData($attributes->getItemByID("fullName"));
        $room = $teacher->getData($attributes->getItemByID("room"));
        $phone = $teacher->getData($attributes->getItemByID("phone"));
        $email = $teacher->getData($attributes->getItemByID("email"));
        $web = $teacher->getData($attributes->getItemByID("web"));
        $image = $teacher->getImages($mainImage)->getItem(0);
```

```
        ?>
<a name="<?=$teacher->getData($attributes->getItemByID("anchorName"));?>"></a>
<div class="teacherDetail">
        <?if ($image != null) {?>
        <img src="<?=$image->getThumbnailPath()?>" alt="foto" />
        <?}?>
        <b><?=$fullName?><br/></b>
        Místnost: <?=$room?><br/>
        Tel.: <?=$phone?><br/>
        <?
        if ($email != "")
                echo '<a href="mailto:'.$email.'">'.$email.'</a><br/>';
        ?>
        <?if ($web != "") echo '<a href="'.$web.'">'.$web.'</a><br/>'?>
</div>
        <?
}
```

Výše uvedená ukázka se může jevit jako nepřehledná. Je to ovšem způsobeno prolínáním PHP kódu a XHTML, kterému se nelze vyhnout. Další dva výpisy katalogu (konzultační hodiny a rozvrhy) jsou velice podobné a nemá smysl je zde uvádět.

Vytvořené stránky: Běžná sekce

Vytvořené stránky: Seznam vyučujících

4.4 Shrnutí náročnosti

Výše uvedeným postupem jsme vytvořili plnohodnotné internetové stránky s širokými možnostmi správy obsahu. Celý proces mi zabral jeden den ne příliš

usilovné práce. Netroufám si odhadovat konkrétní časovou náročnost projektu v případě, že bych neměl k dispozici Publisher. Nárůst by byl ale zjevný a troufám si tvrdit, že by se pohyboval v řádech stovek procent.

Seznam použitých zdrojů

[1] Wikipedia, The Free Encyclopedia:
http://www.wikipedia.org

Převzaté části systému

[1] TinyMCE od Moxiecode Systems
Použit k editaci XHTML textu v modulu WebParts
http://tinymce.moxiecode.com

[2] Image Processor ™ od SmiledSoft.com
Použit k úpravám obrázků v modulu Catalog
http://www.smiledsoft.com/demos/imageproc/index.shtml