Vysoká škola ekonomická v Praze
Fakulta informatiky a statistiky
Vyšší odborná škola informačních služeb v Praze

David Fuchs

Návrh online webové aplikace pracující s podkladovou databází umístěnou na internetovém serveru

Bakalářská práce

2007

Zadávací list

Prohlašuji, že jsem bakalářskou práci na téma návrh online webové aplikace pracující s podkladovou databází umístěnou na internetovém serveru zpracoval samostatně a použil pouze zdrojů, které cituji a uvádím v seznamu použité literatury.

V Praze dne 22. 8. 2007 Podpis

Obsah:

Úvod

Celosvětová síť Internet zažívá v posledním desetiletí velký rozmach. Vděčí za to zejména své službě World Wide Web. V prvopočátcích byly internetové stránky realizovány téměř výhradně statickými soubory umístěnými na serverech. Poté se začaly postupně do popředí dostávat interaktivní webové aplikace. Ty, ať už se jedná o základní stránky nebo sofistikovaná řešení typu elektronických obchodů či firemních intranetů, v drtivé většině využívají ke své činnosti databáze. Podoba samotné databáze, jejího rozdělení do konkrétních souborů apod. nemusí programátora ve většině případů vůbec zajímat. K jejich správě existuje nepřeberné množství více či méně vhodných aplikací.

Od takových by se měl systém navržený v této práci odlišovat jednou základní funkčností. Všechny systémy pro správu databází jsou specializované pro jeden typ databázového serveru. V rámci bakalářské práce vyvinutá aplikace by měla umět komunikovat s více typy databázových serverů, což umožní zejména lidem zabývajícím se po většinu času programováním aplikací jednodušší správu databází, které jejich aplikace využívají, a tím lepší a jednodušší testování vytvořených aplikací. Z toho vyplývá, že tato aplikace není určena ani tak pro specialisty, ale spíše pro vývojáře a osoby se základními znalostmi databází.

Aplikace by měla být navržena jako nezávislá na použité platformě. Proto byla vybrána webová aplikace a skriptovací interpret PHP.

Dalším požadavkem byla jednoduchost práce s aplikací a možnost její použití bez instalace na lokální počítač.

Celá aplikace tedy bude mít charakter webového serveru, kam bude mít možnost přistupovat každý, kdo disponuje internetovým prohlížečem. Základem webových stránek bude značkovací jazyk XHTML. Ten musí dodržovat veškerá pravidla platná pro XML, tzn. správnou strukturovanost a validitu výstupu, což by mělo zaručit stejné zobrazení a vzhled aplikace u různých uživatelů s jinými systémy.

Ovládání by mělo být jednoduché a pro základního uživatele intuitivní, umožňující spravovat všechny důležité prvky databázového serveru.

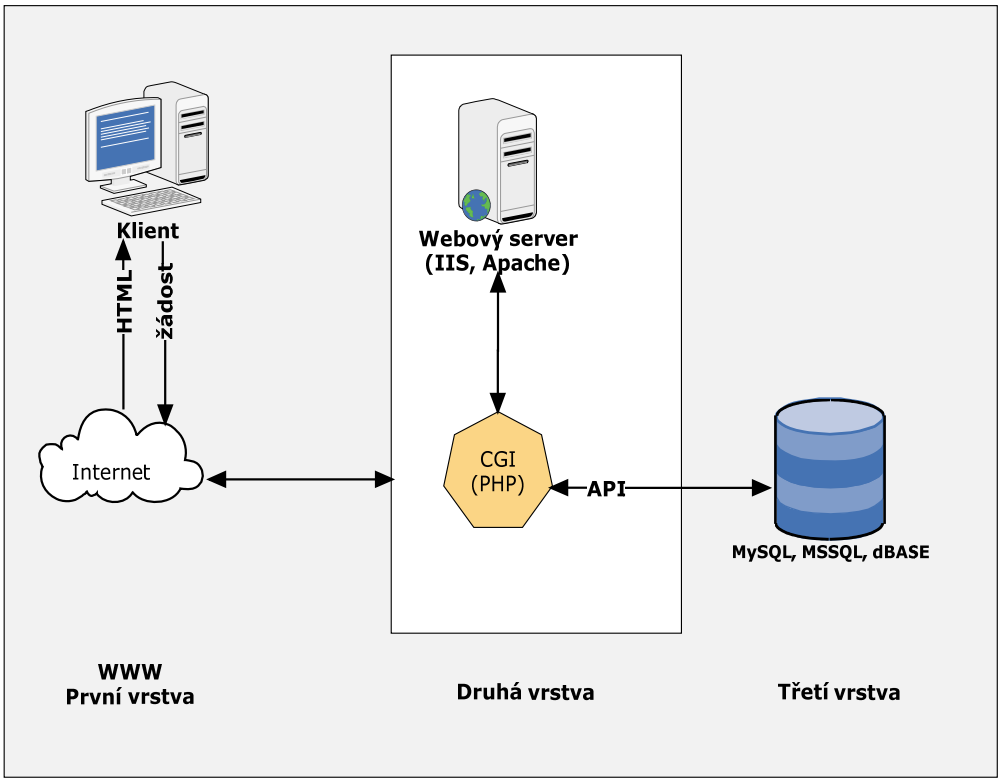
1. Základní popis aplikace

Při vývoji aplikace bylo základem splnit dva hlavní cíle:

1. zpřístupnit správu databázového serveru i méně zkušeným uživatelům,
2. mít možnost spravovat více než jeden typ databázového serveru z jedné aplikace,

a to vše prostřednictvím webové aplikace. Ta se skládá ze tří vrstev:

1. uživatelského rozhraní zobrazeného v prohlížeči běžícím na klientské stanici,
2. webového serveru,
3. databázového serveru.



Obrázek 1 - 3vrstvá architektura

1.1. První vrstva

První vrstvu představuje vlastní klientská stanice s prohlížečem webových stránek. Obsahuje standardní vstupní prvky používané u webových prezentací a aplikací. Grafika a barvy jednotlivých částí aplikace jsou definovány v css souboru.

1.2. Druhá vrstva

Druhá vrstva se skládá z webového serveru a interpretu PHP, jenž pracuje buď jako modul nebo samostatný interpret. Naprogramované skripty jsou použity pro zpracování zadaných údajů do vstupních formulářů a sestavování dotazů pro databázi.

1.3. Třetí vrstva

Tuto vrstvu tvoří vlastní databázový stroj.

Zpracování na straně serveru a generování webových stránek přináší oproti aplikacím na straně klienta mnoho výhod. Mezi hlavní bych zařadil:

- neexistenci problémů s kompatibilitou prohlížečů,
- minimalizaci provozu na síti, neboť potřeba vzájemné komunikace prohlížeče a serveru je značně omezena,
- zrychlení načítání stránek aplikace, protože ve výsledném tvaru se přenáší pouze stránka v HTML kódu,
- poskytnutí dat, které nejsou klientovi k dispozici,
- centrální správa a údržba aplikace.

1.4. Systémové požadavky

Pro běh aplikace na straně serveru je potřeba mít nainstalovaný webový server. Pro platformu Windows to může být například IIS, pro ostatní operační systémy pak Apache, apod.

Na webovém serveru musí být také nainstalován a nakonfigurován interpret jazyka PHP. Ten umí pracovat se všemi možnými prohlížeči typu Internet Explorer, Firefox

a dalšími prohlížeči webu. Tyto prohlížeče bývají součástí OS nebo jsou volně stažitelné jako freeware z internetu. U interpretu musí být v konfiguračním souboru zvolena možnost používat moduly pro práci se zvolenými databázemi.

Pro využitelnost aplikace je také důležité mít přístup k databázovému serveru. Aplikace v nynější podobě umí pracovat s databázemi MSSQL a MySql. Existuje zde ovšem možnost rozšířit jeho funkčnost o další typy databází jako dBase nebo mSQL, a to pouze vytvořením a připojením nové knihovny pro komunikaci s databázovým serverem.

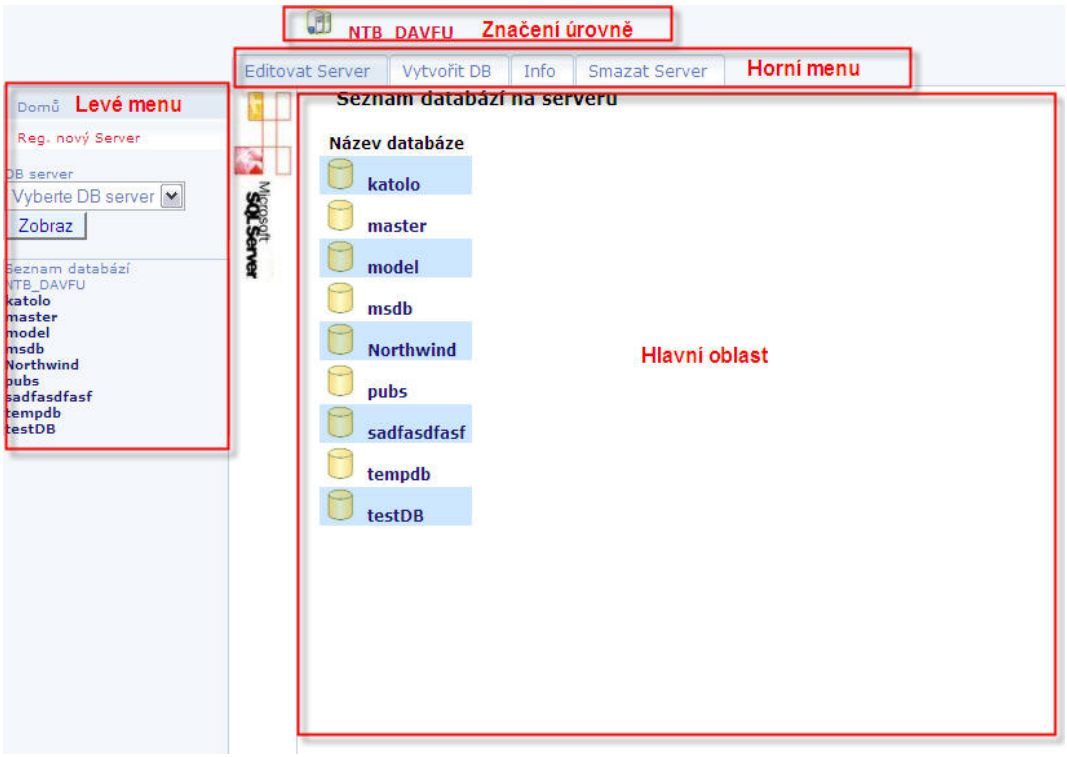
1.5. Instalace

Předpokládejme, že webový server je již zprovozněn. Další podmínkou je přidání podpory interpretu PHP. Ten je podporován na mnoha platformách, a proto je tato aplikace na použité platformě nezávislá. Na Unixových systémech je většinou PHP již jako modul serveru Apache. V případě Windows je nutné vybrat distribuci pro použitý webový server. Jedinou podmínkou na interpret je pouze to, že se musí jednat o verzi, která již podporuje objekty - optimálně od verze 5 výše. PHP je zdarma volně ke stažení na adresách www.php.cz nebo www.php.net.

Celá instalace aplikace pak spočívá pouze v rozbalení archivu do adresáře na webovém serveru.

2. GUI

K zobrazení uživatelského rozhraní, jak již bylo zmíněno, je využit webový prohlížeč.



Obrázek 2 - základní oblasti aplikace

Aplikace se skládá ze 4 hlavních částí.

- Značení úrovně – Zde se uživateli zobrazuje, na které úrovni se právě nachází. Z jednotlivých položek se dá kliknutím přesunout na zpět na nižší.
- Levé menu – Slouží k přístupu k základní navigaci aplikací.
- Horní menu – Toto menu je záložkového typu. Pro každou úroveň je pak specifikována vlastní struktura. Tím jsou uživateli zpřístupněny různé funkce, které lze s danými objekty databáze provádět.
- Hlavní oblast – Do této oblasti se vypisují veškeré výsledky činnosti uživatele a formuláře pro práci s objekty.

Veškeré části aplikace budou ještě podrobněji rozebrány v další části dokumentace.

2.1. Levé menu

Obrázek 3 - levé menu

Tato část obsahuje tlačítko „Domů“ s odkazem na hlavní stranu. Při kliknutí myší na další tlačítko „Reg. nový Server“ se uživatel přesune na formulář pro registraci nového serveru. Pro přímý přístup k zadání vlastního uživatelského SQL dotazu slouží tlačítko „SQL dotaz“, které zobrazí v hlavní oblasti formulář pro jeho zadání a spuštění. Combobox „DB server“ slouží k výběru z již existujících registrací jednotlivých DB serverů a uživatel zde vybírá, ke kterému se chce připojit a se kterým chce pracovat. Ve spodní části je podle úrovně zobrazen seznam jednotlivých databází, popř. databází a její tabulky. Jedním kliknutím na každou z položek se přechází na danou úroveň, a to buď na výpis databáze nebo definici struktury tabulky.

Další vzhled a možnosti rozhraní budou popsány v závislosti na objektu databáze, se kterým uživatel pracuje, respektive podle položek horního menu.

2.2. Horní menu

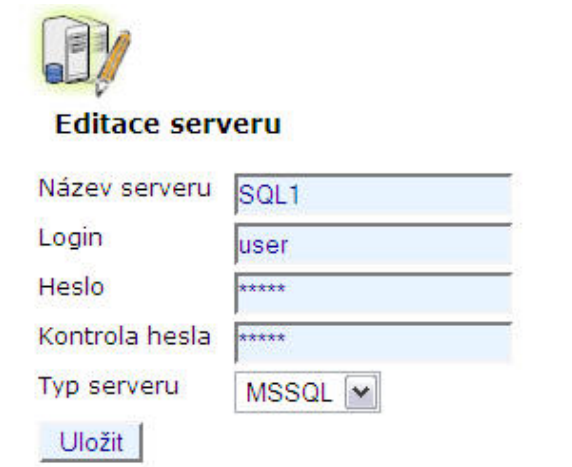
2.2.1. Na úrovni serveru

Obrázek 4 - horní menu - server

Uživateli jsou zobrazena tlačítka pro:

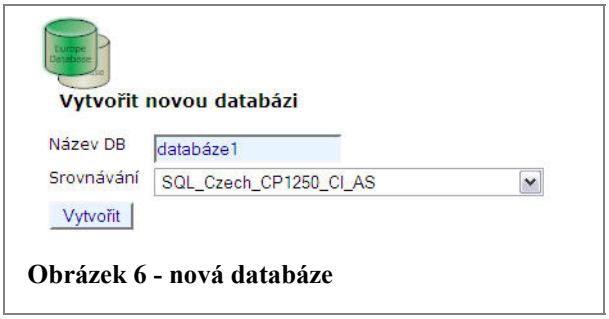
- editaci serveru,
- vytvoření nové databáze na serveru,
- zjištění statistických informací o serveru a stávajícím připojení,
- smazání serveru z nadefinovaných instancí v konfiguračním souboru.

Po kliknutí na editaci serveru se otevře nový formulář. Stejný formulář se otevírá i při registraci nového serveru. Zde musí být vyplněny čtyři základní informace nutné pro připojení k DB serveru. Do políčka název serveru, musí být vyplněn buď síťový název nebo IP adresa počítače, na kterém DB server běží. Login označuje přihlašovací jméno uživatele. Heslo a Kontrola hesla musí být shodné a jedná se o heslo pro přihlášení k instanci serveru. Pokud pro přístup není heslo použito, nechává se nevyplněno. V poslední řadě pak uživatel volí typ serveru, o který se jedná. Stiskem tlačítka se uloží provedené změny.



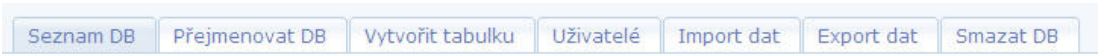
Obrázek 5 - nový / editace serveru

Pro vytvoření nové databáze se objeví tento formulář. Do něj je nutno vyplnit pouze jméno nové databáze a porovnávání, které bude pro ni nastaveno.



Obrázek 6 - nová databáze

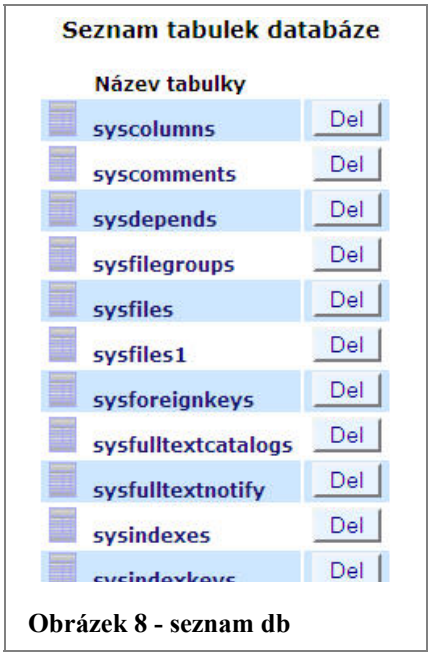
2.2.2. *Na úrovni databáze*



Obrázek 7 - horní menu – databáze

Po přechodu na úroveň správy databáze je uživateli zobrazeno menu s možnostmi:

- zobrazení seznamu databází,
- přejmenování aktuální databáze,
- vytvoření nové tabulky,
- zobrazení seznamu uživatelů,
- obnovu databáze ze zálohy,
- exportu databáze do souborů,
- smazání aktuální databáze.



Obrázek 8 - seznam db

Na této úrovni se uživateli v hlavní oblasti otevře seznam již v ní existujících tabulek . Každou takovou tabulku lze přímo smazat z tohoto seznamu, a to kliknutím na tlačítko „Del“ nebo postoupit na hlubší úroveň kliknutím na název tabulky. V tom případě se zobrazí definice sloupců tabulky.

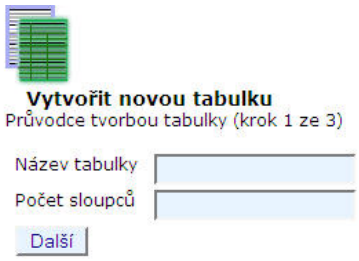
Po kliknutí na přejmenování databáze bude zobrazen formulář „Přejmenovat databázi …..“.

Formulář obsahuje původní název a uživatel musí vyplnit nový, na který se DB přejmenuje.

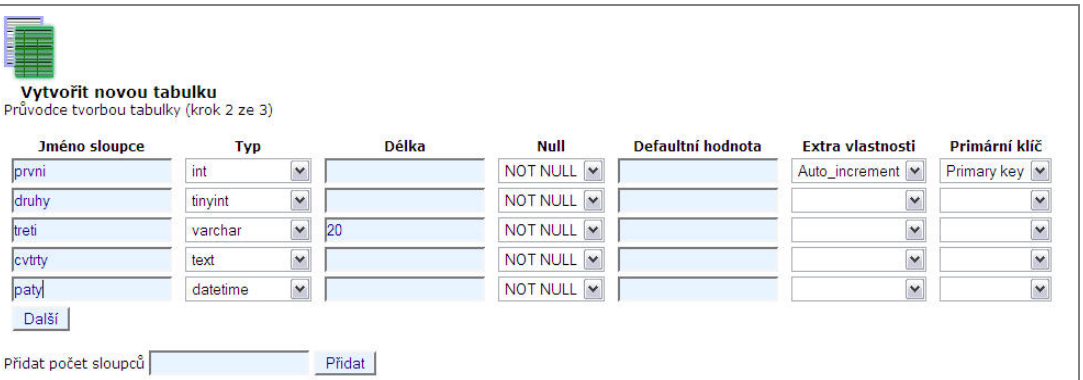
Obrázek 9 - přejmenovat db

Pro vytvoření nové tabulky je připraven průvodce se třemi kroky. V prvním je nutno zadat název nově vytvářené tabulky a počet sloupců, které bude obsahovat.

Stiskem tlačítka „Další“ se dostaneme ke kroku dva. Tady jsou připraveny řádky pro



Obrázek 10 - nová tabulka krok 1



Obrázek 11 - nová tabulka krok 2

jejich definici. V případě, že uživatel v předchozím kroku zadal málo sloupců, může jejich počet zvýšit a připojit pomocí tlačítka „Přidat“.

Obrázek 12 - nová tabulka krok 3

Definice tabulky se skládá ze 7 údajů. Zahrnuje jméno sloupce a jeho typ, délku (tzn. jak velkou oblast bude pole zabírat), zda může obsahovat null hodnotu nebo ne. Pro specifikaci defaultní hodnoty, která se vyplní v případě, že nebude do pole vložen žádný údaj, je ji možno vyplnit do textboxu stejného názvu. V extra vlastnostech lze vybrat, že pole bude v případě vložení nového řádku automaticky navyšovat svou hodnotu a nakonec zda se jedná o primární klíč tabulky nebo ne. Kliknutím na tlačítko další se přesuneme na třetí krok, kde se již jen potvrdí, že uživatel chce nadefinovanou tabulku skutečně vytvořit.

Po kliknutí na položku menu s názvem „Uživatelé“ se dostaneme na správu uživatelských účtů. Je zde vypsán seznam všech uživatelů existujících na serveru. Pro editaci jejich práv slouží tlačítko „Práva“. V případě založení nového uživatele je možné využít políčka pro uživatelské jméno, doménu, heslo a kontrolu hesla. Po odsouhlasení formuláře tlačítkem „Založit“ bude účet na serveru vytvořen.

Obrázek 13 - nový / editace uživatele

Další dvě tlačítka slouží k zálohování a obnovení databáze. V případě obnovy ze zálohy stačí zadat cestu k adresáři nebo souboru, kde se nachází soubory, pro vytvoření zálohy do souborů pak jen cestu k adresáři, kam mají být soubory uloženy.

Obrázek 14 - import dat

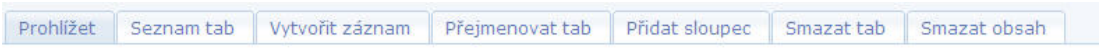
V případě databáze typu MySql je nutno zaškrtnout ještě všechny tabulky určené k záloze.

Obrázek 15 - export dat

Poslední možností v menu je odstranění databáze. Program požádá o potvrzení této akce. Pokud stisknete „Smazat“, bude databáze nenávratně ze serveru smazána.

Obrázek 16 - výmaz databáze

2.2.3. *Na úrovni tabulky*

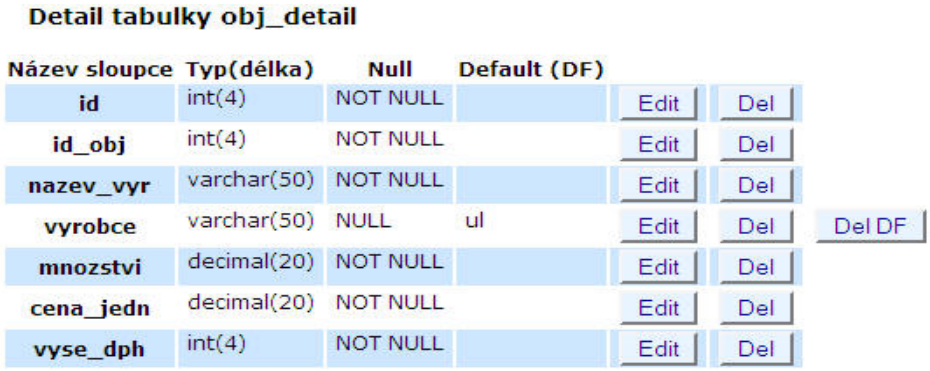


Obrázek 17 - Horní menu – tabulka

V případě tabulky má uživatel možnost:

- prohlížet obsah tabulky,
- zobrazit seznam všech tabulek databáze,
- vytvořit nový záznam do tabulky,
- přejmenovat tabulku,
- přidat sloupec do definice tabulky,
- smazat tabulku,
- odstranit obsah tabulky.

Po přechodu na úroveň tabulky se uživateli zobrazí její definice. V seznamu sloupců



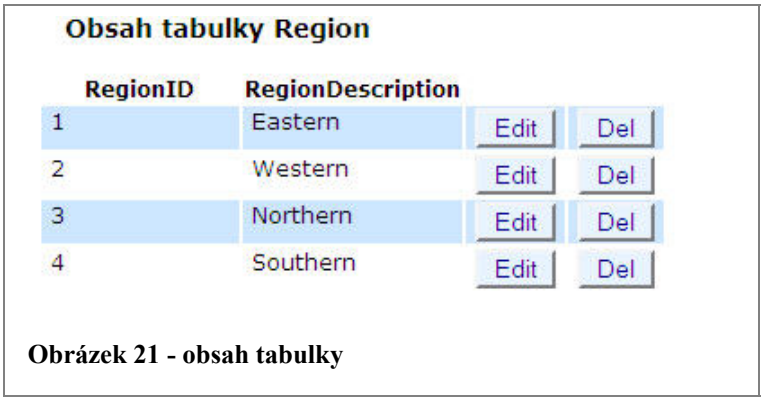
Obrázek 18 - struktura tabulky

lze kliknutím na tlačítko „Edit“ sloupce editovat, pomocí „Del“ smazat. V případě, že se jedná o typ MSSQL, tak se zde objevuje ještě „Del DF“, kterým lze odstranit ze sloupce nastavení defaultní hodnoty. Po stisknutí editačního tlačítka, se zobrazí následující formulář (obrázek 19). Ten obsahuje jednotlivé

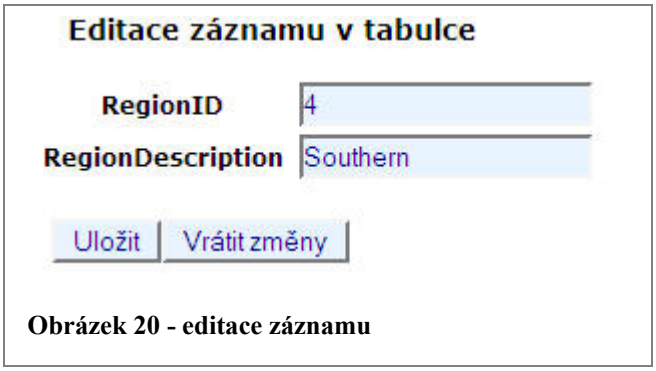
Obrázek 19 - nový / editace sloupce

pole pro specifikaci definice sloupce. Stiskem „Uložit“ se změny v tabulce uloží.

Vybere-li uživatel v menu položku s názvem „Prohlížet“, zobrazí se mu v hlavní oblasti celý obsah tabulky. Jednotlivé záznamy pak může stisknutím tlačítek „Edit“ a „Del“ editovat nebo mazat. V případě editace je uživateli zobrazen tento formulář. Obsahuje seznam všech sloupců, které jsou dány v definici tabulky. Uživatel do jednotlivých polí zapíše požadované hodnoty a stiskem „Uložit“ je zpracuje a vloží.



Obrázek 21 - obsah tabulky



Obrázek 20 - editace záznamu

3. Programovací prostředky

3.1. PHP

Jak jsem již zmínil výše, při tvorbě aplikace byl použit interpretovaný skriptovací jazyk, tzn. že pro provedení skriptu (zdrojového textu) je nutný externí program (tzv. interpreter), který toto vykoná. Vlastní skripty jsou pak následně předány klientovi jako HTML soubor, kam namísto původního textu vepíše výsledek. Klientovi je tak poslán již čistý HTML kód, který byl buď zčásti nebo plně vyprodukován skriptem.

Skriptovací PHP jazyk je poměrně mladý. Vznikl sice již v roce 1994, ale stabilně a širokou částí programátorů začal být používán až s představením verze 3.0., která vznikla v roce 1998.

Autorem PHP a jednou ze současných osob, které se podílejí na jeho vývoji, je Rasmus Lerford.

Dnes je PHP dostupný pro platformy Windows, Linux a Mac OS.

Všechny verze tohoto jazyka jsou uvolněny zcela zdarma použití i samotnému vývoji v něm. Důležitým vylepšením tohoto interpretu bylo, že na platformě 32 bitových Windows ve své verzi 4 začal být dostupný i jako modul (dynamicky připojovaná knihovna .dll) pro Microsoft internet server. To urychluje jeho činnost, neboť se nemusí při každém volání zavádět interpret do paměti znovu. V současné době (polovina roku 2007), je k dispozici již pátá verze s označením 5.2.3.

3.1.1. OOP

Objektové programování je metoda používaná při tvorbě aplikací, která je strukturovanější, přehlednější a více abstraktní než programování klasickým procedurálním způsobem.

Objekt je jednoznačně identifikovatelná entita s vlastnostmi a metodami. Jelikož všechny atributy objektu jsou zapouzdřené uvnitř, nemusí programátor při jeho použití znát vlastní logiku a vnitřní implementaci, ale stačí pouze popis rozhraní, které třída navenek ukazuje. Do jedné třídy tak mohou být umístěny veškeré proměnné a funkce, jež obsluhují nějakou událost. Tím se zpřehlední celý skript, neboť zde nenastává nutnost definování několika funkcí a globálních proměnných, ale pouze třídy, která bude danou událost obsluhovat.

V případě objektů jsou v kódu použity pouze typy veřejných nebo chráněných proměnných a metod:

- chráněné proměnné / metody – jsou uvozeny slovem private a lze k nim tedy přistupovat pouze v rámci toho samého objektu;
- veřejné proměnné / metody – jsou uvozeny slovem publica tím pádem k nim mají přístup a mohou využívat jejich služeb i objekty externí, volat je zvenku.

V jednotlivých třídách se pak mohou vyskytnout ještě speciální metody, zvané konstruktory a destruktory:

- konstruktor – ve třídě vystupuje jako metoda se jménem __construct a je zavolán při každém vytváření nové instance třídy pomocí operátoru new;
- destruktor – pod názvem metody __destruct přichází na řadu v závěru života třídy. Při ukončení její platnosti se provedou veškeré úkony definované v destruktoru a třída zanikne.

3.2. XML

XML je velice efektivní formát, který nachází čím dál více uplatnění v podnikání a při výměně dat. Jeho výhodou je přehlednost a jednoduchost. Tento typ dokumentu je použit pro konfigurační soubor a byl vybrán z toho důvodu, že PHP poskytuje širokou podporu pro práci s ním. S xml je v programu operováno jako s DOM dokumentem, který dovoluje pracovat s ním pomocí DOM API rozhraní.

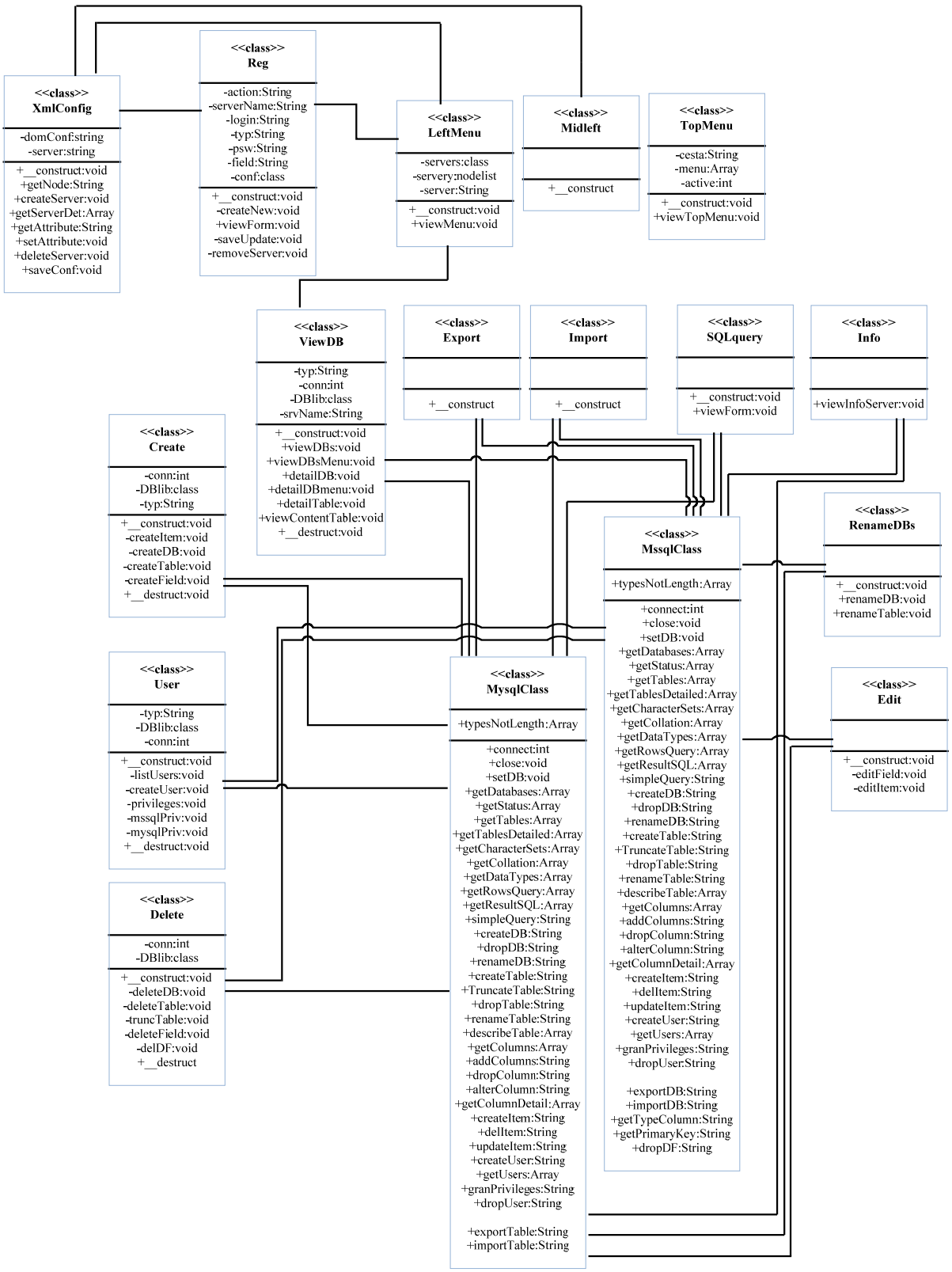
3.3. CSS

Kaskádové styly vznikly kolem roku 1997. Jedná se o kolekce metod, využívaných pro grafickou úpravu webových prezentací. Hlavním úkolem je umožnit návrhářům oddělit to, jak dokument vypadá od jeho kódu a struktury. CSS tak poskytují jednoduchý mechanizmus, jak do takových dokumentů přidat styly (např. fonty, barvy,…).

3.4. SQL

SQL jazyk je základní jazyk pro práci s databázemi. Tento dotazovací jazyk je použit pro logiku metod, které obsahují třídy pro komunikaci s databázovým serverem. Pro získání informací o objektech databáze a dat z tabulek, změny apod. jsou použity buď standardní dotazovací příkazy nebo volání uložených systémových procedur.

4. Schéma aplikace



Obrázek 22 – class diagram

Třídy aplikace lze rozdělit na 3 druhy.

- **Třídy pro práci s xml a konfiguračním souborem**

Tyto třídy, jak již název napovídá, jsou určeny pouze pro práci s xml formátem a konfiguračním souborem. Veškeré změny a dotazy na něj jsou uskutečňovány těmito třídami.

- **Systémové třídy**

Tyto objekty zastřešují funkce tříd pro práci s DB servery. Obsahují metody pro jejich volání a definice uživatelského rozhraní. V případě rozšíření funkcionality o podporu dalších typů databází není nutné dělat složité úpravy těchto knihoven, ale pouze připojit jim nové objekty s logikou pro správu nových typů db.

- **Třídy pro práci s DB servery**

Třídy pro práci s DB servery poskytují rozhraní mezi systémovými třídami a databází.

5. Popis jednotlivých tříd

5.1. Třídy pro práci s konfiguračním souborem

5.1.1. Class XmlConfig – XmlLib.php

Pracuje přímo s xml souborem config.xml.

$domConf

Chráněná proměnná typu DOMDocument, do které je uložena struktura konfiguračního xml souboru.

$server

Chráněná proměnná typu nodelist, ve které jsou uloženy všechny uzly konfiguračního xml dokumenty, specifikované názvem uzlu.

__construct()

Konstruktor třídy XmlConfig. Zjistí, zda existuje xml konfigurační soubor. V případě že ano, načte jeho strukturu do chráněné proměnné $domConf. V případě, že konstruktor soubor v adresáři, ze kterého je skript spouštěn, nenalezne, vytvoří nový se základní strukturou a všemi uzly s definovanými připojovanými knihovnami.

getNode(name)

Veřejná metoda s povinným parametrem typu string. Vrátí objekt typu nodelist, obsahující všechny uzly xml dokumentu, jejichž jméno je definováno v proměnné name.

```
NODELIST getNode( name )
```

createServer(name, attributes)

Veřejná metoda s dvěma povinnými parametry name a attributes. Zapíše novou instanci serveru v konfiguračním souboru.

```
createServer( name, attributes )
name    jméno nové instance DB serveru (string)
attributes      atributy nového uzlu: Jméno serveru, Login, Heslo, Typ serveru (associative array of string)
Typ serveru:    mssql  MSSQL server
                mysql  MySQL server
```

getServerDet(name)

Veřejná metoda, která vrací detaily instance DB serveru specifikovaného proměnnou name. Vrácenou proměnnou je asociativní pole s prvky Name, Login, Psw, Typ.

```
ASSOCIATIVE ARRAY getServerDet( name )
name    jméno instance DB serveru (string)
```

getAttribute(node, attrName)

Veřejná metoda, která zjistí hodnotu atributu určitého uzlu v xml.

```
STRING getAttribute(node, attrName)
node    uzel v XML (node)
attrName        název atributu, jehož hodnotu chceme získat (string)
```

setAttribute(node, attrName, value)

Nastaví hodnotu atributu daného uzlu.

```
setAttribute(node, attrName, value)
node    uzel XML, který obsahuje daný atribut (node)
attrName        název atributu (string)
value   nová hodnota (string)
```

deleteServer(name)

Vymaže instanci DB serveru z konfiguračního souboru.

```
deleteServer(name)
name    jméno instance, kterou chcete smazat (string)
```

saveConf()

Uloží změny v konfiguračním souboru a odpojí dokument.

5.1.2. Class Reg – RegSrvLib.php

Třída zastřešující XmlConfig. Obsahuje grafické rozhraní pro vytvoření, změnu a vymazání instancí serverů v konfiguračním souboru.

$action

Jedná se o chráněnou proměnnou typu string, do které se ukládá URL adresa obsahující akci, jež se má vykonat.

$serverName

Chráněná proměnná typu string s názvem DB serveru.

$login

Chráněná proměnná typu string obsahující uživatelské jméno pro přihlášení k DB serveru.

$typ

Chráněná proměnná typu string, ve které je uložena informace o typu DB serveru.

$psw

Chráněná proměnná typu string s uživatelským heslem nutným pro přihlášení k DB serveru.

$field

Chráněná proměnná typu string s definicí uživatelského ovládacího prvku formuláře.

$conf

Chráněná proměnná obsahující objekt XmlConfig.

$serverNode

Chráněná proměnná typu XmlNode s odkazem na aktuální vybraný uzel XML s registrací DB serveru.

__construct()

Veřejná metoda konstruktoru. Při načtení třídy připojí objekt XmlConfig do proměnné conf. Uloží do proměnné serverNode odpovídající uzel XML a pokud se jedná o existující instanci serveru, naplní proměnné login, typ, psw, field a action.

createNew()

Chráněná metoda, která po zavolání zkontroluje shodnost zadaných hesel při tvorbě zakládání nového DB serveru a zavolá metodu třídy XmlConfig pro fyzické vytvoření instance v konfiguračním souboru.

viewForm()

Veřejná metoda obsahující uživatelské formuláře pro vytvoření nové, změnu nebo vymazání instance DB serveru. Pro rozhodnutí o zobrazení formuláře využívá název proměnné definované v URL adrese.

saveUpdate()

Chráněná metoda zajišťující fyzické uložení změn u vybraného registrovaného DB serveru do konfiguračního souboru.

removeServer()

Chráněná metoda. Při jejím volání je vymazána instance serveru, uzel v XML v konfiguračním souboru.

5.2. Systémové třídy

5.2.1. Class Create - CreateLib.php

Třída zastřešující třídy pro práci s DB serverem. Zde jsou definovány metody týkající se vytváření DB objektů.

$conn

Chráněná proměnná typu int obsahující identifikátor spojení s DB serverem.

$DBlib

Chráněná proměnná s odkazem na objekt zastřešené třídy pro práci s DB serverem.

$typ

Chráněná proměnná typu string obsahující typ DB serveru.

__construct()

Veřejná metoda třídy. Jedná se o konstruktor. Při vytvoření nové instance třídy připojí třídu XmlConfig. V závislosti na typu serveru připojí také knihovnu pro práci s DB serverem a vytvoří spojení se serverem. Tato metoda také rozhoduje o typu vytvářeného objektu a zobrazeném uživatelském rozhraní.

createItem()

Chráněná metoda. V úvodu nastaví spojení na konkrétní databázi na serveru a v závislosti na existenci superglobální proměnné confirm buď provede vytvoření záznamu v tabulce nebo zobrazí formulář pro zádání nového.

createDB()

Chráněná metoda, poskytující možnost vytvoření nové databáze na serveru. Stejně jako metoda createItem se rozhoduje podle existence proměnné confirm, zda vytvořit novou instanci databáze nebo zobrazit formulář pro zadání parametrů nutných pro zakládání DB.

createTable()

Chráněná metoda obsahující formuláře průvodce založením nové tabulky a funkce pro její fyzické uložení do DB.

createField()

Chráněná metoda s grafickým rozhraním pro založení nového sloupce do existující tabulky a funkcemi pro jeho vlastní vložení a update tabulky.

__destruct()

Veřejný destruktor třídy. Při ukončení života třídy uzavře připojení k DB serveru.

5.2.2. Class Delete – DeleteLib.php

Třída, která zastřešuje třídy pro práci s DB serverem. Obsahuje metody pro mazání a odstranění objektů z DB serveru.

$DBlib

Chráněná proměnná s odkazem na objekt zastřešené třídy pro práci s DB serverem.

$conn

Chráněná proměnná typu int obsahující identifikátor spojení s DB serverem.

__construct

Veřejná metoda - konstruktor. Při vytvoření instance třídy se připojí XmlConfig pro zjištění nastavení DB serveru a vytvoří session k dané instanci.

Je zde obsažena logika rozhodování o typu objektu a volání vlastních metod pro práci s ním.

deleteDB(nameDB)

Chráněná metoda s povinným parametrem nameDB, která odstraní danou databázi ze serveru.

```
deleteDB( nameDB )
nameDB        jméno odstraňované databáze (string)
```

deleteTable(nameDB, nameTable)

Chráněná metoda zajišťující vymazání tabulky ze specifikované databáze. Obsahuje také definici formulářů.

```
deleteTable( nameDB, nameTable )
nameDB        jméno databáze, ve které se mazaná tabulka nachází (string)
nameTable     jméno tabulky, která má být smazána (string)
```

truncTable(nameDB, nameTable)

Chráněná metoda, která provádí kompletní vymazání všech záznamů tabulky a grafické rozhraní pro zadání takového požadavku.

```
truncTable( nameDB, nameTable )
nameDB        jméno databáze, ve které se tabulka nachází (string)
nameTable     jméno tabulky, která má být vyprázdněna (string)
```

deleteField(nameDB, table, nameField)

Chráněná metoda. Odstraní daný sloupec z definice tabulky v dané databázi.

```
deleteField( nameDB, table, nameField )
nameDB        jméno databáze, kde je daná tabulka (string)
table    jméno tabulky obsahující sloupec k odstranění (string)
nameField     jméno sloupce, který má být odstraněn z tabulky (string)
```

delDF(nameDB, table, nameField)

Chráněná metoda poskytující službu odstranění omezení defaultní hodnoty ze sloupce tabulky.

delDF(nameDB, table, nameField)
nameDB databáze, obsahující tabulka (string)
table jméno tabulky obsahující sloupec (string)
nameField jméno sloupce, ze kterého má být odstraněna defaultní hodnota (string)

Public __destruct()

Veřejný destruktor třídy. Uzavře připojení k databázi.

5.2.3. Class ViewDB – ViewDBLib.php

$typ

Chráněná proměnná typu string, kde je uložen typ databáze.

$conn

Chráněná proměnná typu int s identifikátorem připojení k serveru.

$DBlib

Chráněná proměnná s odkazem na třídu pro práci s DB serverem.

$srvName

Chráněná string proměnná se jménem serveru.

__construct()

Veřejný konstruktor. Nastaví připojení k DB serveru.

viewDBs()

Veřejná metoda třídy. Obsahuje grafické rozhraní zobrazení jednotlivých databází existujících na serveru

viewDBsMenu()

Veřejná metoda s definicí zobrazení všech databází serveru upravené pro potřeby uživatelského menu.

detailDB(DBname)

Veřejná metoda. Zobrazí seznam všech tabulek, které se nachází v dané databázi.

```
detailDB( DBname )
DBname        název databáze, ze které chceme získat seznam tabulek (string)
```

detailDBMenu(DBname)

Veřejná metoda zobrazující seznam všech dostupných tabulek v databázi, upravený pro potřeby uživatelského menu.

```
detailDBMenu( DBname )
DBname        název databáze,  jejíž detail chceme vypsat (string)
```

detailTable(tableName, db)

Veřejná metoda. Zobrazí uživateli detail - všechna pole, které definice tabulky obsahuje, jejich popis a vlastnosti.

```
detailTable( tableName, db )
tableName     název tabulky, ze které bude vypsán detail (string)
db       název databáze,  kde se nachází tabulka (string)
```

viewContentTable(tableName, db)

Veřejná metoda. Vypíše kompletní obsah tabulky s názvy jednotlivých sloupců.

```
viewContentTable( tableName, db )
tableName     název tabulky, jejíž obsah bude vypsán (string)
db       název databáze,  kde se nachází tabulka (string)
```

__destruct()

Destruktor třídy. Uzavře spojení s DB serverem.

5.2.4. Class User – UserLib.php

Třída určená k zobrazování uživatelů, definování nových a nastavování práv uživatelů k jednotlivým databázím. Obsahuje i definice grafického rozhraní pro podporu těchto funkcí.

$typ

Chráněná proměnná typu string s uloženým typem databáze.

$DBlib

Chráněná proměnná s odkazem na objekt zastřešené třídy pro práci s DB serverem.

$conn

Chráněná proměnná typu int obsahující identifikátor spojení s DB serverem.

__construct()

Veřejná metoda. Konstruktor třídy. Při jejím vzniku nastaví připojení k DB serveru. Podle superglobálních proměnných rozhoduje, zda zobrazit rozhraní pro správu uživatelů nebo přístupových práv

listUsers()

Chráněná metoda obsahující grafické rozhraní pro zobrazení seznamu uživatelů

createUser()

Chráněná metoda. Na základě proměnné confirm rozhoduje, zda zobrazí formulář pro založení nového uživatele nebo zavolá metodu třídy pro práci s DB a fyzicky ho na serveru založí.

privileges()

Chráněná metoda, která plní funkci zastřešující metody pro mssqlPriv a mysqlPriv, kdy rozhoduje, která z nich bude v závislosti na typu serveru použita.

mssqlPriv(), mysqlPriv()

Chráněné metody pro práci s přístupovými právy. Jelikož každý z typů serverů má jinou strukturu práv, musí být definována pro každý typ vlastní metoda s rozhraním odpovídajícím pro jednotlivé typy DB serverů.

__destruct()

Veřejný destruktor třídy uzavírající při jejím zániku spojení s DB.

5.2.5. Class SQLquery – SQLLib.php

Třída s rozhraním pro zadávání a provádění SQL dotazů definovaných uživatelem.

__construct()

Veřejný konstruktor třídy. Při vzniku instance třídy vytvoří spojení s DB serverem. Pokud byl zadán SQL dotaz, zobrazí jeho výsledek do tabulky.

viewForm()

Veřejná metoda. Zobrazí formulář pro zadání nového SQL dotazu.

5.2.6. Class RenameDBs – RenameDBLib.php

Třída poskytující služby pro přejmenování objektů existujících v DB.

__construct()

Veřejný konstruktor třídy. Podle proměnné object vybere vlastní metodu pro zobrazení formuláře. Volá také metody pro vlastní změnu názvu objektů DB. Po provedení všech požadovaných úkonů uzavírá spojení s DB serverem.

renameDB()

Veřejná metoda obsahující definici rozhraní pro změnu názvu objektu databáze.

renameTable()

Veřejná metoda. Jsou zde stanoveny definice formuláře pro změnu názvu tabulky existující v DB.

5.2.7. Class Edit – EditLib.php

Třída zastřešující metody pro editaci záznamů nebo definic tabulek.

__construct()

Veřejná metoda. Konstruktor třídy. Připojí databázi a rozhodne, jestli a který objekt DB bude editován.

editField()

Chráněná metoda. Obsahuje definici grafického rozhraní pro editaci jednotlivých záznamů v tabulce.

editItem()

Chráněná metoda s definicí formuláře pro editaci definice sloupce tabulky.

5.2.8. Class Import – ImpExpLib.php

Třída, která využívá importních služeb knihoven pro práci s DB serverem.

__construct()

Veřejná metoda. Jedná se o konstruktor třídy. V závislosti na typu serveru připojí požadovanou knihovnu pro komunikaci se serverem a zavolá metody obsluhující tuto akci. Obsahuje též formulář pro výběr zálohy, ze které se má DB obnovit.

5.2.9. Class Export – ImpExpLib.php

Třída určená pro export DB ze serveru a jejich zálohu do souboru na filesystem.

__construct()

Veřejný konstruktor třídy. Zobrazí formulář pro výběr objektu, který má být zálohován a pole pro zadání cílového umístění. V závislosti na existenci proměnné confirm buď provede vlastní export nebo zobrazí formulář.

5.2.10.Class Info – InfosLib.php

Třída pro zobrazení základních informací o DB serveru.

viewInfoServer()

Veřejná metoda. Podle typu serveru připojí požadovanou knihovnu pro komunikaci, naváže spojení a zobrazí informace o DB serveru.

5.3. Třídy pro práci s DB servery

5.3.1. Class MysqlClass – MysqlLib.php

5.3.2. Class MssqlClass – MssqlLib.php

Jedná se o třídy zprostředkovávající služby ostatním třídám. Komunikují přímo s DB serverem pomocí SQL dotazů. Pro možnost rozšíření aplikace o nové typy serverů, mají vlastní metody těchto tříd unifikované vstupní a výstupní parametry. Každá třída pak obsahuje ještě další metody, specifické pro daný typ DB serveru, ty však jsou využívány pouze vlastními metodami uvnitř těchto tříd.

Metody a vlastnosti společné pro oba typy DB serverů

V této části popisu se nachází metody a vlastnosti společné pro oba typy serverů. V případě doplnění funkčnosti o nové typy serverů musí vstupní a výstupní parametry, počty a typy proměnných odpovídat definicím metod již existujících tříd pro práci s DB serverem. Logika a struktura dotazů uvnitř jednotlivých metod se může lišit.

$typesNotLength

Veřejná proměnná typu array(of String). Obsahuje výčet všech proměnných, používaných DB servery, u kterých není zadávána délka pole. U těchto typů není při zakládání nové tabulky nebo změny její definice brána v potaz vyplněná hodnota atributu délka.

Pro MySql

```
array('boolean', 'double',' date', 'datetime', 'time',
'tinyblob', 'tinytext', 'mediumblob', 'mediumtext',
'longblob', 'longtext');
```

Pro MSSQL

```
array('sql_variant', 'ntext', 'sysname', 'bit', 'bigint',
'image', 'timestamp', 'money', 'smallmoney', 'int',
'smallint', 'real');
```

connect(server, username, password)

Veřejná metoda. Vytvoří spojení s DB serverem. Všechny vstupní parametry jsou povinné. V případě úspěšného připojení vrací identifikátor spojení, jinak hodnotu false.

```
ID connect(server, username, password)
server  název DB serveru (string)
username        uživatelské jméno (string)
password        heslo (string)
ID      id spojení (int) / false (boolean)
```

close(conn)

Veřejná metoda, která uzavře připojení s identifikátorem conn.

```
close(conn)
conn    identifikátor spojení s DB (int)
```

setDB(nameDB)

Veřejná metoda. Nastaví připojení na specifikovanou databázi.

```
setDB(nameDB)
nameDB          jméno databáze (string)
```

getDatabases()

Veřejná metoda, která zjistí všechny dostupné databáze na serveru.

```
array(Of string) getDatabases()
```

getStatus(connection)

Veřejná metoda pro získání informací o databázovém serveru.

```
assoc_array(Of string) getStatus(connection)
connection      identifikátor spojení s DB (int)
návratové asociativní pole:
infoStat        informace o DB serveru
verzeServeru    verze DB serveru
host    typ a verze protokolu použitá pro připojení k DB
```

getTables(database)

Veřejná metoda, která získá seznam všech tabulek existujících v dané databázi. Návratová hodnota typu array obsahuje názvy tabulek.

```
ARRAY(OF STRING) getTables(database)
database        název databáze, ze které chceme získat seznam tabulek (string)
```

getTablesDetailed(database)

Veřejná metoda určená pro popis tabulek v DB. Vrací 2-rozměrné pole s detailními informacemi o tabulkách.

```
ARRAY(OF ARRAY) getTablesDetailed(database)
database        název DB (string)
```

getCharacterSets()

Veřejná metoda. Při volání vrátí seznam všech použitelných kódových stránek s detailními informacemi o nich, se kterými DB server umí pracovat.

```
ARRAY(OF ARRAY) getCharacterSets()
```

getCollation()

Veřejná metoda vracející pole se všemi použitelnými porovnáváními.

```
Array(Of array) getCollation()
```

getDataTypes()

Veřejná metoda. Vrací pole použitelných datových proměnných.

```
Array(Of string) getDataTypes()
```

getRowsQuery(dotaz)

Veřejná metoda využitá pro SQL dotazy, jejichž výsledkem jsou data. Metoda vrací 2-rozměrné pole pouze s hodnotami vrácenými DB serverem jako odezvu na dotaz.

```
ARRAY(OF ARRAY) getRowsQuery(dotaz)
dotaz   SQL dotaz, který se má vykonat (string)
```

getResultSQL(dotaz)

Veřejná metoda pro SQL dotazy, jejichž výsledkem jsou data a hlavička s názvy sloupců. Vrací asociativní pole se dvěma prvky – hlavička a výsledné hodnoty. Na začátku otestuje metoda dotaz, zda bude vracet nějaké hodnoty. Pokud ne, je dotaz odeslán k vykonání metodě simpleQuery.

```
Assoc-array(Of array) getResultSQL(dotaz)
dotaz   SQL dotaz k vykonání (string)
asociativní pole, které metoda vrací obsahuje:
fields   jednotlivé hlavičky výsledku dotazu (array(Of string))
content          hodnoty vrácené DB serverem (array(Of array))
```

simpleQuery(dotaz)

Tato veřejná metoda je určena k provedení dotazů, které nevrací žádné data. Metoda vrací poslední status provedení příkazu odeslaného DB serverem.

```
STRING simpleQuery(dotaz)
dotaz   SQL příkaz, který se má vykonat (string)
```

createDB(name, collation)

Veřejná metoda provádějící vytvoření nové databáze na serveru. Vrací status o realizaci příkazu.

```
STRING createDB(name, collation)
name   jméno zakládané DB (string)
collation          porovnávání, které se má u DB použít (string)
```

dropDB(name)

Veřejná metoda pro odstranění DB ze serveru. Vrací status o odebrání databáze.

```
STRING dropDB(name)
name   jméno odebírané DB (string)
```

renameDB(oldName, newName)

Veřejná metoda využitá pro přejmenování existující DB na serveru. Návratovou hodnotou je status o přejmenování.

```
STRING renameDB(oldName, newName)
oldName      staré jméno db (string)
newName      nové jméno db (string)
```

createTable(tableName, columns)

Veřejná metoda. Realizuje tvorbu nové tabulky do db, ke které je zrovna nastaveno spojení. Vrací status o vytvoření tabulky.

```
STRING createTable(tableName, columns)
tableName      název nové tabulky (string)
columns        definice sloupců tabulky asociativní pole s prvky:
        name   název sloupce (string)
        type   typ hodnoty v sloupci (string)
        length velikost typu hodnoty (string)
        null   udává, zda je povolena null hodnota (NULL / NOT NULL)
        default definice defaultní hodnoty (string)
        flags  identifikuje, zda bude pole autoincrementální (string)
        primaryKey     identifikace prim. klíče tabulky (string)
```

truncateTable(tableName)

Veřejná metoda poskytující službu vymazání obsahu tabulky. Vrací status provedení.

```
STRING truncateTable(tableName)
tableName      název tabulky (string)
```

dropTable(tableName)

Veřejná metoda pro odebrání specifikované tabulky z db. Vrací status db o úspěšnosti.

```
STRING dropTable(tableName)
tableName      název tabulky (string)
```

renameTable(oldName, newName)

Veřejná metoda pro provedení přejmenování tabulky. Vrací poslední status db.

```
String renameTable(oldName, newName)
oldName      staré jméno tabulky (string)
newName      nové jméno tabulky (string)
```

describeTable(tableName, db)

Veřejná metoda vracející detailní popis tabulky. Návratovou hodnotou je pole obsahující jméno sloupce, typ, velikost, null a defaultní hodnotu.

```
Array(Of string) describeTable(tableName, db)
tableName      název tabulky (string)
db      název databáze, kde se tabulka nachází (string)
```

getColumns(table)

Veřejná metoda využitá pro získání seznamu sloupců tabulky.

```
ARRAY(OF STRING) getColumns(table)
table    název tabulky (string)
```

addColumns(table, columns)

Veřejná metoda pro přidání jednoho nebo více sloupců do definice tabulky. Vrací status provedení změny definice tabulky.

```
String addColumns(table, columns)
table    název tabulky (string)
columns      definice sloupců tabulky asociativní pole s prvky:
        name    název sloupce (string)
        type    typ hodnoty v sloupci (string)
        length  velikost typu hodnoty (string)
        null    udává, zda je povolena null hodnota (NULL / NOT NULL)
        default definice defaultní hodnoty (string)
        flags   identifikuje, zda bude pole autoincrementální (string)
        primaryKey      identifikace prim. klíče tabulky (string)
```

dropColumn(table, nameCol)

Veřejná metoda poskytující odstranění sloupce z tabulky. Návratová hodnota typu string obsahuje status db k akci.

```
STRING dropColumn(table, nameCol)
table    název tabulky (string)
nameCol        název sloupce určeného k odstranění z tabulky (string)
```

alterColumn(table, columns)

Veřejná metoda využitá pro změnu definice sloupce tabulky. Vrací status provedení změny.

```
STRING alterColumn(table, columns)
table    název tabulky (string)
columns        definice sloupců tabulky asociativní pole s prvky:
        name   název sloupce (string)
        type   typ hodnoty v sloupci (string)
        length  velikost typu hodnoty (string)
        null   udává, zda je povolena null hodnota (NULL / NOT NULL)
        default definice defaultní hodnoty (string)
        flags   identifikuje, zda bude pole autoincrementální (string)
        primaryKey     identifikace prim. klíče tabulky (string)
```

getColumnDetail(tableName, column)

Veřejná metoda. Vrátí detail sloupce, který definice tabulky obsahuje. Metoda vrací pole s hodnotami názvu sloupce, typu, null, defaultní hodnoty, automatického zvyšování hodnoty a zda bude pole sloužit jako primární klíč tabulky.

```
ARRAY(OF STRING) getColumnDetail(tableName, column)
tableName      název tabulky (string)
column         název sloupce (string)
```

createItem(tableName, columns, values)

Veřejná metoda zprostředkovávající vložení nových hodnot do tabulky. Vrací status o úspěšném vložení nových záznamů.

```
STRING createItem(tableName, columns, values)
tableName    název tabulky (string)
columns      seznam sloupců tabulky, kam budou hodnoty vkládány (array(Of string))
values       hodnoty, které chceme vložit (array(Of string))
```

delItem(table, heads, items)

Veřejná metoda využitá pro mazání hodnot z tabulky. Vrací status o provedení akce.

```
STRING delItem(table, heads, items)
table    název tabulky (string)
heads    seznam sloupců tabulky (array(Of string))
items    seznam hodnot specifikující, který řádek má být smazán (array(Of string))
```

updateItem(table, heads, itemsOld, itemsNew)

Veřejná metoda. Poskytuje změnu údajů v tabulce. Vrací status uskutečnění změny.

```
STRING updateItem(table, heads, itemsOld, itemsNew)
table      název tabulky (string)
heads      definice sloupců tabulky (array(Of string))
itemsOld   seznam starých hodnot, které mají být změněny (array(Of string))
itemsNew   seznam nových hodnot (array(Of string))
```

createUser(userName, password)

Veřejná metoda pro vytvoření nového uživatele v db. Vrací poslední status db.

```
STRING createUser(userName, password)
userName    nové uživatelské jméno (string)
password    heslo uživatele (string)
```

getUsers()

Veřejná metoda, která získá seznam uživatelů definovaných na db serveru, ke kterému je právě nastaveno spojení.

```
Array(Of string) getUsers()
```

granPrivileges(user, database, privilegesDB, privilegesAll)

Veřejná metoda, kterou lze nastavit uživatelská práva k databázi a tabulkám. Vrací status přiřazení práv specifikovanému uživateli.

```
STRING granPrivileges(user, database, privilegesDB, privilegesAll)
user     uživatelský účet (string)
database         název databáze (string)
privilegesDB   seznam práv k db, které mají být uživateli nastaveny (array(Of string))
privilegesAll    seznam práv ke všem objektům databáze (array(Of string))
```

dropUser(userName)

Veřejná metoda poskytující odstranění uživatele ze serveru. Vrací status provedení operace.

```
STRING dropUser(userName)
userName         uživatelský účet určení k odstranění (string)
```

Metody speciální pro MySql

Tyto veřejné metody jsou vytvořeny speciálně pro DB server MySql. Jelikož neumí exportovat přímo celé databáze, je nutné exportovat do souborů jednotlivé tabulky.

exportTable(tables, adrName)

Metoda umožňující export jednotlivých tabulek db do souborů. Vrací status zálohování.

```
STRING exportTable(tables, adrName)
tables   seznam tabulek určených k exportu (array(Of string))
adrName          cesta k adresáři, kam mají být tabulky zazálohovány (string)
```

importTable(tables, adrName)

Metoda určená pro obnovení tabulek ze zálohy. Vrací status obnovení.

```
STRING importTable(tables, adrName)
tables   názvy tabulek připravených k obnově (array(Of string))
adrName          cesta k adresáři, kde se nachází záloha (string)
```

Metody speciální pro MSSQL

Veřejné metody podporující metody třídy pro komunikaci s MSSQL serverem.

getTypeColumn(tableName, column)

Metoda pro získání typu sloupce. Z vlastních metod využívá jejích služeb updateItem(), kde zjišťuje, zda je pole typu autoincrement, ve kterém je update zakázán. Vrací typ sloupce.

```
STRING getTypeColumn(tableName, column)
tableName       název tabulky (string)
column          název sloupce (string)
```

getPrimaryKey(tableName)

Metoda pro zjištění definice primárního klíče v tabulce. Využito vlastní metodou getColumnDetail(). Vrací název sloupce, který je v tabulce definován jako primární klíč.

```
STRING getPrimaryKey(tableName)
tableName       název tabulky (string)
```

dropDF(tableName, fieldName)

Metoda pro odstranění nastavení defaultní hodnoty z definice pole. Protože MSSQL neumožňuje odebrat pole z definice tabulky s definovanou DF hodnotou, musí být odebrána nejprve ona a až poté pole tabulky. Vrací status po jejím odebrání.

```
STRING dropDF(tableName, fieldName)
tableName       název tabulky (string)
fieldName       název pole (string)
```

exportDB(dbName, path)

Metoda pro export celé databáze do souboru. Vrací status exportu.

```
STRING export(dbName, path)
dbName          název databáze k exportu (string)
path    cesta na filesystem, kam bude uložen soubor (string)
```

importDB(dbName, path)

Metoda umožňující obnovení dané databáze ze zálohy. Vrací status obnovení ze zálohy.

```
STRING importDB(dbName, path)
dbName          název db, která má být obnovena (string)
path     cesta, kde se nachází soubor se zálohou (string)
```

5.4. Třídy uživatelského rozhraní

5.4.1. Class TopMenu – TopMenuLib.php

Třída obsahující definici uživatelského menu v horní části stránky. V závislosti na úrovni, kde se uživatel nachází se zobrazují jednotlivé záložky definované v proměnné menu.

$cesta

Chráněná proměnná obsahující definici pro zobrazení, na které úrovni se uživatel právě nachází.

$menu

Chráněná proměnná s definicí jednotlivých záložek a odkazů. Jedná se o 2-rozměrné asociativní pole s položkami

nazev – zobrazený název v menu,

url – odkaz na skript.

$active

Chráněná proměnná typu integer, jež určuje, které pole menu bude vypsáno jako aktivní.

__construct()

Veřejný konstruktor. V závislosti na existenci superglobálních proměnných s názvy server, DB a table rozhoduje, jaké položky budou v menu zobrazeny.

viewTopMenu()

Veřejná metoda zobrazující menu uživateli.

5.4.2. Class Midleft – MidleftLib.php

Třída obsahující definici rozhodnutí a zobrazení typu serveru, ke kterému je uživatel připojen.

__construct()

Veřejný konstruktor, který pro lepší orientaci uživatele zobrazí odpovídající obrázek s typem serveru.

5.4.3. Class LeftMenu – LeftMenuLib.php

Třída sloužící k zobrazení levého uživatelského menu.

$servers

Chráněná proměnná, ke které je připojena třída XmlConfig.

$servery

Chráněná proměnná typu nodelist, obsahující jednotlivé xml uzly s definovanými DB servery.

$server

Chráněná proměnná typu string, kam je zapsán název připojeného serveru.

__construct()

Veřejný konstruktor. Při volání třídy nastaví jednotlivé hodnoty vlastním proměnným.

viewMenu()

Veřejná metoda. Slouží k zobrazení uživatelského rozhraní se základním menu pro výběr nebo registraci DB serveru a v závislosti na úrovni, na které se uživatel nachází, také seznam databází a tabulek s možností přímého přechodu na jejich detail.

5.5. Ostatní třídy

5.5.1. TestIncludes.php

Skript pro kontrolu, zda existují a jsou použitelné všechny knihovny nutné pro bezproblémový chod programu. V případě, že nebude knihovna nalezena, je tato situace uživateli ohlášena s názvem souboru, který nelze nalézt.

5.6. Základní stránka uživatelského rozhraní

5.6.1. Index.php

Základní skript s definicí layoutu uživatelského rozhraní. Je spouštěn jako první při zadání načtení adresy webového serveru. V kódu jsou obsaženy definice pro html prohlížeč a správné zobrazení.

6. Konfigurační soubor

6.1.1. Config.xml

Konfigurační soubor k aplikaci je vytvořen ve formátu xml. Tento soubor se musí nacházet ve stejné složce jako základní skript aplikace index.php.

Soubor se skládá ze 2 základních uzlů:

servers

Obsahuje seznam nastavení všech registrovaných instancí DB serverů. Každé z nich jsou přiřazeny 4 atributy s údaji pro připojení k DB serveru.

Name	Jméno serveru
Login	Uživatelské jméno pro přihlášení k DB serveru
Psw	Přihlašovací heslo
Typ	Typ DB serveru (MSSQL, MySQL)

including_classes

Tento uzel má základní význam pro běh SQL manageru. Každý jeho potomek představuje jeden soubor s třídami nutnými pro správnou funkci aplikace. Odkaz na tyto soubory je zaznamenán v atributu value a je to cesta, kde se jednotlivé knihovny nachází. Nemusí být proto všechny na jednom místě nebo ve stejné složce jako index.php.

příklad config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<setup>
<servers>
<server Name="server1" Login="user" Psw="heslo" Typ="mysql"/>
</servers>
<including_classes>
<class value="libs/MysqlLib.php"/>
<class value="libs/MssqlLib.php"/>
<class value="libs/RegSrvLib.php"/>
<class value="libs/ViewDBLib.php"/>
<class value="libs/LeftMenuLib.php"/>
<class value="libs/ContentLib.php"/>
<class value="libs/TopMenuLib.php"/>
<class value="libs/RenameDBLib.php"/>
<class value="libs/SQLLib.php"/>
<class value="libs/InfosLib.php"/>
<class value="libs/DeleteLib.php"/>
<class value="libs/CreateLib.php"/>
<class value="libs/UserLib.php"/>
<class value="libs/EditLib.php"/>
<class value="libs/MidleftLib.php"/>
<class value="libs/ImpExpLib.php"/>
</including_classes>
</setup>
```

Závěr

Tato práce si nebrala za úkol vytvořit profesionální řešení pro specialisty, ale umožnit základní možnost správy databázového serveru pro nezkušené uživatele, či ty, kteří nepotřebují složitý a detailní, ale rychlý a přehledný přístup k více serverům a databázím najednou s možností rychlého a jednoduchého přechodu mezi nimi.

Tato aplikace slouží jako skutečný příklad použití technologie PHP pro implementaci přístupu a správy databází prostřednictvím webu. Lze ji použít na jakékoliv platformě, která podporuje grafické uživatelské prostředí. Je jednoduše spravovatelná a rozšiřitelná, neboť stačí upravit skripty na webovém serveru a nenastává zde nutnost přeinstalovat celou aplikaci na klientské stanici. Nevýhoda této volby je sice zřejmá, a to nutnost spravovat webový server. Na druhé straně však umožňuje přistupovat k systému i různě geograficky rozmístěných uživatelů.

Pro jednoduchou správu databázových serverů poskytuje jednoduše použitelné funkce a uživatelsky přívětivé prostředí.

Aplikace umí již v dnešním návrhu spolupracovat společně s databázovými servery MySql a MSSQL, je ovšem relativně jednoduše rozšiřitelná o další typy a to přidáním knihovny pro komunikaci s dalším požadovaným typem databáze.

Seznam obrázků

Seznam použité literatury:

1. Kolektiv autorů, Programujeme PHP profesionálně 2. opravené a aktualizované vydání. Brno: Computer Press, a. s., 2004. 656s. ISBN 80-7226-310-2

2. GUTMANS, Andi, BAKKEN, Stig S. a RETHANS, Derick. Mistrovství v PHP 5. 1. vydání. Brno: CP Books, a.s., 2005. 655 s. ISBN 80-251-0799-X

3. WILLIAMS, Hugh. E. a LANE, David. PHP a MySQL : vytváříme webové databázové aplikace. 1. vydání. Praha: Computer Press, 2002. 530 s. ISBN 80-7226-760-4

4. MySQL funkce [online] 24. 3. 2007 [cit. 15. 6. 2007]. Dostupné z www <http://cz.php.net/manual/cs/ref.mysql.php>

5. Microsoft SQL Server Functions [online] 24. 3. 2007 [cit. 14. 6. 2007]. Dostupné z www <http://cz.php.net/manual/cs/ref.mssql.php>

6. DOM XML Functions [online] 24. 3. 2007 [cit. 2. 8. 2007]. Dostupné z www <http://cz2.php.net/manual/cs/ref.domxml.php>

7. Málek, Vilém. Instalace PHP na Windows [online] 23. 7. 2002 [cit. 12. 5. 2007]. Dostupné z www <http://interval.cz/clanek.asp?article=1355>

8. Funkce pro práci s třídami/objekty [online] 24. 3. 2007 [cit. 20. 6. 2007]. Dostupné z www <http://cz.php.net/manual/cs/ref.classobj.php>

9. Kysela, Jiří. Všechno, co jste chtěli vědět o PHP [online] 18.4. 2000 [cit. 1. 8. 2007]. Dostupné z www <http://interval.cz/clanek.asp?article=198>

10. Fingarfae. Úvod do softwarového inženýrství,aneb co všechno neznáme (1. díl) [online] 16.10.2004 [cit. 5. 8. 2007]. Dostupné z www <http://owebu.cz/filozofie/vypis.php?clanek=385>

11. Kadlec, Václav. Vyvíjíme db aplikaci, 1. díl – úvod [online] 3. 2. 2003 [cit. 6. 8. 2007]. Dostupné z www <http://www.dbsvet.cz/view.php?cisloclanku=2003022001>

12. Kadlec, Václav. Vyvíjíme db aplikaci, 2. díl [online] 20. 2. 2003 [cit. 6. 8. 2007]. Dostupné z www <http://www.dbsvet.cz/view.php?cisloclanku=2003022001>

Poděkování

Zde bych rád poděkoval Ing. Kateřině Jeníčkové za vedení mé bakalářské práce a čas, který jí věnovala.