VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

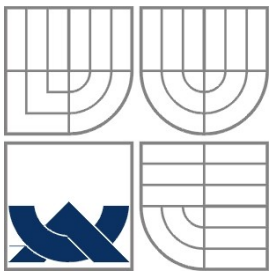
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PROSTŘEDÍ PRO TVORBU INTERAKTIVNÍCH WEBOVÝCH STRÁNEK

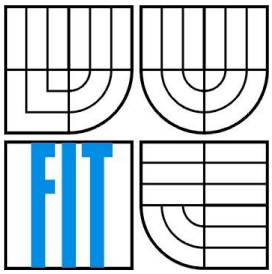
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Jaroslav Moravec
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PROSTŘEDÍ PRO TVORBU INTERAKTIVNÍCH WEBOVÝCH STRÁNEK

INTERACTIVE WEB PAGE DESIGN ENVIRONMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Jaroslav Moravec

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Radek Burget, Ph.D.

BRNO 2008

Abstrakt

Tato diplomová práce popisuje prostředí pro tvorbu a správu interaktivních webových stránek. Zabývá se jak návrhem struktury, tak visuální stránkou. Základní myšlenkou je, že se stránka skládá z jednotlivých prvků a tyto prvky lze na stánku libovolně poskládat. Těchto prvků je několik druhů: interaktivní, obsahové, databázové a informativní. Dále toto prostředí obsahuje nástroje pro správu účtů, řízení přístupu, správu databáze, audit, podporu více jazyků a některé další.

Klíčová slova

webové stránky, webová aplikace, prostředí, webový server, PHP, MySQL, introspekce, refaktorizace, interaktivní prvky, obsahové prvky, databázové prvky, správa účtů, řízení přístupu, IEML,

Abstract

This master's thesis describes an environment for creation and management of interactive web pages. It deals with both the structure design and the visual part. The basic idea is that the page consists of individual elements that can be arbitrarily composed together. There exist several kinds of such elements: interactive, content, database and informative elements. Furthermore, the environment includes tools for account management, access control, database administration, auditing, multi-language support and some more.

Keywords

web page, web application, environment, web server, PHP, MySQL, introspection, refactorization, interactive components, contentual components, database components, account administration, access control, IEML,

Citace

Moravec Jaroslav: Prostředí pro tvorbu interaktivních webových stránek. Brno, 2008, diplomová práce, FIT VUT v Brně.

Prostředí pro tvorbu interaktivních webových stránek

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Radka Burgeta, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jaroslav Moravec
29.12.2007

Poděkování

Velmi rád bych poděkoval vedoucímu mé diplomové práce Ing. Radku Burgetovi, Ph.D. Poděkování si jistě zaslouží i moje rodina a přátelé.

Obsah

1 Úvod

Jednou z nejdůležitějších věcí v dnešním světě jsou informace. Takové informace k člověku proudí ze všech stran ať pravdivé či nepravdivé, úplné či neúplné, strukturované nebo nestrukturované. Jedním z velmi oblíbených a rozšířených tzv. informačních toků je internet. Na internetu lze nalézt velké množství informací ze všech možných oborů, vědních disciplín i zájmových kategorii. Někdy je ale nutné, informace nejen vyhledávat, ale i poskytovat široké veřejnosti. Ať už jde o informace ryze osobního charakteru nebo o informace komerční sloužící k reklamě či propagaci.

V takovém případě má zájemce několik možností. Může se obrátit na odbornou firmu, která mu vyrobí webové stránky na míru, na kterých budou patřičné informace vhodně prezentovány. Toto řešení je profesionální leč velmi drahé. V druhém případě je možné tvorbu přenechat nějakému polo profesionálovi, nebo se po přečtení několika příruček a učebnic o to pokusit sám. V obou těch případech může být výsledek dosti rozpačitý, zvláště nejedná li se pouze o statické webové stránky.

Existuje zde ovšem ještě třetí možnost. Můžeme zakoupit nebo i legálně stáhnout nějaký polotovar, který po malých úpravách a naplněním potřebnými daty vede k dobrému výsledku za rozumnou cenu. Jsou to například všelijaké redakční a publikační systémy. Kvalita těchto systémů je ale u různých produktů velmi odlišná a nabízené možnosti leckdy malé. Prakticky se omezují na tvorbu a editaci článků, správu časových harmonogramů a v lepším případě jsou zde pevně předdefinovány některé prvky stránky jako jsou například ankety. Struktura okna aplikace bývá pevně daná. Je možné měnit pouze barevné složení a velikost prvků stránky.

Z těchto důvodů jsem se rozhodl vytvořit komplexní prostředí pro tvorbu webových aplikací. To by umožňovalo vytvářet a dynamicky měnit všechny prvky stránky, měnit strukturu okna, pracovat s databázovými tabulkami, automaticky generovat formuláře a další výstupy z databáze, vytvářet a modifikovat články, komplexně spravovat obrázky aplikace nebo dokonce vytvářet vlastní strukturu stránek pomocí jednoduchého metajazyku, a to vše bez znalosti programovacích jazyků a technik. Celé toto prostředí a všechny jeho možnosti budou podrobněji popsány v této diplomové práci.

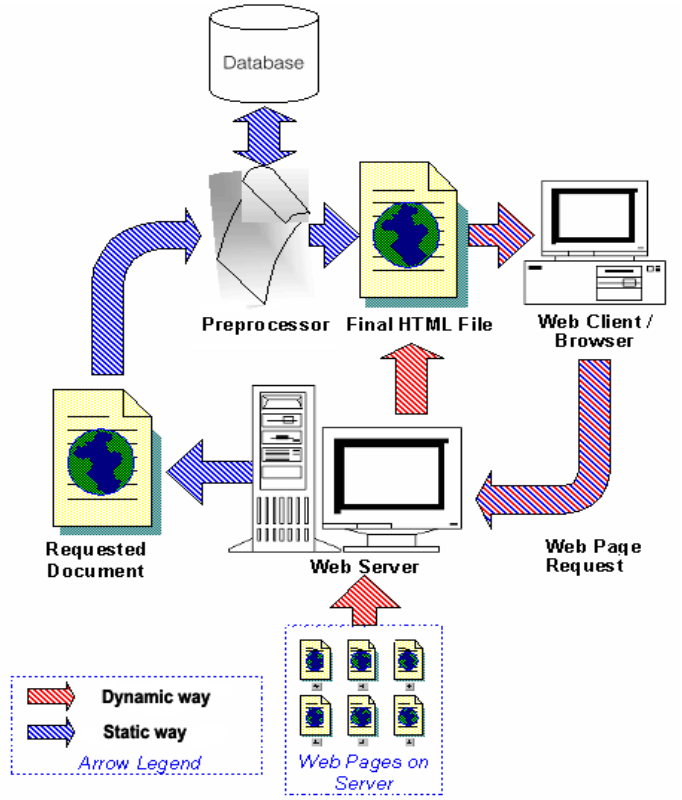
V druhé kapitole bude popsáno vývojové prostředí pro implementaci. Tj. základní principy architektury klient-server, programovací jazyk PHP a databázový server MySQL. Třetí kapitola bude obsahovat popis prostředí pro tvorbu interaktivních webových stránek. V další kapitole zmíním implementaci tohoto prostředí. Poslední kapitola je závěr.

2 Popis cílové platformy

Nejprve je nutné se trochu seznámit s vlastními principy a fungováním internetu a posléze se budeme zaobírat obecnými prostředími pro tvorbu webových aplikaci.

2.1 Principy klient-server

Klient-server je síťová architektura, která odděluje klienta (často aplikaci s grafickým uživatelským rozhraním) a server. Jednotlivé instance klientů komunikují se serverem, který obvykle běží na vzdáleném počítači. Klasickou ukázkou je právě prohlížení webových stránek, kde webový prohlížeč je klient, který při požadavku uživatele na novou stránku kontaktuje vzdálený server a vyžádá si od něj patřičnou webovou stránku. Server tedy poskytuje služby, které na vyžádání „konzumuje“ klient.



Obrázek 2.1: Architektura klient-server [6]

Charakteristika serveru:

- Pasivní
- Čeká na požadavky od klienta
- Při přijetí požadavku jej obslouží

Charakteristika klienta:

- Aktivní
- Posílá požadavky serveru
- Čeká na odpovědi

2.2 Prostředí webového serveru

Jednotlivé webové servery se mohou v různých jednotlivostech značně lišit. Přesto mají několik společných vlastností.

Každý webový server je připojen k počítačové síti a přijímá požadavky ve tvaru HTTP. Tyto požadavky vyřizuje a vrací odpověď počítači, který vznesl požadavek. Odpověď obvykle představuje nějaký HTML dokument. Může to být ale i dokument v jiném formátu - text, obrázek apod. Součástí odpovědi je i tzv. Stavový kód odpovědi. Ten udává, zda byl požadavek vyřízen v pořádku, či zda došlo k nějakým potížím. Běžným stavovým kódem, označující stav OK je 200. Dále jsou to řády:

- 3xx - problémy spojené s přesměrováním
- 4xx - chyby související s vyřízením požadavku (stránka není dostupná, apod.)
- 5xx - interní chyby serveru

(Podrobnější seznam hlášek je v první příloze)

Obvykle server nějakým způsobem protokoluje přijímané požadavky. To pomáhá správci webového serveru vytvářet statistiky a podle typu a množství požadavků optimalizovat obsah, způsob uložení i způsob prezentace požadovaných dat.

Webový server má v zásadě dvě možnosti, jak získávat informace, které vrací klientům. Buď to jsou předem připravené datové soubory (HTML stránky), které webový server bez změny poskytne klientovi, jde o tzv. statický obsah. Nebo teprve na základě požadavku klienta jsou data shromážděna (přečtena ze souboru, databáze, nebo nějakého koncového zařízení), zformátována a připravena k prezentaci ve formátu HTML a poskytnuta webovému prohlížeči. Zde se jedná o tzv. dynamický obsah).[5]

K dynamickému vytváření obsahu se používá celá řada různých technologií (Perl, PHP, ASP, ASP.NET, JSP a další). Statický obsah je schopen server poskytnout významně rychleji než dynamický. Na druhé straně pomocí dynamického obsahu lze poskytovat mnohem větší obsah informací a lze reagovat i na různé dotazy klientů. Proto se v praxi v mnoha případech oba způsoby poskytování obsahu kombinují - například pomocí cachování. Nejrozšířenější programy, které zabezpečují službu webového serveru, jsou *Apache HTTP Server*, *Internet Information Services*, *Sun Java System Web Server*.[5]

Obrázek 2.2: Apache HTTP Server

Obrázek 2.3: Internet Information Services

Obrázek 2.4: Sun Java System Web Server

2.3 Technologie PHP

PHP (Hypertext Preprocessor) řadíme do skupiny skriptovacích jazyků (procesorové instrukce), které se provádějí na straně serveru. PHP je na serveru závislé, protože na něm běží jeho interpretr, který skripty provádí. PHP se tímto odlišuje např. od JavaScriptu, jehož skripty se stahují přímo s HTML stránkou a jsou vykonány na straně klienta jeho prohlížečem.

Má to své výhody i nevýhody. Výhodou PHP v tomto případě je, že se ke zdrojovým kódům skriptů nedostane obvykle nikdo jiný než autor, kdežto ke zdrojovému kódu JavaScriptu se dostane každý, kdo si stáhne HTML stránku, v níž je skript obsažen. JavaScript má výhody ve své možnosti dynamicky reagovat na událost způsobenou klientem (např. pohyb kurzoru myši...), což PHP nedokáže, protože k provedení každé své nové události musí být vždy prohlížečem znovu odeslán požadavek na server. Proto je nejvhodnější variantou kombinovat PHP s JavaScriptem nebo jiným, dynamicky reagujícím jazykem (např. VB Script).

Samotné PHP skripty se zapisují přímo do HTML stránky (nejčastěji s příponou *.php). PHP interpreter na serveru pak pracuje tak, že HTML příkazy rovnou ukládá do výsledné HTML stránky, ale narazí-li na PHP skript, nejprve ho provede, a potom je do HTML stránky zapsán jeho výsledek. To je celý princip dynamického generování HTML stránek, což je základním posláním jazyka PHP. [5]

2.3.1 Co všechno PHP dokáže

PHP dokáže v podstatě všechno, co ostatní skriptovací jazyky pracující na straně serveru (např. CGI, ASP...) - dokáže získávat data z formulářů na HTML stránkách a dále s těmito daty pracovat, může číst a ukládat cookies, dynamicky generovat stránky atd. Jeho nejsilnější parketou je však široká spolupráce s databázemi. Podporuje tyto databáze (MySQL PostgreSQL, MS SQL server, mSQL, Oracle Velocis, Sybase Solid, Adabas D Informix, a další). PHP obsahuje i funkce pro práci se staršími databázovými systémy, např. dBase (DOS a Windows) nebo dbm (UNIX). Rovněž je v PHP podporován u nás málo známý systém FilePro.

Také můžete pracovat s databázemi pomocí rozhraní ODBC (Open DataBase Connectivity). Tato možnost vám přijde vhod, zejména když budete chtít čerpat data např. z MS Access nebo z MS Excelu. PHP může sloužit i jako brána k dalším, na internetu běžně poskytovaným službám, protože obsahuje knihovny některých internetových protokolů, jako je HTTP, FTP, POP3, SMTP, LDAP, SNMP, NNTP atd.[5]

2.3.2 Historie PHP

Jazyk PHP vytvořil v roce 1994 Rasmus Lerdorf, když si naprogramoval v Perlu jednoduché počítadlo přístupů na jeho stránky. Aby spouštění Perlu tolik nezatěžovalo server, přepsal ho do jazyka C. Tento systém se brzy stal populárním, a proto ho autor rozšířil a uvolnil pod názvem Personal Home Page Tools, později Personal Home Page Construction Kit. No a když Lerdorf systém rozšířil i o možnost začleňování SQL příkazů do stránek, práci s formuláři a zobrazování výsledků dotazů SQL, získal systém název PHP/FI 2.0 (Professional Home Page/Form Interpreter verze 2.0). Pod tímto názvem byl už jako jednoduchý programovací jazyk šířen do celého světa. Verze 2.0 však pracovala jen na svém domovském operačním systému, kterým je LINUX (UNIX). Proto bylo vytvořeno PHP 3.0, které již pracuje i na 32-bitových Windows a na operačním systému MACINTOSHE. S verzí 3.0 se upustilo od významu zkratky PHP a systém se dále označuje jako hypertextový preprocesor. V roce 2000 byla uvolněna verze PHP 4.0, která je šířena pod názvem ZEND. Udává se, že tato verze je 8 - 10 x rychlejší než verze předešlá. O čtyři roky později pak verze 5 s vylepšeným objektovým přístupem, podobným jazyku Java.[1]

Verze	***Datum***
PHP 1.0	8. června 1995
PHP 2.0 (PHP/FI)	16. dubna 1996
PHP 3.0	6. června 1998
PHP 4.0	22. května 2000
PHP 4.1	10. prosince 2001
PHP 4.2	22. dubna 2002
PHP 4.3	27. prosince 2002
PHP 4.4	11. července 2005
PHP 5.0	13. července 2004
PHP 5.1	25. listopadu 2005
PHP 5.2	2. listopadu 2006

Tabulka 2.1: Datum vydaní verzí PHP

2.3.3 Základní vlastnosti jazyka PHP

Krátce o vlastnostech PHP. Jazyk PHP je dynamicky typový, tzn. že datový typ proměnné se určí v okamžiku přiřazení hodnoty. Díky tomu má PHP dva typy porovnání, '==' stejný jako v C, a '===' který platí jen když jsou oba dva výrazy stejného typu. Pole jsou heterogenní, mohou tedy obsahovat jakékoli údaje, stejně tak jako jejich indexy. Řetězce lze uzavírat jak do uvozovek (obsah je parsován), tak do apostrofů (obsah není parsován). Syntaxe PHP se velice podobá syntaxi programovacího jazyka C. Příkazy od sebe můžeme oddělit středníkem nebo každý příkaz uvést samostatně mezi TAGy. Dalšími neobjektovými vlastnostmi se zabývat nebudeme. Bližší informace lze nalézt na webu (http://www.php.net).[2]

2.3.4 Objektové vlastnosti PHP

Nyní si více přiblížíme objektovou část jazyka PHP (dále jen OOP). OOP přináší do PHP skriptů celkové zpřehlednění, protože umožňuje definovat tzv. třídy, což jsou speciální datové typy, které mohou obsahovat proměnné (data), ale zároveň i funkce pracující s těmito daty. Do jedné třídy tak můžeme umístit všechny proměnné a funkce obsluhující nějakou událost, čímž dosáhneme zpřehlednění celého skriptu, protože nemusíme definovat několik funkcí a globálních proměnných, ale nadefinujeme si pouze třídu, která bude danou událost obsluhovat. Pokud máme nadefinovanou nějakou třídu, můžeme ve skriptu vytvářet její instance – objekty. Pojmy instance a objekt jsou v podstatě synonyma. Objekt je tedy proměnná typu třída, pomocí níž přistupujeme k datům a funkcím v tomto objektu.[5]

Definice třídy se skládá z klíčového slova *class* následovaného názvem třídy a párem složených závorek. Název třídy může být cokoli s výjimkou vyhrazených slov. Definice jednoduché třídy vypadá takto:

```
class jméno_třídy
{
      definice členských proměnných;
      definice členských funkcí;
}
```

Definice členské proměnné se ve třídách uvozuje klíčovým slovem vyjadřující jeho viditelnost popřípadě další vlastnosti (public, protected, private, static). Identifikátor proměnné začíná symbolem $, stejně jako u procedurálního přístupu. V definici třídy lze rovnou provést inicializaci proměnných. Nyní si nadefinujeme jednoduchou třídu, která bude uchovávat hodnoty proměnných id, title, src popisující obrázek:

```
class picture
{
     private $id;
     public $title = "no title";
     protected $src;
}
```

Nyní již máme vytvořenou třídu, a proto můžeme vytvářet její instance – objekty. Operátor new zajistil alokaci paměti pro proměnnou $P. K jednotlivým členským proměnným třídy se přistupuje pomocí operátoru ->. Další práce s objektem je stejná jako s každou jinou proměnnou.

```
$P = new picture;
$P->id = 0;                      // nelze přistupovat k proměnné
$P->title = "Veverka";           // přiřadí do title hodnotu Veverka
echo $P->src;                    // nelze přistupovat k proměnné
echo $P->title;                  // zobrazí hodnotu Veverka
```

Členské funkce se většinou definují pro manipulaci s daty uvnitř objektu. Definici členské funkce si ukážeme na předchozím příkladu, do kterého doplníme členskou funkci, která bude mít za úkol zobrazit obrázek v HTML. Členské funkce mohou jako každé jiné samozřejmě obsahovat i parametry.

```
class picture
{
     private $id;
     public $title = "no title";
     protected $src;
     public function show()
     {
         echo "<img src='$this->src' title='$this->title' />";
     }
}
```

Konstruktor a destruktor představuje speciální členskou funkci. Konstruktor je zavolán při každém vytváření objektu pomocí operátoru *new*. Destruktor při zrušení instance třídy. Konstruktory se nejčastěji využívají pro snadnější inicializaci objektu. Destruktory pro končení nedodělaného, např. uzavření souboru. Použití vypadá následovně.

```
function __construct( parametry ){ tělo funkce}
function __destruct( parametry ){ tělo funkce}
```

Dědičnost je možnost vytvořit novou třídu, která zdědí členské proměnné a členské funkce z nějaké jiné třídy. V nově vytvořené třídě pak můžeme definovat další členské proměnné a funkce. Opět si vše předvedeme na příkladu, kde bude chtít vytvořit třídu fotografie.

```
class picture
{
     private $id;
     public $title = "no title";
     protected $src;
}
clas foto extends picture  // foto je dceřiná třída třídy picture
{
     private $src = "fotogalery/";
}
```

2.4 MySQL

MySQL je databázový systém, vytvořený švédskou firmou MySQL AB. Jeho hlavními autory jsou Michael „Monty“ Widenius a David Axmark. Je považován za úspěšného průkopníka dvojího licencování. Je k dispozici jak pod bezplatnou licencí GPL, tak pod komerční placenou licencí. MySQL je multiplatformní databáze. Komunikace s ní probíhá, jak už název napovídá, pomocí jazyka SQL. Podobně jako u ostatních SQL databází se jedná o dialekt tohoto jazyka s některými rozšířeními.[7]

2.4.1 Vlastnosti MySQL

Pro svou snadnou implementovatelnost (lze jej instalovat na Linux, MS Windows, ale i další operační systémy), výkon a především díky tomu, že se jedná o volně šiřitelný software, má vysoký podíl v dnes používaných databázích. Velmi oblíbená a často nasazovaná je kombinace MySQL, PHP a Apache jako základní software webového serveru.

MySQL bylo od počátku optimalizováno především na rychlost, a to i za cenu některých zjednodušení: má jen jednoduché způsoby zálohování, a až donedávna nepodporovalo pohledy, triggery, a uložené procedury. Tyto vlastnosti jsou doplňovány teprve v posledních letech, kdy začaly nejčastějším uživatelům produktu (programátorům webových stránek) již scházet.[2]

Vlastnost	***Podpora***
cizí klíče	od verze 3.23 podporovány v tabulkách typu InnoDB
transakce	od verze 3.23 podporovány v tabulkách typu InnoDB
podpora různých znakových sad	od verze 4.1
podpora časových pásem v datech	od verze 4.1
pod dotazy	od verze 4.1
uložené procedury	od verze 5.0
triggery	od verze 5.0
pohledy	od verze 5.0

Tabulka 2.2: Přehled podporovaných vlastností [5]

Obrázek 2.5: Logo MySQL [7]

2.4.2 Typy databázových tabulek

MySQL nabízí několik typů databázových tabulek (storage engine), které se liší svými možnostmi, použitím a způsobem ukládání dat do souborů. Každý typ má své výhody a nevýhody a hodí se pro různé situace. Jedním z hlavních rozdílů je podpora transakcí. Zde jsou zmíněny nejdůležitější typy tabulek.

2.4.2.1 Typ MyISAM

Nejpoužívanější typ tabulek v systému MySQL. Poprvé se objevil v MySQL 3.23.0. Tyto tabulky nepodporují transakce, ale na druhou stranu jsou velmi rychlé a typ MyISAM je optimalizován pro dotazy SELECT. Typ MyISAM se dále rozděluje do tří podtypů. Statické tabulky se vyznačují tím, že neobsahují žádný datový typ s proměnnou délkou (varchar, text, blob a jejich varianty). V tom případě je pro každý údaj v tabulce pevně vyhrazené místo a nedochází ke fragmentaci. I proto jsou všechny dotazy na tuto tabulku mnohem rychlejší. Naproti tomu dynamické tabulky obsahují nějaké datové typy s proměnnou délkou a každý takový údaj musí navíc obsahovat informace o své délce a o svém rozmístění. Dochází také z fragmentaci částí záznamu a opět je navíc na ně třeba uchovávat odkazy. Specialitou jsou komprimované tabulky, které slouží k uložení archivních dat, jejich obsah je zkomprimovaný a není možné do nich přidávat další záznamy ani je editovat.[3]

2.4.2.2 Typ MERGE

Tento typ je jakási nadstavba nad typ MyISAM. Slouží k rozložení zátěže tím, že data rozdělíme do více identických tabulek a každou můžeme např. umístit na jiný pevný disk. Každá tato tabulka má pak svá vlastní data a ke všem se přistupuje pod jejich společným aliasem. Tento typ také může řešit problém s limitem velikosti souborů v některých operačních systémech.[3]

2.4.2.3 Typ HEAP

Zvláštností tohoto typu je, že data se uchovávají pouze v paměti. Z toho plyne, že slouží pouze k dočasnému uchovávání mezivýsledků či často potřebných dat. Co se týče rychlosti, jednoznačně vítězí. Ale protože jsou data jen v paměti, můžeme o ně pádem systému přijít. Je zde však mnoho omezení - nelze použít klíče pro klauzuli ORDER BY, klauzule WHERE má také mnoho omezení, nelze použít sloupce typu BLOB ani TEXT a nelze použít AUTO_INCREMENT.[3]

2.4.2.4 Typ InnoDB

Tyto tabulky používají transakce a lze využívat příkazy BEGIN, COMMIT a ROLLBACK. Jsou optimalizovány pro relativně větší rychlost dotazů INSERT a UPDATE oproti typu MyISAM. Ve stručnosti o transakcích. U některých důležitých změn v databázi musíme zabezpečit, že se sada dotazů vykoná celá anebo se nestane nic. Nesmí se stát, že v polovině příkazů dojde k chybě a nebude vše dokončeno. V té situaci jsou řešením transakce. Transakci zahájíme příkazem BEGIN, vykonáme

příkazy a pak celou transakci buď potvrdíme příkazem COMMIT, ale celou zrušíme příkazem ROLLBACK. Až do potvrzení transakce se změny pro ostatní klienty nijak neprojeví.[3]

2.4.2.5 Typ BDB (Berkeley Database)

Taktéž podporuje transakce, ale rozhraní pro MySQL je prozatím ve fázi beta a není doporučeno je příliš využívat.[3]

2.4.3 Pohled (databáze)

Poměrně žhavou novinkou v MyQSL jsou pohledy, proto zde budou pár slovy popsány. Pohled (anglicky View) je databázový objekt, který uživateli poskytuje data ve stejné podobě jako tabulka. Na rozdíl od tabulky, kde jsou data přímo uložena, obsahuje pohled pouze předpis, jakým způsobem mají být data získána z tabulek a jiných pohledů. Z toho vyplývají některé základní rozdíly v chování tabulky a pohledu:

1. Data tabulky lze přímo modifikovat pomocí příkazů DML SQL (*INSERT*, *UPDATE*, *DELETE*). Data poskytovaná pohledem nelze přímo modifikovat - výsledek, který pohled poskytuje, se změní v případě, že se změní data v tabulkách, ze kterých pohled čerpá.
2. Na rozdíl od tabulky pohled nezabírá v paměti téměř žádné místo, protože neobsahuje data, ale pouze předpis pro získání dat (obvykle příkaz *SELECT*).
3. Získání výsledku především u komplikovanějších pohledů může být časově výrazně náročnější než u přímého přístupu k tabulce, neboť data musí být při každém použití pohledu získána z podkladových pohledů a tabulek (výpočet může být často dost složitý). Tento problém lze řešit vytvořením vhodných indexů na podkladových tabulkách. Druhou možností používanou především u agregačních pohledů v datových skladech je použití „hybridu“ mezi tabulkou a pohledem - tzv. materializovaného pohledu.

Stejně jako ostatní databázové objekty, je i pohled modifikován pomocí příkazů DDL SQL (CREATE,ALTER,DROP).[5]

2.4.4 Dotazovací jazyk SQL

SQL je standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích. SQL je zkratka anglických slov Structured Query Language (strukturovaný dotazovací jazyk). V 70. letech 20. století probíhal ve firmě IBM výzkum relačních databází. Bylo nutné vytvořit sadu příkazů pro ovládání těchto databází. Vznikl tak jazyk SEQUEL (Structured English Query Language). Cílem bylo vytvořit jazyk, ve kterém by se příkazy tvořily syntakticky co nejblíže přirozenému jazyku (angličtině).

K vývoji jazyka se přidaly další firmy. V r. 1979 uvedla na trh firma Relational Software, Inc. (dnešní Oracle Corporation) svoji relační databázovou platformu Oracle Database. IBM uvedla v roce

1981 nový systém SQL/DS a v roce 1983 systém DB2. Dalšími systémy byly např. Progress, Informix a SyBase. Ve všech těchto systémech se používala varianta jazyka SEQUEL, který byl přejmenován na SQL.

Relační databáze byly stále významnější, a bylo nutné jejich jazyk standardizovat. Americký institut ANSI původně chtěl vydat jako standard zcela nový jazyk RDL. SQL se však prosadil jako de facto standard a ANSI založil nový standard na tomto jazyku. Tento standard bývá označován jako SQL-86 podle roku, kdy byl přijat.

V dalších letech se ukázalo, že SQL-86 obsahuje některé nedostatky a naopak v něm nejsou obsaženy některé důležité prvky týkající se hlavně integrity databáze. V roce 1992 byl proto přijat nový standard SQL-92 (někdy se uvádí jen SQL2). Zatím nejnovějším standardem je SQL3 (SQL-99), který reaguje na potřeby nejmodernějších databází s objektovými prvky.

Standardy podporuje prakticky každá relační databáze, ale obvykle nejsou implementovány vždy všechny požadavky normy. A naopak, každá z nich obsahuje prvky a konstrukce, které nejsou ve standardech obsaženy. Přenositelnost SQL dotazů mezi jednotlivými databázemi je proto omezená. [5]

SQL příkazy se dělí do čtyř základních skupin.

2.4.4.1 Příkazy pro manipulaci s daty

Jsou to příkazy pro získání dat z databáze a pro jejich úpravy. Označují se zkráceně DML - Data Manipulation Language („jazyk pro manipulaci s daty“).

- *SELECT* – vybírá data z databáze, umožňuje výběr podmnožiny a řazení dat.
- *INSERT* – vkládá do databáze nová data.
- *UPDATE* – mění data v databázi (editace).
- *DELETE* – odstraňuje data (záznamy) z databáze.
- *EXPLAIN PLAN FOR* – speciální příkaz, který zobrazuje postup zpracování SQL příkazu. Pomáhá uživateli optimalizovat příkazy tak, aby byly rychlejší.
- *SHOW* - méně častý příkaz, umožňující zobrazit databáze, tabulky nebo jejich definice [7]

2.4.4.2 Příkazy pro definici dat

Těmito příkazy se vytvářejí struktury databáze – tabulky, indexy, pohledy a další objekty. Vytvořené struktury lze také upravovat, doplňovat a mazat. Tato skupina příkazů se nazývá zkráceně DDL – Data Definition Language („jazyk pro definici dat“).

- *CREATE* – vytváření nových objektů.
- *ALTER* – změny existujících objektů.
- *DROP* – odstraňování objektů. [7]

2.4.4.3 Příkazy pro řízení dat

Do této skupiny patří příkazy pro nastavování přístupových práv a řízení transakcí. Označují se jako DCL – Data Control Language („jazyk pro ovládání dat“), někdy také TCC – Transaction Control Commands („jazyk pro ovládání transakcí“).

- *GRANT* – příkaz pro přidělení oprávnění uživateli k určitým objektům.
- *REVOKE* – příkaz pro odnětí práv uživateli.
- *BEGIN* – zahájení transakce.
- *COMMIT* – potvrzení transakce.
- *ROLLBACK* – zrušení transakce, návrat do původního stavu. [7]

2.4.4.4 Ostatní příkazy

Do této skupiny patří příkazy pro správu databáze. Pomocí nich lze přidávat uživatele, nastavovat systémové parametry (kódování znaků, způsob řazení, formáty data a času apod.). Tato skupina není standardizována a konkrétní syntaxe příkazů je závislá na databázovém systému. V některých dialektech jazyka SQL jsou přidány i příkazy pro kontrolu běhu, takže lze tyto dialekty zařadit i mezi programovací jazyky.[7]

2.5 Shrnutí použitých technologii v projektu

Na konec je nutné uvést kompletní výčet použitých technologii potřebných k implementaci tohoto projektu. Jak bylo výše uvedeno, celá aplikace poběží na Apache2 HTTP Serveru s programovacím jazykem PHP 5 (2.2.3 –Technologie PHP). Jako databázový server je použit MySQL 5 (2.2.4 - MySQL) kompatibilní s webovým serverem. Dále značkovací jazyky HTML a XHTML (2.1 – Principy klient-server), skriptovací jazyk JavaScript a jazyk pro tvorbu kaskádových předloh CSS.

3 Návrh prostředí pro tvorbu IWS

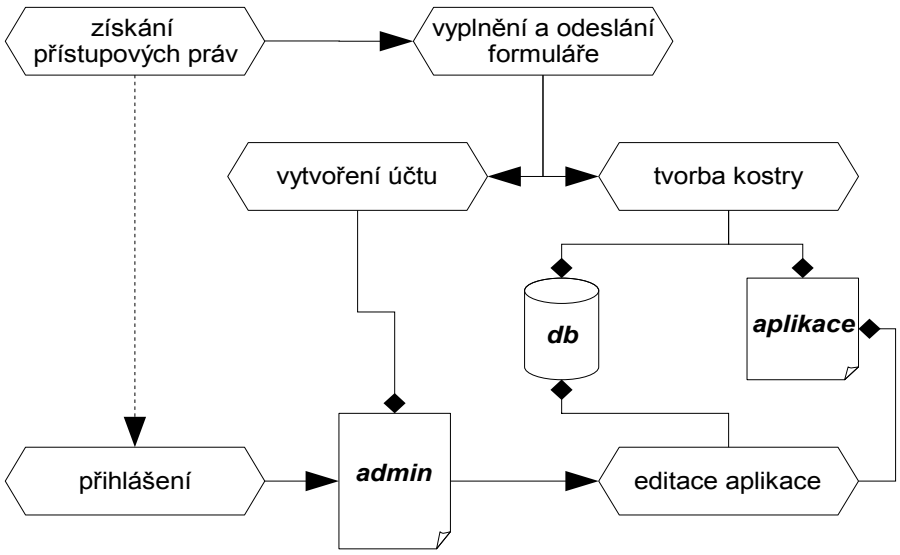
Na všech webových stránkách je čas od času nutné něco změnit. Prostředí pro tvorbu interaktivních webových stránek je webová aplikace umožňující tvorbu a následnou administraci webových stránek. Dalo by se říci, že toto prostředí v sobě jímá rozšířený publikační systém, databázový administrační systém s vysokoúrovňovým přístupem, návrhář webových stránek s několik dalších systémů pro správu webového místa.

Toto prostředí umožňuje vytvářet a dynamicky měnit všechny prvky stránky, měnit strukturu a vzhled okna, pracovat s databázovými tabulkami, automaticky generovat formuláře a další výstupy z databáze, vytvářet a modifikovat články, komplexně spravovat obrázky aplikace nebo dokonce vytvářet vlastní strukturu stránek pomocí jednoduchého metajazyka, a to vše bez znalosti programovacích jazyků a technik (kromě HTML). Celé toto prostředí a všechny jeho možnosti budou podrobněji popsány v následujících kapitolách.

3.1 Tvorba aplikace

Chceme-li vytvořit novou aplikaci pomocí toho prostředí je nutné znát všechna potřebná přístupová hesla, které nám poskytne webmáster (kapitola: „Úrovně přihlašování“). Poté je možné vygenerovat kostru aplikace. Tu je možné dále upravovat a administrační části prostředí. Pro generování kostry aplikace je vytvořeno speciální prostředí, ve kterém se musí vyplnit formulář s několika údaji:

- jméno aplikacemi
- přihlašovací jméno hlavního administrátora
- heslo hlavního administrátora



Obrázek 3.1: Postup při tvorbě nové aplikace

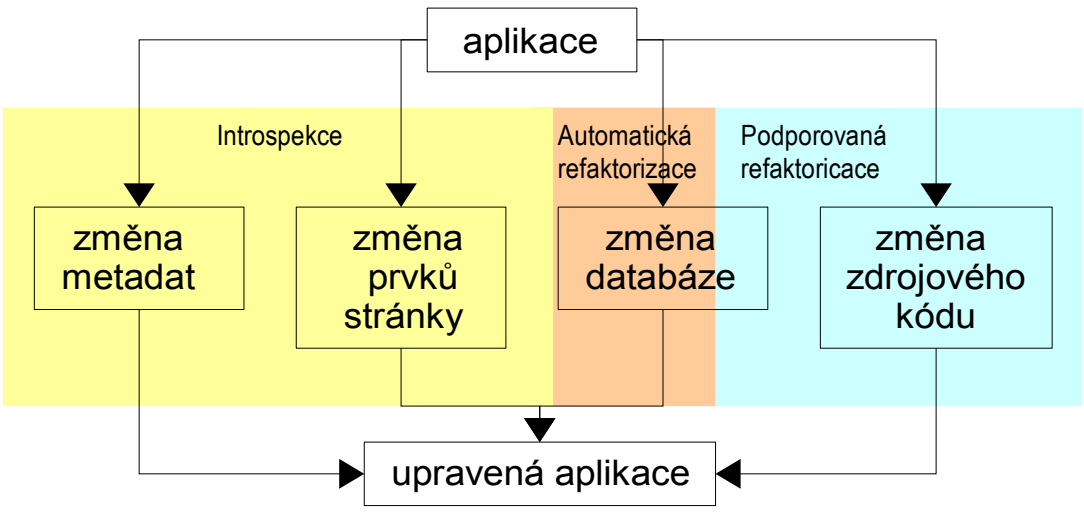
Po vyplnění těchto údajů (údaje lze explicitně měnit) a odeslání formuláře se vytvoří kostra aplikace. Také je vytvořen účet v administračním prostředí pro tuto aplikaci. Zde je pro přihlášení aplikace dále vyvíjena a upravována. Postup vytvoření nové aplikace je vysvětlen na předchozím obrázku. V administrační části prostředí se nachází hlavní menu, ze kterého zle přistupovat do všech částí systému. Položky hlavního menu lze rozdělit do několika základních skupin:

- práce s daty a databázovými entitami (editace datových struktur, změny struktury tabulek, tvorba indexů, změna viditelnosti, změna metadat a změna dat)
- přihlašování, správa účtů, řízení přístupu (editace uživatelů, editace uživatelských rolí)
- záznam událostí (záznam všech relevantních událostí v prostředí)
- textové editace (správa některých prvků stránek jako jsou články, ankety, td.)
- fotogalerie (správa obrázků, upload obrázku, tvorba fotogalerii)
- tvorba stránek (vytváření menu, tvorba obsahu stránek, editace základních informací, práce s webovými šablonami, atd.)
- pomocné nástroje (kalkulačka, zámek, nápověda)

Vše bude podrobněji vysvětleno v dalších kapitolách. Náhledy administrační aplikace jsou v přílohách G, H, I, J a K.

3.2 Podporované změny aplikace

Veškeré změny aplikace je možné z pohledu webové aplikace rozdělit do následujících čtyrech skupin: změna dat aplikace, introspekce a plně automatizovaná refaktorizace, aplikací podporovaná refaktorizace. Změna aplikačních dat je u dynamických webových stránek celkem běžná záležitost, takže se sní zde nebudu vůbec zaobírat. Jen pro úplnost sem patří hlasování v anketě, změna uživatelského jména nebo hesla, přidání a úprava údajů databázové tabulky, a další.



Obrázek 3.2: Změny aplikace, introspekce, refaktorizace

3.2.1 Introspekce

Za introspekcí je považována taková událost, která vede ke změna aplikace, ale není potřeba měnit zdrojový kód aplikace nebo strukturu databáze. V tomto prostředí postačí změna metadat nebo změna prvku stránky. Metadaty jsou myšlena taková data, která popisují vlastnosti nebo chování objetu nebo procedur v aplikaci. Prvky stránky jsou popsány v následujících kapitolách: „Řídící prvky stránky“, „Elementární prvky stránky“, „Databázové prvky stránky“. Je nutné si uvědomit, že jde o introspekci vztaženou k cílové aplikaci, nikoli k administrátorské aplikaci, pro tu jde pouze o změnu dat. Typickými webovými aplikacemi založených na introspekci jsou publikační a redakční systémy. Introspektivní operce v tomto prostředí jsou:

- přidání, editace a odstranění modu viditelnosti dat databázové tabulky
- přidání, editace a odstranění viditelnosti databázové tabulky
- přidání, editace a odstranění článku
- přidání, editace a odstranění obrázku
- přidání, editace a odstranění ankety
- přidání, editace a odstranění souboru
- přidání, editace a odstranění zprávy administrátora
- přidání, editace a odstranění položky menu a stránky aplikace
- přidání, editace a odstranění uživatele
- přidání, aktivace a odstranění souboru s kaskádovými styly
- změna nebo editace webové šablony
- editace jazykových řetězců (nelze přidávat a mazat)
- editace bázových informaci (jméno aplikace, ikony, autor, klíčová slova,..)
- některé další

3.2.2 Automatizovaná refaktorizace

Automatizovaná refaktorizace je taková změna aplikace, která vyžaduje změnu zdrojového kódu nebo změnu struktury databáze, ale tato změna se nemusí provádět ručně (provádí se pomocí nějakého uživatelského prostředí). Takovými změnami jsou:

- přidání, editace a odstranění uživatelské databázové tabulky aplikace se kterou se pracuje pouze pomocí prvků stránek
- přidání, editace a odstranění sloupce, indexů uživatelské databázové tabulky aplikace se kterou se pracuje pouze pomocí prvků stránek
- přidání, editace a odstranění položky menu (stránky aplikace), nejedná-li se o stránku s externím programátorským kódem

Podrobnější popis těchto událostí bude v následujících kapitolách.

3.2.3 Aplikací podporovaná refaktorizace

Aplikací podporovaná refaktorizace, je taková změna aplikace, při které dochází ke změně zdrojového kódu nebo ke změně struktury databáze, ale tuto změnu není možné plně automatizovat. Typicky se taková změna neobejde bez částečné nebo úplné ruční změny kódu. Změnu struktury databáze lze ve většině případů automatizovat. Změnami tohoto typu jsou:

- přidání, editace a odstranění databázové tabulky aplikace, se kterou se pracuje v nějakém programátorském kódu
- přidání, editace a odstranění databázové tabulky aplikace, která je v relaci s nějakou jinou databázovou tabulkou
- přidání, editace a odstranění sloupce, indexů databázové tabulky aplikace, se kterou se pracuje v nějakém programátorském kódu
- přidání, editace a odstranění relace databázových tabulek aplikace
- přidání, editace a odstranění položky menu a stránky aplikace, jedná-li se o stránku s externím programátorským kódem
- přidání jazykové mutace
- některé další změny aplikace, kde je využito lidové tvořivosti

Podrobnější popis výše uvedených událostí bude v následujících kapitolách. Kde bude uvedeno jak se s nimi pracuje a proč patří zrovna do této kategorie změn.

Existují samozřejmě také aplikací nepodporované refaktorizace. Jsou to všechno změny, které by vyžadovaly změnu zdrojového kódu v jádře prostředí pro tvorbu aplikace. Takovými změnami mohou být například:

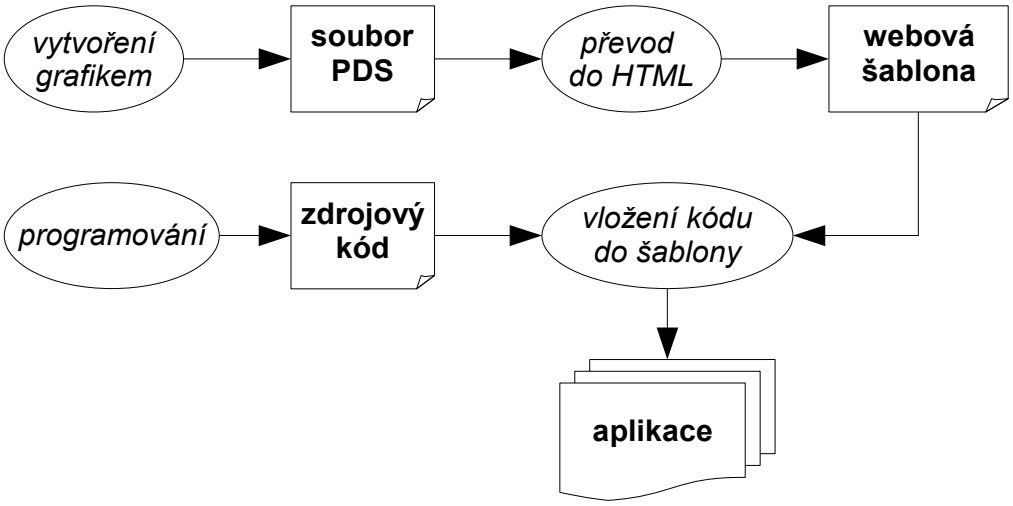
- tvorba nového typu prvku stránky nebo modulu
- změna systému přihlašování a bezpečnostní politiky
- přidání předdefinované stránky (kapitola: „Předdefinované stránky“)
- jakákoli jiná změna návrhu prostředí

3.3 Obsah a vzhled stránek

Vzhled webových stránek je jednou z nejdůležitějších věcí, které na první pohled určuje kvalitu stránek, leč někdy mylně. Dnešní trend je takový, že grafiku a vzhled stránek necháme udělat grafika, ale aplikační část a databáze je vytvořena programátorem. Je velmi dobré, aby práce těchto lidí mohla být separátní. Pro tento účel existují tak zvané webové šablony (*templates*).

3.3.1 Webové šablony

Webová šablona není nic jiného než HTML nebo XHTML soubor s množstvím obrázků použitých v grafice a soubory s předlohou kaskádových stylů. Někdy zde bývají i soubory se skripty v Java Scriptu, Visual Basicu nebo Active X. U graficky velmi zdařilých šablon nejsou výjimkou ani Flash soubory. Webová šablona může být také ve formátu PDS (formát pro Adobe Photoshop)



Obrázek 3.3: Začlenění webové šablony do aplikace

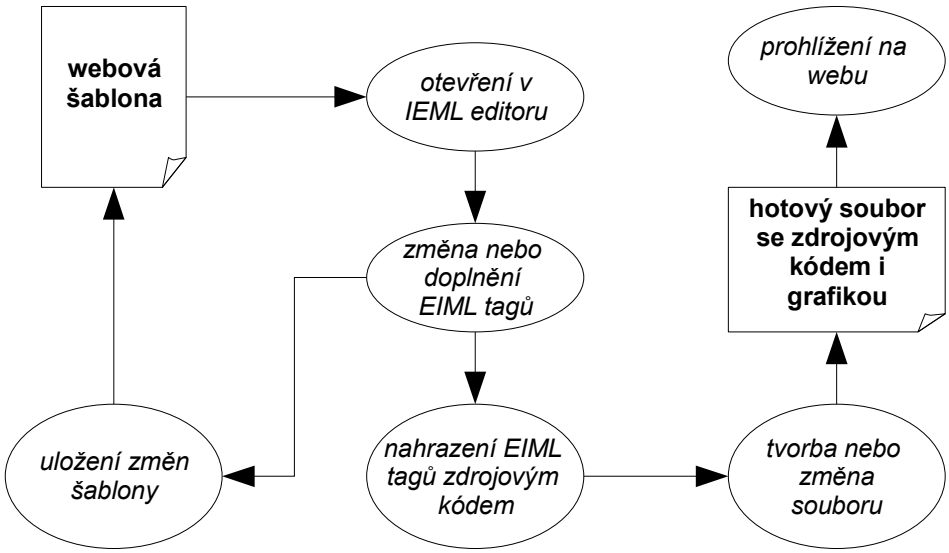
Takové to webové šablony jsou používaný ve většině moderních redakčních a publikačních systémů. Výhodou je, že grafika se dá snadno změnit, protože mezi ni a zdrojovým kódem aplikace není tak úzká vazba, nebo je možné mít pro každou stránku jinou webovou předlohu. Další velkou výhodou je usnadnění práce. Vývoj webové aplikace podporující šablony je zhruba následující.
Jak plyne z předchozího obrázku, šablony s musí nějakým způsobem propojit se zbytkem zdrojového kódu aplikace. To je možné provést několika způsoby.

Pokud je vše programováno objektovým způsobem, stačí v šabloně zavolat instanci třídy a její metody. V procedurálním přístupu se volají funkce. To ovšem může zdrojový kód šablony velmi znepřehlednit a někteří programátoři zde vytváří i konstrukce, které jsou nežádoucí a vedou ke spustě chybám. Výhody šablony se zde prakticky ztrácí. Dalo by se říci že se již nejedná o šablony.

Další, asi nejvíce používaná možnost, je použití meta řetězců. Tyto řetězce se doplní do šablony, poté se celá šablona parseruje a meta řetězce jsou nahrazeny hodnotou výstupu funkce nebo metody objektu navázané na tento řetězec. To to je dobrý způsob avšak má jeden velký nedostatek, který si většina programátorů neuvědomuje. Totiž k tomuto parserování dochází při každém načtení stránky (požadavku klienta o stránku). Většinou se do pamětí serveru načítá celý soubor šablony a dále se sním pracuje. Výstupní data funkcí se musí taktéž ukládat do řetězců, které mohou být i řádově delší než původní soubor šablony. Ať se zdá tato metoda jakkoli absurdní, používá podobného principu, dle mích zkušeností, 60% redakčních systémů.

3.3.2 IEML editor šablon

Přemýšlel jsem, jak by bylo možné zdokonalit předchozí metodu tak, aby zůstali zachovány její výhody a současně byly odstraněny všechny její nevýhody. Je tedy jasné, že v okamžiku přístupu na stránku (načtení stránky) musí už být v šabloně zdrojový kód nikoli meta řetězce (jak je tomu u první popsané metody). K parserování a nahrazení musí tedy dojít v okamžiku připojení šablony k aplikaci (tedy ve fázi vývoje systému). K těmto účelů jsem vytvořil editor webových šablony, který je součástí administračního rozhraní systému. Editor využívá metajazyka EIML, který je využit také v editoru obsahu stránek (IEML je podrobněji popsán v dalších kapitolách). Tato metoda připojení webové šablony je lépe vysvětlena následujícím obrázkem.



Obrázek 3.4: Připojení webové šablony

Screen shot IEML editoru je v příloze G) Screen shot IEML editoru

3.3.3 Správce webových šablon

V administrační části nesmí chybět ani správce webových šablon. Počet šablon pro aplikaci není teoreticky omezen. Zde jsou zobrazeny všechny šablony obsažené v aplikaci s kterými lze provádět následující akce:

- otevření šablony v IEML editoru
- smazaní šablony
- přidání nové šablony
- aktivace a deaktivace šablony
- volba šablony pro stránku aplikace

3.4 IEML a prvky stránky

3.4.1 Metajazyk IEML

Obsah stránek (i šablon) je možné vytvářet pomocí metajazyka IEML v editoru obsahu stránek (i editoru šablon). Editor obsahu stránek je totožný s editorem šablon. Jediným rozdílem je editovaný objekt. Zdrojový kód je v editoru zadáván ručně, nebo pomocí formulářů. IEML (interactive expansion markcap language) je zvláštní sada tagů (elementů) s novými vlastnostmi, které jsem vytvořil speciálně pro potřebu popsání prvků stránek. Syntaxe je obdobná jako u ostatních značkovacích jazyků (HTML, XML), liší se pouze počátečním znakem *$* za první špičatou závorkou. Tyto tagy se mohou na stránce libovolně kombinovat s HTML nebo XML tagy, pokud budou dodržena obecná pravidla pro tvorbu HTML. Elementy jsou jak párové, tak nepárové. Přesná syntaxe IEML elementů je následující:

```
<$ jméno_elementu [jméno_atritutu="honota_atributu"]*/>
```

Pro párové elementy:

```
<$ jméno_elementu [jméno_atritutu="honota_atributu"]*/> </ jméno_elementu>
```

Symbol *$* značí že se jedná o IEML element, mezi symboly < a *$* nesmí být mezera. Pokud nejsou předchozí podmínky splněny překladač si myslí že jde o HTML element a nedojde k nahrazení funkčním kódem aplikace.

Každý EIML tag tedy reprezentuje nějaký prvek stránky. Seznam atributů daného elementu reprezentuje seznam vlastností jemu ekvivalentnímu prvku stránky. Prvků stránky je poměrně mnoho, proto jsou rozděleny do několika skupin. Toto rozdělení nemá ovšem žádný vliv na syntaxi EIML jazyka.

3.4.2 Rozdělení prvků stránky

Prvky stránky je možné rozdělit do čtyř základních skupin podle jejich vlastností a způsobu použití:

- interaktivní prvky stránky
- obsahové prvky stránky
- databázové prvky stránky
- informativní prvky stránky
    - hlavičkové prvky stránky

Hlavičkové prvky stránky jsou speciálním případem informativních prvku. Z několika důvodů bylo nutné je uvést samostatně. Popis skupin a jejich prvků je podrobněji rozepsán v následujících kapitolách. U každého prvku budou uvedeny vlastnosti, způsob využití a konstrukce elementu v metajazyku IEML.

3.5 Interaktivní prvky stránky

Interaktivní prvky stránky jsou takové prvky, které provádí nějakou akci. Tzn. ukládají data, mění nastavení, dovolují procházení aplikací nebo umožňují nějakou speciální funkčnost. Prvky s těmito vlastnostmi mohou být použity jak v editoru šablon, tak v editoru obsahu stránek.

Definovány jsou tyto interaktivní prvky: hlavička odkazů, roletové menu, primární menu, sekundární menu, přihlašovací formulář, počítadlo návštěv, panel jazyků, anketa, odkaz, modul vyhledávání, fórum, mapa stránek a programátorský modul.

3.5.1 Menu

Menu slouží pro procházení stránek a lepší (přehlednější) orientaci ve struktuře stránek. V aplikaci mohou být použity tři typy menu: hlavička odkazů, roletové menu a dělené menu (primární a sekundární menu). Na stránce je možné použít všechny typy současně aniž by docházelo ke kolizím. Není problém použít kterékoli menu duplicitně (triplicitně,...).

3.5.1.1 Hlavička odkazů

Hlavička odkazů je určena především pro zobrazení klíčových stránek, nebo externích stránek (typicky rodičovského nebo sourozeneckého charakteru). Menu je pouze jednoúrovňové. Pokud je součástí hlavičky odkazů položka typu složka (kapitola: „Menu a systém stránek“), bude po kliknutí na tuto položku zobrazena stránka obsahující všechny dceřiné položka. Orientace hlavičky odkazů je pouze horizontální. V IEML je hlavička odkazů reprezentována elementem *hlinks*, který nemá žádné atributy.

Příklad:

```
<$hlinks />
```

3.5.1.2 Roletové menu

Roletové menu je klasické menu, kde se po najetí na položku, obsahující dceřiné položky, zobrazí nabídka těchto položek. Toto menu je tedy víceúrovňové. Počet podporovaných úrovní menu v tomto prostředí je pět. Rozšíření na více úrovní je aplikací podporovaná refaktorizace (je nutné provést úpravu kaskádových stylů). Roletové menu má element *menu* bez atributů.

Příklad:

```
<$menu />
```

3.5.1.3 Dělené menu

Dělené menu je menu složené ze dvou částí (primární a sekundární). Při aktivaci položky primárního menu, která má dceřiné prvky, je modifikováno sekundární menu (bude obsahovat právě tyto položky). Výhoda je v tom, že tyto dvě části se mohou nacházet na různých místech stránky,

nezávisle na sobě. Z toho vyplývá, že je maximální počet úrovní je dvě. Dělené menu má dva elementy, *primenu* pro primární část a *secmenu* pro sekundární část.

Příklad:

```
<$primenu /> <br /> <$secmenu />
```

3.5.2 Přihlašovací formulář

Přihlašovací formulář je místo pro přihlášení registrovaného uživatele. Může být zobrazeno vertikálně nebo horizontálně. Vertikální menu bývá obvykle skryto a před přihlášením se musí rozrolovat. Proběhne-li přihlášení úspěšně, je u obou variant místo přihlašovacího formuláře zobrazena přezdívka uživatele, některé další informace a odkaz pro možnost odhlášení. Informace o mechanizmech přihlášení jsou v kapitole: „Přihlašování, správa účtů, řízení přístupu“.

V jazyce IEML je přihlašovací formulář reprezentován elementem *logfrm*. Dále je možné pomocí atributu *orint* určit zobrazeni horizontální nebo vertikální. Nechybí ani atribut *class*.

Příklad:

```
<$logfrm head="vert" class="loginform" />
```

3.5.3 Počítadlo návštěv

Toto počitadlo návštěv není vnitřně nijak implementováno, ale jedná se o počítadlo společnosti TOPlist (více informací na http://www.toplist.cz). V administrační části se uloží kód, který při zobrazení na webové stránce funguje jako počitadlo přístupů a zároveň jako tabulka obsahující statistiku.

Obrázek 3.5: TOPlist počítadlo přístupů

Příklad:

```
<$counter />
```

3.5.4 Panel jazyků

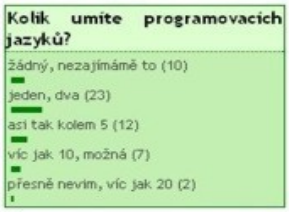
Panel jazyků je nabídka aplikací podporovaných jazyků. Kliknutí na ikonu vypraného jazyka se všechny řetězce (budou-li k dispozici) přeloží do tohoto jazyka. Bližší popsáno v kapitole Podpora více jazyků. Příslušný element v IEML je langpa.

Příklad:

```
<$langpa />
```

3.5.5 Anketa

Prvek anketa reprezentuje klasikou anketu pro zjištění názorů návštěvníků stránek. V editoru anket je možné vytvořit novou anketu nebo editovat a smazat anketu stávající. Anketa musí obsahovat anketní otázku a teoreticky neomezený počet nabídnutých možností. Kdykoli lze možnost přidat, odebrat nebo editovat hodnoty těchto možností. Anketa má také parametr který říká, zda má být aktivní. To je výhodné zejména tehdy, když chceme dočasně anketu zrušit. Je-li anketa na více místech aplikace, skryjí se všechny její výskyty. V anketě se dá hlasovat pouze jednou z jedné IP adresy.



Obrázek 3.6: Ukázka ankety

Pro tvorbu ankety se využívá elementu opin. Seznam atributů pro tento element je v následující tabulce.

Atribut	*Význam*	*Možné hodnoty*	*Povinný*	*Default*
src	Zdrojová anketa v databázi.	string	ano	není
class	Třída kaskádových předloh.	string	ne	není

Tabulka 3.1: Seznam atributů a možných hodnot elementu opin

Příklad:

```
<$opin src="bes-film" class="anket" />
```

3.5.6 Odkaz

Jedná se obyčejný hypertextový odkaz. Pomocí formulářů v IEML editoru lze snadno vyprat odkazovány objekt. Pokud se použije element *a* jako element IEML a ne HTML, bude před generováním výsledné stránky zkontrolováno, zda koncový objekt existuje a je přístupný za současného nastavení a přístupových práv (kapitola: Přihlašování, správa účtů, řízení přístupu). IEML element *a* má stejné atributy jako obyčejný odkaz v HTML.

Příklad:

```
<a href="extern_pages/index.htm" name="link1" target="_parent"
class="mlink" >Úvodní strana</a>
```

3.5.7 Mapa stránek

Mapa stránek zobrazí seznam všech stránek aplikace. Struktura stránek je stejná jako v menu (rodič – potomek). Déle je zde obsažen seznam stránek nenacházejících se v žádném menu. Jsou zobrazeny pouze aktivní stránky, které mají ACI nižší nebo stejné jako aktivní uživatel (kapitola: „Přihlašování, správa účtů, řízení přístupu“). Na mapě stránek nejsou zobrazeny duplicitní stránky, pokud se odkaz na tyto stánky nachází v obsahu jiné stránky. Mapu stránek je možné vložit elementem *pmape*. Tento element nemá žádné atributy.

Příklad:

```
<$pmape />
```

3.5.8 Stránka vyhledávání

Další velmi důležitou funkcí v aplikaci je vyhledávání. Na stránce vyhledávání je možné nastavit několik upřesňujících parametrů. Prvním parametrem je oblast vyhledávání. Je možné vyhledávat v následujících oblastech:

- vyhledávání v databázových tabulkách (mimo systémových tabulek)
- vyhledávání v obrázcích
- vyhledávání ve článcích
- vyhledávání v stránkách (obsahu stránek)
- vyhledávání v souborech

Vždy je nutné označit minimálně jednu oblast vyhledávání. Druhým parametrem je volba case-senzitivity, neboli rozlišování malých v velkých písmen. Je možné také nastavit note-senzitivitu, tj. rozlišování interpunkčních znamének. Posledním volitelných parametrem je míra shody. Hodnota je v rozmezí 50-100%. 50-ti procentní míra shody znamená, že slovo (fráze) bude považováno za vyhovující, pokud je alespoň polovina symbolů shodná na polovině pozicích. Při 100 procentní shodě se hledá přesně to které slovo (fráze). Výsledkem hledání je seznam odkazů na objekty, ve kterých bylo hledané slovo podle výše uvedených kritérii nalezeno. Zobrazeny jsou pouze ty objekty, které mají ACI nižší nebo stejné jako uživatel (kapitola: „Přihlašování, správa účtů, řízení přístupu“).

Modul pro vyhledávání má přidělen element *srch*, který má několik atributů.

Atribut	***Význam***	***Možné hodnoty***	***Povinný***	***Default***
arel	Zobrazit výběr vyhledávacích oblastí.	true, false	ne	true
cast	Zobrazit výběr možností pro case-senzivitu	true, false	ne	true
math	Zobrazit výběr míry shody.	true, false	ne	true

Tabulka 3.2: Seznam atributů a možných hodnot elementu srch

Příklad:

```
<$srch arel="ne" cast="ano" math="ne"/>
```

3.5.9 Fórum

Jako modul je zde také fórum. Jedná se o klasické fórum s hlavním seznamem diskusí. Po vstupu do diskuse je zobrazen seznam všech příspěvků v aktuální diskusi. Dále je možné vkládat vlastní příspěvky s využitím citací předchozích uživatelů a možností vkládání emotikon (smajlíků). Seznam atributu k elementu *forum* je v následující tabulce.

Atribut	***Význam***	***Možné hodnoty***	***Povinný***	***Default***
order	Pořadí příspěvků.	asc, desc	ne	acs
topic	Zobrazit pouze vybraný topic.	topic id	ne	all

Tabulka 3.3: Seznam atributů a možných hodnot elementu forum

Příklad:

```
<$forum order="aes" topic="7" />
```

3.5.10 Programátorský modul

Programátorský modul reprezentuje programátorem vytvořenou stránku. V případě, že toto prostředí pro tvorbu interaktivních webových aplikací bude využíváno nějakým dalším specifických způsobem (internetový obchod, fotogalerie,...), musí být ručně implementovány některé části aplikace. To se provede tak, že vytvoříme nový PHP soubor. V tomto souboru musí být obsažena třída *source_class* jako dceřiná třída třídy *jm_module*. Dále tato třída musí obsahovat metodu run(p1,p2), která má dva parametry, jejíchž hodnota je předána z atributu elementu. Třída musí mít samozřejmě i konstruktor (šablona této třídy je v příloze B). V souboru mohou být i další zcela libovolné třídy, které ovšem musí být v relaci s třídou *source_class* . Lze připojovat i ostatní PHP soubory. Pokud bude dodrženo přesné předávání parametrů konstruktorů těchto tříd, budou i tyto stránky podporovat multijazyčnost, dynamickou změnu obrázků a přístup k databázi.

Atribut	***Význam***	***Možné hodnoty***	***Povinný***	***Default***
page	Cesta k PHP souboru	string(path)	ano	není
par1	První parametr předávaný třídě.	př: 'm:d:y' nebo 'name-day'	ne	není
par2	Druhý parametr předávaný třídě.	string	ne	není
class	Třída kaskádových předloh.	string	ne	není

Tabulka 3.4: Seznam atributů a možných hodnot elementu source

Příklad:

```
<$source page="fotoview.php" class="album" />
<$source page="item_list.php" par1="mobil" par2="color_display" />
```

3.6 Obsahové prvky stránky

Obsahové prvky stránky jsou takové prvky, u kterých je výsledkem jejich obsah. Jsou převážně statické a nemají žádnou funkcionalitu. Vkládat se mohou jak do obsahu stránek tak do šablony a to libovolně krát na jakékoliv místo. Úprava prvků, tvorba (upload na server) a mazání prvků se provádí v administrační části prostředí. Těmito prvky jsou: obrázek, galerie, článek, kniha, administrátorské zprávy, zdrojový kód a interní soubor.

3.6.1 Obrázek

Prvek obrázek reprezentuje obrázek uložený v souborovém systému na serveru (i na jiném než našem serveru). Podporované formáty obrázků jsou: gif, tiff, jpeg, png a bmp. Prvek jako takový je uložen v databázi s těmito parametry: titulek, cesta, ACI. Titulek je textový řetězec který slouží popsání obsahu obrázku. Parametr cesta je cesta k obrázku v systému souborů. ACI je index kontroly přístupu (kapitola: „Přihlašování, správa účtů a řízení přístupu“). Obrázek lze do databáze obrázku přidat uploadem nového obrázku na server. Relace mezi obrázky v databázi a fyzickými obrázky na serveru je 1:1. Bude-li obrázek z databáze odstraněn, bude smazán i obrázek v souborovém systému. IEML element reprezentující obrázek je *img*, který má atributy stejné jako HTML element *img*. Pokud se použije obyčejný HTML element nebude kontrolována přítomnost a přístup ke zdrojovému obrázku.

Příklad:

```
<$img src="images/icon_project.png" alt="projekt" title="Projekt"/>
```

3.6.2 Galerie obrázků

Všechny obrázky jsou v administračním systému zobrazovány jako v souborovém systému. Prvek galerie obrázků má tu schopnost, že ho lze navázat na libovolnou složku tohoto virtuálního souborového systému, a vytvoří ze všech jeho členů fotogalerii. Editací obsahu složky se automaticky mění i obsah galerie. Galerii patří element *galery*, který má následující atributy.

Atribut	***Význam***	***Možné hodnoty***	***Povinný***	***Default***
scr	Určení zdrojové složky	string(path)	ano	není
count	Počet obrázků na jedné stránce	číslo	ne	32
prew	Zobrazení náhledu pro kliknutí na obrázek	true, false	ne	true
title	Zobrazovat titulky obrázků	true, false	ne	false
class	Třída kaskádových předloh.	string	ne	není

Tabulka 3.5: Seznam atributů a možných hodnot elementu galery

Příklad:

```
<$galery src="fotky" count="34" modify="false" prew="false"  />
```

3.6.3 Článek

Základními stavebními kameny webové stránky jsou články. Jak už bylo řečeno, články se dají vkládat jak do obsahu stránek, tak šablony stránky. Relace mezi článkem a stránkou (stránkou jako měnícím se obsahem stránky, kapitola: „Menu a systém stránek“) je m:n. To znamená, že na jedné stránce může být více článků zároveň a jeden článek může být současně na více stránkách. U klasických publikačních systémů to není obvyklé.

Parametry tohoto prvku jsou: titulek článku, vlastní text, datum vytvoření, zobrazovat titulek, zobrazovat datum, zobrazovat tisk, kategorie a index kontroly přístupu. Vlastní text článku může být upravován v editoru článků (FCKeditor). Zde je text formátován podobným způsobem jako v MS Office Word. Ostatní parametry je možné měnit v tabulce článků. Parametry zobrazovat titulek, zobrazovat datum, zobrazovat tisk je možné měnit i dále v editoru obsahu stránky (kapitola: „Obsah stránek, IEML“). Hodnoty atributů jsou vázány pouze k danému výskytu prvku.

Článek je prezentován elementem *artc*. Seznam všech atributů je v následující tabulce.

Atribut	***Význam***	***Možné hodnoty***	***Povinný***	***Default***
src	Zdrojový článek v databázi.	string	ano	není
date	Určuje zda bude zobrazeno datum vytvoření.	true|false|default	ne	default
title	Titulek v hlavičce formuláře.	true|false|default| string	ne	default
class	Třída kaskádových předloh.	string	ne	není

Tabulka 3.6: Seznam atributů a možných hodnot elementu artc

Pokud má atribut date hodnotu true, bude u článku zobrazeno datum vytvoření článku, které je uloženo v databázi. Hodnota false značí, že u článku nebude uvedeno žádné datum. V případě hodnoty string se použije vlastní datum.

Příklad:

```
<$artc src="about me" titulek="true" date="13.5.2007" />
<$artc class="articles" src="cpplang" date="true" titulek="Úvod do C++" />
```

3.6.4 Kniha

Kniha není nic jiného než soubor článků. U každého článku je určeno do jaké patří kategorie. Navážeme-li knihu k některé kategorii, budou se v této knize zobrazovat všechny články této kategorie. Veškerá editace knihy (přidání, editace, odebrání článku) se provádí automaticky. Je určitá analogie k prvku galerie. Kniha je v metajazyce IEML prezentována elementem *book*, který má následující atributy.

Atribut	*Význam*	*Možné hodnoty*	*Povinný*	*Default*
them	Zdrojová kategorie článků	string	ano	není
order	Určuje pořadí článků.	time|title	ne	time
prew	Zobrazovat náhled v obsahu knihy	true|false	ne	true
date	Zobrazovat datum článku.	true|false	ne	true
title	Zobrazovat nadpis článku.	true|false	ne	true
class	Třída kaskádových předloh.	string	ne	není

Tabulka 3.7: Seznam atributů a možných hodnot elementu book

Příklad:

```
<$book them="počítače" order="time" prew="false" date="true" title="false"
class="mybook" />
```

3.6.5 Administrátorské zprávy

Zprávy od administrátora mají ryze informativní charakter. Jde o krátké zprávy orámované barevným rámečkem. Barva rámečku je dána důležitostí zprávy. Od zelené (nízká), přes žlutou (střední), až po červenou (vysoká). Dále je možné nastavit aktivitu a dobu platnosti zprávy. V IEML je seznam zpráv administrátora uveden pod elementem *admes*.

Příklad:

```
<$admes />
```

Aktivní	*Popis*
nikdy	zpráva se nezobrazuje, to je dobré pokud je zpráva používána nepravidelně
vždy	zpráva je aktivní pořád
v čase	zpráva je aktivní dokud nevyprší doba její platnosti, pote zmizí

Tabulka 3.8: Aktivita a datum platnosti zprávy administrátora

3.6.6 Interní soubor

Prvek interní soubor reprezentuje soubor uložený na serveru (i jiném než našem serveru). Použití tohoto prvku se od předchozích trochu liší. Může být součástí obsahu stránky, ale už zde nemůže být nic jiného (kapitola: „Menu a systém stránek“). Nelze ho vložit ani do šablony. Práce se souborem je velmi podobná práci s obrázkem. Mají i stejné parametry titulek cesta a ACI. Při smazání souboru v seznamu souboru dojde opět k jeho fyzickému odstranění. Podporované typy souborů jsou: prostý text (txt), rich text(rtf), offline HTML stránka (html) , pdf, prezentace MS Power Point, zip, rar, x-tar, x-zip,x-gzip a x-zip-compressed. Tento prvek není interpretován v IEML.

3.7 Databázové prvky stránky

Databázové prvky stránky, jak už název napovídá, potřebují ke své existenci databázi. Je nutné mít v databázi již vytvořené entity (tabulky, sloupce, ...) (kapitola: „Administrace databázové části“) a teprve poté nad nimi můžeme vytvářet databázové prvky stránek, jako je tabulka, formulář nebo SQL dotaz. všechny tyto prvky provádí pouze příkazy typu DML SQL.

3.7.1 Prvek tabulka

Prvek tabulka je HTML tabulka vytvořená dle vzorové tabulky v databázi. Výstupní data jsou nejen zobrazena, ale je možné je i editovat. Viditelnost jednotlivých sloupců tabulky dána nastavením viditelnosti modu a relací mezi tabulkou a tímto modem (kapitola: „Viditelnost prvků tabulky“). Parametry výstupní tabulky jsou dány hodnotou atributu elementu *dbtable*. Přehled atributů je v tabulce.

Atribut	***Význam***	***Možné hodnoty***	***Povinný***	***Default***
src	Zdrojová tabulka v databázi.	string	ano	není
mode	Zobrazovací mód. Určuje viditelnost.	string	ne	root
title	Titulek v hlavičce tabulky.	true|false|string	ne	false
sort	Určuje bude-li možno data v tabulce řadit.	true|false	ne	false
change	Určuje bude-li možno data v tabulce měnit.	true|false	ne	false
class	Třída kaskádových předloh.	string	ne	není

Tabulka 3.9: Seznam atributů a možných hodnot elementu dbtable

Pokud má atribut *title* hodnotu *true*, bude mít tabulka stejný titulek jako má tabulka v databázi. Hodnota *false* znamená, že tabulka nebude obsahovat žádný titulek. V případě hodnoty *string* se použije titulek vlastní. Atribut *class* mají všechny elementy. Má obvyklý význam a stejné použití jako v jazyce HTML.

Příklad:

```
<$dbtable src="users" mode="output" titulek="true" sort="false"
change="true"/>
<$dbtable class="blackstyle" src="events" titulek="Audit" />
```

3.7.2 Prvek formulář

Tento prvek je webový formulář, automaticky vytvořený podle vlastností tabulky v databázi. Formulář obsahuje formulářová políčka, název položky a informace o položce ve všech používaných jazycích, informace o datovém a logickém typu položky (Kapitola: „Datové a logické typy“). Podle těchto dvou typu se automaticky určí typ vstupního formulářového pole s odpovídajícími omezeními (např. délka řetězce). Pro datové typ *enum* je políčko typu *select*, typ *set* je prezentován *multiple*

select, textové datové typy jsou nahrazeny formulářovým prvkem *input* s odpovídajícím omezením délky textu. Pokud je použit datový typ text nebo je maximální délka textu větší než 64 znaků je použito *textarea*. U datových typů obsahujících čas je přidáván kalendář pro usnadnění vkládání časových údajů. U číselných hodnot je použit prvek *input*. Při ukládání do databáze je důkladně kontrolován datový i logický typ, maximální velikost, neprázdnost a další omezení. IEML element pro tento prvek se jmenuje *dbform* a má následující atributy.

Atribut	***Význam***	***Možné hodnoty***	***Povinný***	***Default***
src	Zdrojová tabulka v databázi.	string	ano	není
mode	Zobrazovací mód. Určuje jaké položky mají být viditelné.	string	ne	root
title	Titulek v hlavičce formuláře.	true|false|string	ne	false
class	Třída kaskádových předloh.	string	ne	není

Tabulka 3.10: Seznam atributů a možných hodnot elementu dbform

Příklad:

```
<$dbform src="users" mode="input" titulek="true" />
<$dbform class="blackstyle" src="events" titulek="Audit" />
```

3.7.3 Prvek SQL dotaz

Jelikož oba předcházející prvky mají velice konkretní tvar a někdy je nutné použít velice specifický výstup z databáze, je zde definován obecný prvek SQL dotaz. Protože se jedná o prvek zobrazující nějaký výstup dat z databáze na HTML stránku, je možné použít jen dotazy typu *SELECT*. Jinak může být SQL dotaz zcela libovolný s ohledem na formát výstupu dat. Tento příkaz je optimalizován na krátké jednořádkové výsledky (např. počet řádků, aktuální událost). Pro zobrazení např. obsahu databázové tabulky je tedy vhodnější použít element *dbtable*, protože zde není možné pracovat s mody a viditelností, ani nijak formátovat výsledek dotazu.V IEML je presentován elementem *sqlr*.

Atribut	***Význam***	***Možné hodnoty***	***Povinný***	***Default***
src	Zdrojový SQL dotaz.	string	ano	není
type	Určuje v jakem HTML tagu bude výsledek zobrazen.	div|p|spam|table	ne	spam
class	Třída kaskádových předloh.	string	ne	není

Tabulka 3.11: Seznam atributů a možných hodnot elementu sqlr

Příklad:

```
<$sqlr src="select id from user where id=10;" type="div" />
<$sqlr class="events" src="select * from events where data = now()" />
```

3.8 Informativní prvky stránky

Tyto prvky zobrazují informace o webových stránkách. Hodnota těchto informativních prvků je dána nastavením v administračním systému. To přispívá k celkově snazší faktorizaci aplikace, i když se data tohoto typu v praxi tak často nemění. Mezi informativní prvky patří název aplikace, název stránky, autor, copyright, datum poslední aktualizace a aktuální datum. Tyto prvky jsou celkem jednoduché, proto uvedu jen stručné informace v následující tabulce.

Prvek	***IEML element***	***Příklad***
název aplikace	*webname*	`<$webname />`
název aktuální stránky	*pagename*	`<$pagname />`
autor	*autor*	`<$author />`
copyright	*copyrightautor*	`<$copyright />`
datum poslední aktualizace	*actdate*	`<$actdate />`
aktuální datum	*nowdate*	`<$nowdate />`

Tabulka 3.12: Seznam informativních prvků stránky

3.8.1 Prvek aktuální datum

Tento prvek je trochu složitější, proto ho zde budu více specifikovat. Pro nastavení formátu data obsahuje element *nowdate* atribut *format*. Klíčovým slovem *name-day* se zobrazí, kdo má aktuální den svátek. V následující tabulce je seznam všech atributů.

Atribut	***Význam***	***Možné hodnoty***	***Povinný***	***Default***
src	Určí jaký čas se má zobrazit.	today|tomorrow|yesterday	ne	today
format	Určí v jakém má být čas formátu.	př: 'm:d:y' nebo 'name-day'	ne	d.m Y
class	Třída kaskádových předloh.	string	ne	není

Tabulka 3.13: Seznam atributů a možných hodnot elementu nowdate

Příklad:

```
<$date src="tomorrow" class="day" format="name-day" />
<$date class="date" format="Y-m-d" />
```

3.9 Administrace databázové části

K administraci databázové části prostředí je přistupováno podobně jako je tomu u ostatních databázových administrátorů (phpMyAdmin). Jediným rozdílem je to, že databázové entity (tabulky, sloupce tabulek) jsou popsány několika dalšími metadaty určujících například viditelnost nebo textové informace, které se používají především pro konfiguraci databázových prvků stránek (kapitola: „Obsah stránek, IEML“).

3.9.1 Správa tabulek

Základní entitou v databázi je tabulka. V databázi pro toto prostředí se nachází dva typy tabulek. Systémové tabulky, které obsahují data zajištující interní chod prostředí. Těmi se zde nebudeme zabývat (kapitola: „Návrh databáze“). A uživatelské tabulky (uživatelem je zde myšlen administrátor aplikace). Rozhone-li se administrátor pro tvorbu nějaké nové funkčnosti, vyžadující ukládání dat, je právě toto nejvhodnější způsob. Stačí vytvořit uživatelskou databázovou tabulku (dále jen tabulku), nastavit několik parametru (vis. další kapitoly) a vytvořit nad touto tabulkou nějaký databázový prvek stránky (kapitola: „Obsah stránek, IEML“) a vše je hotové. Tabulky mají následující parametry:

Parametr	***Popis***	***Změna***	***Povinný?***
jméno	interní jméno tabulky v databázi	podpor. refaktorizace	ano
typ	kapitola: „Typy databázových tabulek“	podpor. refaktorizace	ano
titulek	externí (visuální) jméno tabulky	introspekce	ano
komentář	popisné informace o tabulce	introspekce	ne

Tabulka 3.14: Popis parametrů databázové tabulky

3.9.2 Struktura tabulky

Pokud máme vytvořenou tabulku, je potřeba vytvořit její strukturu, tzn vytvořit v tabulce nějaké sloupce. Způsob je podobný jako u ostatních databázových adminů. Po vytvoření tabulka implicitně obsahuje sloupec *id,* který je celočíselného typu a tvoří primární index tabulky. Tento sloupec nelze měnit ani odstranit s výjimkou popisných informací a viditelnosti (vis. dále). V tabulce je možné kromě tohoto indexu vytvořit i další indexy na jiných sloupcích (kapitola: „Tvorba indexů a relací tabulek“). Při vytvoření nového sloupce se musí nastavit několik parametrů, které je možné explicitně upravovat. Seznam parametrů je v následující tabulce.

Parametr	*Popis*	*Změna*	*Povinný?*
jméno	interní jméno sloupce v databázi	podpor. refaktorizace	ano
typ	datový typ [2]	podpor. refaktorizace	ano
spec. typu	upřesnění typu[2]	podpor. refaktorizace	ano/ne
vlastnosti	vlastnosti typu[2]	atomat. refaktorizace	ne
nulovost	určuje zda musí být hodnota zadána	podpor. refaktorizace	ano
logický typ	speciální vlastnosti (následující kapitola)	atomat. refaktorizace	ne
default	implicitní hodnota sloupce	atomat. refaktorizace	ne
extra	extra vlastnosti [2]	atomat. refaktorizace	ne
titulek	externí (visuální) jméno sloupce	introspekce	ano
komentář	popisné informace o sloupci	introspekce	ne
pořadí	pořadí sloupců v databázovém prvku str.	introspekce	ano
módy	mody viditelnosti (následující kapitola)	introspekce	ano

Tabulka 3.15: Popis parametrů sloupce databázové tabulky

3.9.3 Datové a logické typy

3.9.3.1 Datové typy

V předchozí kapitole jsme narazili na datové typy sloupců v databázi. Tyto datové typy je možné rozdělit do čtyř základních skupin:

- řetězcové (char, varchar, text, blob,...)
- číselné (int, float, double, decimal,...)
- časové (date, time, datetime, timestamp,...)
- výčtové (enum, set)

Bližší informace o datových typech v databázi lze získat třeba na webové stránce http://dev.mysql.com/doc/refman/5.1/en/data-types.html.

3.9.3.2 Logické typy

Oproti tomu logické typy jsem nadefinoval speciálně pro toto prostředí. Logický typ neříká jaký typ hodnoty obsahuje daná entita, ale jak se s touto entitou (potažmo hodnotou) bude zacházet v prostředí aplikace. Existuje ovšem vazba mezi logickým a datovým typem, neboť už sám datový typ naznačuje, jak by se z danou hodnotu mělo zacházet. Proto určité logické typy entity vyžadují, aby tato entita byla pouze požadovaného datového typu. Například logický typ „aktuální čas“ bude vyžadovat některý z časových datových typů. Seznam všech logických typů je v tabulce.

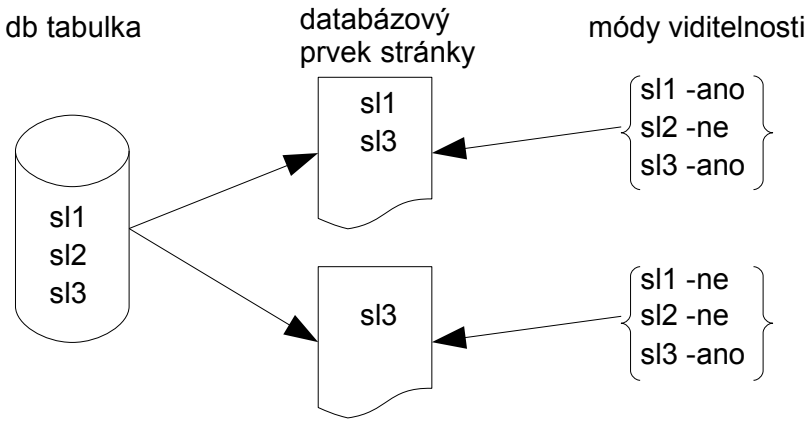
Jméno	*Popis*	*Datový typ*
heslo	Ve výstupních databázových prvcích stránky budou místo hodnoty zobrazovány hvězdičky. Ve vstupních databázových prvcích stránky bude zobrazeno ještě jedno formulářové políčko pro kontrolu hesla. Hodnota se uloží pomocí MD5.	varchar(32)
aktuální čas	Použije se v případě, když chceme mít informaci o čase ukládání záznamu. Pak stačí element označit jako neviditelný a použít tento logický typ. Při uložení dat formuláře se použije pro tento element aktuální časová hodnota.	časový typ
odkaz	Máme-li nějakou hodnotu symbolizující URL, tak pomocí tohoto logického typu z ní bude aktivní hypertextový odkaz.	řetězcový datový typ
email	Máme-li nějakou hodnotu symbolizující emailovou adresu, lze z ní udělat aktivní odkaz na emailovou adresu (mailto:...)	řetězcový datový typ
obrázek	Reprezentuje-li nějaká hodnota adresu obrázku. Pomocí tohoto logického typu bude zobrazen místo hodnoty adresy obrázek.	řetězcový datový typ

Tabulka 3.16: Logické typy entit

Pokud entita výše uvedených logických typů nebude mít odpovídající hodnotu, nebude tento logický typ v aplikaci akceptován. Typová kontrola vztahující se k datovým typům je prováděna již při volbě logického typu. Počítám, že se seznam logických typů bude dále rozrůstat o další členy.

3.9.4 Viditelnost a mody

Mody slouží k nastavení viditelnosti jednotlivých sloupců tabulky, použité jako zdroj databázového prvku stránky. Tudíž relace mezi modem viditelnosti a tabulkou v databázi je nastavená v konkrétním databázovém prvku stránky a je platná pouze pro tento prvek. Tzn. databázová tabulka může mít více módu (maximálně však tolik v kolika je prvcích stránek) a mod může být použit pro libovolný počet tabulek (i vícekrát pro tutéž tabulku). Asi nejlépe to jde vidět na obrázku.



Obrázek 3.7: Schéma relací mezi tabulkami a mody viditelnosti

Viditelnost sloupců tabulek se upravuje v administrátorském prostředí aplikace stejně jako vytváření nových modu, mazání a úprava stávajících. Implicitně je nadefinován mod *root*, ve kterém jsou viditelné všechny entity a jehož nastavení nelze měnit. Vytvoření databázového prvku stránky s vazbou na databázovou tabulku a mody viditelnosti je popsáno v kapitole: „Obsah stránek, IEML“.

3.9.5 Tvorba indexů a relací tabulek

Práce s indexy a relacemi mezi tabulkami se velice podobá práci v aplikaci phpMyAdmin.

3.9.5.1 Tvorba indexů

Existují tři základní důvody proč na sloupci (sloupcích) vytvořit index (kromě primárního indexu který je v tabulce implicitně vytvořen). Prvním důvodem je, zaručení unikátnosti hodnoty ve sloupci (sloupcích). To je možné provést pomocí indexu *unique*. Druhý důvod je umožnění fulltextového vyhledávání ve sloupci. To lze zajistit indexem typu *fulltext* (pozor toto nepodporují všechny typy tabulek). Třetím důvodem je vytvoření relace mezi sloupci dvou různých tabulek, zde postačí obyčejný typ *index*.

3.9.5.2 Tvorba relací

Relace jsou mocným nástrojem ve světě databází, které nám umožní udržovat konzistenci mezi tabulkami. MySQL podporu pro relace jen pro některá úložiště a pokud používáme výchozí MyISAM, tak jsme o tuto možnost ochuzeni. Vytvoření relací mezi tabulkami je opět velmi podobné jako v Aplikaci phpMyAdmin. Stačí určit mezi kterými sloupci tabulek se má relace vytvořit a určit chování, pokud by mělo dojít k porušení referenční integrity. Je možné použít následující typy referenční integrity:

- cascade (smaže/aktualizuje všechny podřízené záznamy)
- set null (zrušení operace, pokud níže v kaskádě je RESTRICT nebo NO ACTION)
- no action (neprovede nic)
- restrict (zakáže provedení akce)[3]

3.9.6 Data tabulky

Pokud by nějakým způsobem došlo k nekonzistenci dat v databázi nebo byla vložena nepřípustná data, musí existovat možnost jak to napravit. V administrační části prostředí je sekce pro úpravu dat uživatelských databázových tabulek. Je možné změnit, smazat nebo přidat data, bez ohledu na to, jakým uživatelem byla vytvořena. Je nutné dbát nejvýší opatrnosti, neboť není prováděna žádná kontrola integrity dat s dalšími entitami prostředí (kromě referenční integrity). Kontrola logických a datových typů je samozřejmě aktivní.

3.10 Menu a systém stránek

Jedním z nejdůležitějších prvků, sloužících pro navigaci a procházení jednotlivými stránkami aplikace, je menu. Práce s těmito komponentami je popsána v kapitole: „Řídící prvky“. Menu, stejně jako mapa stránek, presentuje strukturu stránek v aplikaci. Tvorbou a úpravou struktury aplikace se budeme zabývat právě v této kapitole.

3.10.1 Stránka, parametry stránky

Stránka, jako prvek struktury stránek, je definována několika parametry. Některé z parametrů je nutné zadat při tvorbě stránky a už s nedají explicitně měnit. Jiné obsahují pouze popisné informace a jejich změna je introspektivní. Kompletní seznam všech parametrů stránky je v následující tabulce:

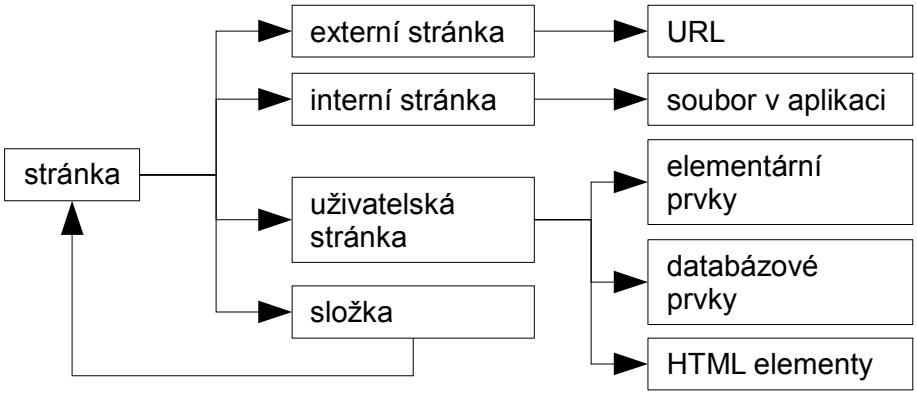
Parametr	***Popis***	***Změna***	***Povinný?***
jméno	Jméno stránky zobrazené v menu, na mapě stránek i v jiných odkazech na tuto stránku.	introspekce	ano
komentář	Text zobrazující se jako doprovodná informace při najetí myši na odkaz k této stránce.	introspekce	ne
pozice	Určuje pozici stránky v menu nebo na mapě stránek.	introspekce	ano
typ	Určuje typ stránky (vis. následující kapitola)	nelze měnit	ano
rodič	Určuje umístění stránky ve struktuře stránek (vis. následující kapitola).	introspekce	ano
viditelnost	Má-li být stránka viditelná. Vhodné pokud chceme stránku dočasně vypnout.	introspekce	ano
v menu	Stránka bude zobrazena v roletovém nebo děleném menu (kapitola: „Řídící prvky“).	introspekce	ano
v hlavičce	Stránka bude zobrazena v hlavičce odkazů (kapitola: „Řídící prvky“).	introspekce	ano
klíčová sl.	Sada klíčových slov specifických pro tuto stránku. Přidává se ke standardní sadě klíčových slov aplikace.	introspekce	ne
ACI	Index bezpečnosti (kapitola: „Přihlašování, správa účtů, řízení přístupu“).	introspekce	ano
obsah	Vlastní obsah stránky. (kapitola: „Obsah stránky, IEML“)	introspekce	ano

Tabulka 3.17: Parametry stránky ve struktuře stránek

Doplňující informace: Skryjeme-li stránku pouze tak, že jí vypustíme z menu a hlavičky, bude tato stránka přístupná z případného odkazu v obsahu jiné stánky. Stoprocentní skrytí stránky lze docílit pouze pomocí parametru viditelnost.

3.10.2 Typy stránek, struktura

Rozlišujeme čtyři základní typů stránek: složky, uživatelské stránky, interní stránky a externí stránka. Typ stránky je zvolen při její tvorbě a nedá se explicitně změnit. To proto, že by se jednalo o zásadní změnu přístupu k obsahu stránek. Její změna by nebyla ani automatickou refaktorizací, což by vedlo k obrovským problémům při správě menu. Každá stránka musí mít určenou rodičovskou stránku. Rodičovskou stránkou může být pouze stránka typu složka nebo kořenový adresář *root*.



Obrázek 3.8: Schéma struktury systému stránek

3.10.3 Složky

Jak již název napovídá, jedná se o stránku, která v sobě obsahuje dceřiné stránky. Pomocí složek lze vytvořit libovolnou stromovou strukturu systému stránek, potažmo menu (předchozí obrázek). Tento typ stránek nemá žádný vlastní obsah. Proto je možné obsah nastavit. Tzn. nastavit co se zobrazí při kliknutí na tuto stránku. V zásadě jsou čtyry možnosti:

- nezobrazí se nic
- zobrazí se seznam potomků
- zobrazí se obsah prvního potomka
- zobrazí se obsah posledního potomka

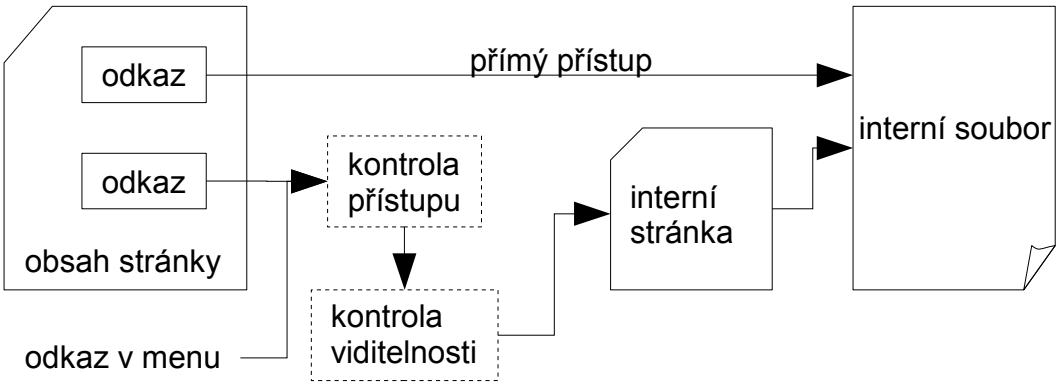
Toto lze nastavit pouze globálně (tj. všechny složky se budou chovat stejně), aby nedocházelo k rozporům při procházení menu stránek. Může se zdát, že prakticky použitelná je pouze druhá možnost, ale není tomu tak. Používá-li se v aplikaci pouze dělené menu (kapitola: „Interaktivní prvky“) je vhodné použít třetí nebo čtvrtou variantu, protože mít seznam potomků zobrazen duplicitně není žádoucí.

3.10.4 Uživatelské stránky

Tento typ stránek je charakteristický tím, že má plně volitelný obsah. Stránka může obsahovat skoro libovolná data a mnoho před implementovaných funkčností. Podrobnější popis práce s obsahem stránek je popsán v kapitole: „IEML a prvky stránky“.

3.10.5 Interní stránky

Interní stránka je taková, která obsahuje odkaz na interní soubor (kapitola: „Elementární prvky stránky). Tj. nějaký dokument nacházející se serveru. Interní stránka může odkazovat pouze na jeden interní soubor. Ovšem na jeden interní soubor se může odkazovat více interních stránek. Může se zdát, že používaní interních stránek kromě menu je zbytečné, protože je možné jednoduše na potřebný objekt vytvořit odkaz. Tím se ale připravíme o možnost řízení přístupu k obsahu stránky, volbu viditelnosti, definici klíčových slov a další výhody. Pokud budeme striktně používat interní stránky pro přístup k interním souborům, budeme mít nad těmito soubory plnou kontrolu (kapitola: „Přihlašování, správa účtů, řízení přístupu“). Explicitní změna odkazu na zdrojový interní soubor je samozřejmá.



Obrázek 3.9: Způsoby přístupu k internímu souboru

3.10.6 Externí stránka

Externí stránka odkazuje na nějakou externí adresu. Tzn. na webovou stránku mimo server (respektive mimo aplikaci). Stránka může být umístěna v menu nebo v obsahu stránky. Použití externích stránek v obsahu stánky má stejné vlastnosti jako v předchozím případě, avšak zde postrádá hlubšího významu. Tj. stránka nacházející se na jiném serveru, ke které se přistupuje pomocí URL je (musí být) veřejně přístupná, protože nedochází k žádnému sdílení cockies ani sessions. Tedy vlastní zabezpečení externí stránky postrádá významu. Jediné co se může v některých případech hodit je volba viditelnosti.

3.11 Přihlašování, správa účtů, řízení přístupu

Jako každá složitější aplikace musí i toto prostředí pro správu interaktivních webových aplikací podporovat přístup rozdělený na uživatele z různými přístupovými právy. Navíc je tu implementováno něco, co by se s trochou nadsázky dalo považovat za povinné řízení přístupu. Podrobněji to vysvětlím v kapitole řízení přístupu. Ovšem nejdříve je nutné se seznámit s úrovněmi přihlašování a celkovým bezpečnostním schématem.

3.11.1 Úrovně přihlašování

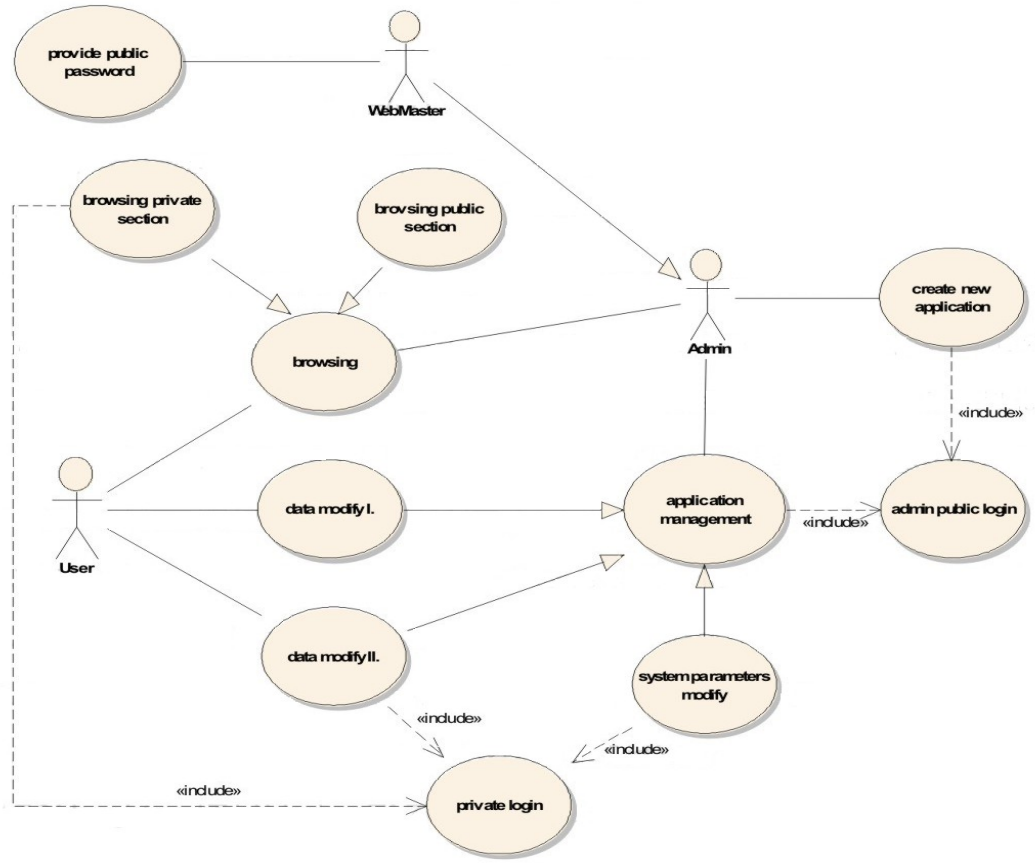
V tomto prostředí existují čtyři základní úrovně uživatelů a čtyři základní úrovně hesel. Není ale pravdou, že si tyto dvě skupiny navzájem odpovídají. Na nejvyšším stupni je webmaster, poté administrátor aplikace, registrovaný uživatel a nejníže neregistrovaný uživatel. Webmaster není osoba reálně implementovaná v systému. Je to jakási virtuální osoba kořenové úrovně. Tuto funkci nemůže zastávat administrátor, jelikož prostředí je navrženo jako multi-aplikační. Administrátorem je považována osoba mající přístup a starající se pouze o jednu konkrétní aplikaci (popřípadě více, ale jde pouze o jeho vlastní aplikace). Registrovaný uživatel má přístup do některých privátních sektorů webové aplikace (záleží na řízení přístupu). Ne však do administračního systému. Neregistrovaný uživatel má přístup pouze do veřejných sektorů.

Nejvyšší utajení mají takzvaná super hesla, poté jsou hesla pro přístup do administrační části konkrétní aplikace (administrační hesla), hesla pro vstup do rozcestníku administrační části a na konec uživatelská hesla. Super hesla slouží pro změnu administračních hesel. Relace mezi uživateli a hesly je v následující tabulce.

Uživatel	***Heslo***
webmaster	super heslo veřejné heslo
administrátor	super heslo veřejné heslo administrační heslo
registrovaný uživatel	uživatelské heslo
neregistrovaný uživatel	bez hesla

Tabulka 3.18: Relace mezi uživateli a hesly

Při jakých příležitostech se hesla používají a kam mají přístup uživatele výše uvedených kategorii je nakresleno na use-case diagramu. Jednotlivé případy užití jsou popsány níže.



Obrázek 3.10: Diagram případů užití, správa přístupů

Poskytnutí veřejného hesla, správce webových stránek (webmaster) poskytne oprávněné osobě veřejné heslo pro přístup do administračního systém. Ten si tímto může založit vlastní webovou aplikaci a stává se jejím správcem.

Přihlášení veřejným heslem, jde o přihlášení do obecné části administrační aplikace.

Přihlášení vlastním heslem, po přihlášení do obecné části administrační aplikace se lze přihlásit do své vlastní aplikace vzniklé podle bodu 2. Nebo jde o přihlášeni uživatele vlastní webové aplikace.

Vytvoření nové aplikace je možné za výše uvedených podmínek. Budoucí administrátor zadá název aplikace, své jméno a heslo. Posléze bude vygenerována kostra aplikace s implicitním nastavením.

Prohlížení obsahu se dělí na dvě části. Část veřejnou ke které mají přístup všichni návštěvníci stránek. A část soukromou kterou mohou vidět pouze registrovaní návštěvnicí stránek, administrátor a webmaster.

Správa aplikace se dělí na tři části. Systémové změny, Datové změny I. a Datové změny II.

Systémové změny jsou všechny takové změny, které ovlivňují chod systému nebo jeho komponent. Systémovými změnami jsou:

- přidání, modifikace a odstranění databázové tabulky

- přidání, modifikace a odstranění sloupce, indexu nebo relace databázové tabulek
- přidání, modifikace a odstranění modu viditelnosti dat databázové tabulky
- přidání, modifikace a odstranění viditelnosti databázové tabulky
- přidání, modifikace a odstranění článku
- přidání, modifikace a odstranění obrázku
- přidání, modifikace a odstranění ankety
- přidání, modifikace a odstranění souboru
- přidání, modifikace a odstranění zprávy administrátora
- přidání, modifikace a odstranění položka menu a stránky aplikace
- přidání, modifikace a odstranění uživatele
- přidání, aktivace a odstranění souboru s kaskádovými styly
- modifikace návrhu struktury okna aplikace
- modifikace jazykových řetězců (nelze přidávat a mazat)
- modifikace bázových informaci (jméno aplikace, ikony, autor, klíčová slova,..)
- vyprázdnění auditních dat
- změna administračního hesla pomocí super hesla

Datové změny I. jsou veškeré datové změny takové, které může provádět kdokoli, tzn. i návštěvník webových stránek. Jsou to:

- hlasování v anketě
- příspěvek ve fóru

Datové změny II. jsou takové změny dat, které může provádět jen přihlášený uživatel, administrátor nebo webmaster. Těmi jsou:

- přidání, modifikace a odstranění dat některých databázových tabulek
- založení nového topicku ve fóru

3.11.2 Správa účtů

Správou účtů je myšlena především správa uživatelských účtů (nikoli administrátorských-aplikací). Správa uživatelských účtů se provádí v administrátorské aplikaci, kde je zobrazená tabulka všech registrovaných uživatelů. Tabulka obsahuje tři sloupce. Prvním z nich je uveden nick uživatele, v druhém hash hesla uživatele v MH5 a posledním ACI uživatele (vis následující kapitola). Pokud bychom používali aplikaci třeba jako internetový obchod a vznikla by potřeba přidat další sloupce (informace o uživateli), není to možné. Řešením je vytvořit si vlastní databázovou tabulku uživatelů s vlastními údaji (možnost použít i logické typy) a povázat je pomocí cizího klíče tabulky (kapitola: „Databázové prvky stránky“).

Veškeré údaje v tabulce lze změnit bez vědomí uživatelů. Z toho plyne paradox, že administrátor nemůže zjistit heslo registrovaného uživatele, ale může mu ho změnit. Což je z hlediska bezpečnosti velmi výhodné, protože většina uživatelů má v oblibě používat v mnoha aplikacích stále

stejná hesla (např. email, e-shop). Na druhou stranu dost uživatelů svá hesla zapomíná. V tomto případě administrátor uživateli přidělí nové heslo.

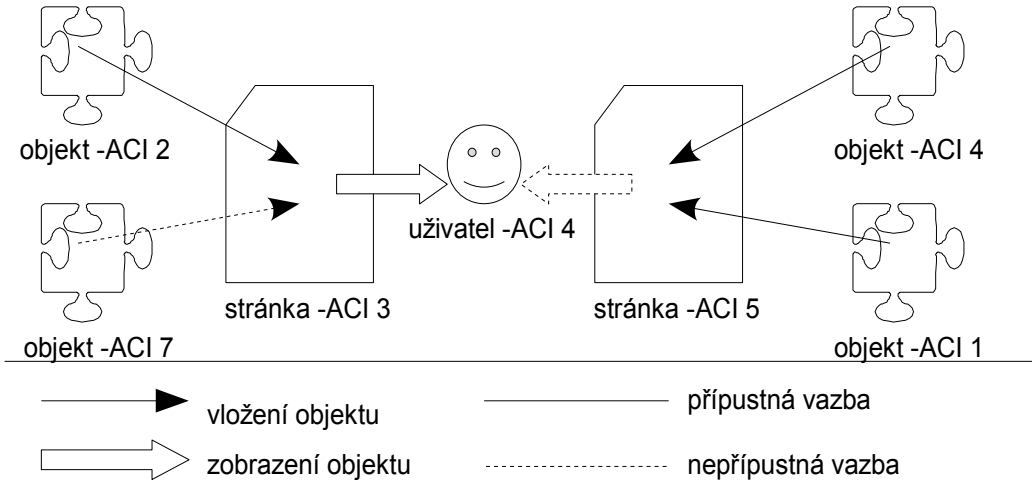
Samozřejmě lze přidat nového uživatele nebo odstranit uživatele stávajícího. Možnost registrovat se a měnit své údaje je uživateli odepřena, z důvodu složité implementace řízení přístupu. Vše může provádět pouze administrátor. Řešení tohoto problému je otázka příštích verzí.

3.11.3 Řízení přístupu

Jak již bylo v úvodu řečeno, dalo by se s nadsázkou říci, že je zde implementováno povinné řízení přístupu. Každý objekt (např. stránka, obrázek, článek, uživatel) má v popisných informacích uvedeno takzvané ACI (acces control index - index řízení přístupu). Je to číselná hodnota 0 až 9. Objekt s ACI nula je volně zobrazitelný. Oproti tomu objekt s ACI 9 má nejvyšší míru utajení.

ACI popisuje nejen míru utajení objektu, ale i míru přístupu objektu k ostatním objektům. To znamená, že objekt nemůže obsahovat (přistupovat k) objekt(u) s vyšším ACI, než má on sám. Může obsahovat (přistupovat) pouze (k) objekt(u) s nižším nebo stejným ACI. Například na stránce s ACI 3 nemůže být článek s ACI 7, a tuto stránku si nemůže zobrazit uživatel s ACI 1 (vis. obrázek).

Možná se může zdát, že je toto opatření pro aplikaci tohoto typu zbytečné. Ale představme si situaci, kdy se na soukromé stránce nachází například obrázek (mimo jiných). Obrázky se na stránky vkládají z databáze obrázků, takže jeden obrázek může být zobrazen na mnoha stránkách. Jak tedy rozumě kontrolovat, aby se obrázky na soukromích stránkách neobjevili omylem i na stránkách volně přístupných. V horším případě, jak zabránit neregistrovanému uživateli zobrazit výsledek vyhledávání obrázků v aplikaci, aniž bychom věděli, jaký z nich je na soukromé stránce a jaký na veřejné. To by znamenalo zavedení nějakého n-arního stromu objektů (v lepším případě) a při každé takové operaci by se tento graf musel prohledávat. Což by při velkém množství dat bylo značně komplikované. Proto jsem raději zvolil způsob podobný povinnému řízení přístupu.



Obrázek 3.11: Schéma řízení přístupu k objektům

Z obrázku lze vypozorovat jeden nedostatek. Uživateli s ACI 4 není zobrazen objekt s ACI 4 a ACI 1, ač by na to měl mít nárok. To je ovšem jen zdánlivé neboť se na stránce s vyšším ACI předpokládá přítomnost objektu minimálně stejného ACI. Kdybychom připustili zobrazení takové stánky (při nezobrazení objektu s vyšší ACI), obsah by mohl být neúplný či chaotický. Navíc se předpokládá, že oprávnění k přístupu je určováno především ACI stránky a je velmi pravděpodobné, že objekt s ACI 1 a ACI 4 je zobrazen na nějaké, tomuto uživateli přístupné stránce, neboť by jinak nebylo potřeba, aby stránka s ACI 5 měla takto vysoký koeficient.

3.12 Podpora více jazyků

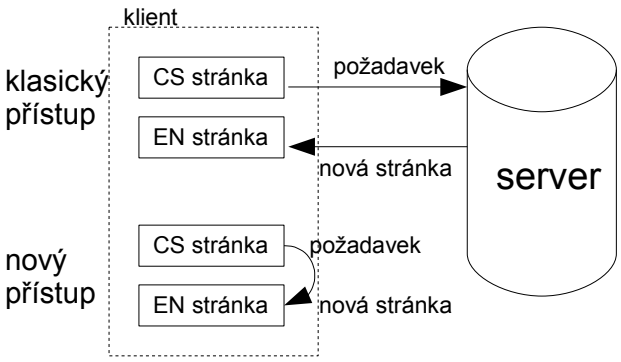
V dnešní multikulturní době, je skoro nepostradatelné, aby webové stránky obsahovali informace alespoň ve dvou jazycích (národní jazyk a angličtina), ne-li více. Tím spíše jedná-li se o stránky komerční, mající reklamní a propagační charakter.

3.12.1 Změna jazyka

U některých stránek na internetu dochází při změně jazyka k úplné strátě kontextu (ocitneme se opět na indexové stránce). To je ovšem extrémní případ (http://www.vodafone.cz, http://www.studentagency.cz). Další možností je, že sice nedojde k úplné ztrátě kontextu, ale pouze refreshi stránky nebo přechodu na jinou stránku se stejným obsahem v požadovaném jazyce. Toto je nejčastěji používaná varianta (http://www.czechcomputer.cz, http://www.fit.vutbr.cz). Ve většině případů je tato možnost plně postačující. Komplikace by mohli nastat při posílání *post dat* na server (ukládání dat pomocí formuláře). Je-li rychlost připojení dostatečně malá a změna jazyka se provede v nevhodnou chvíli, mohlo by dojít k nekonzistenci dat v databázi, duplicitě dat nebo se data nemusí uložit vůbec (díky nějakému kontrolnímu mechanizmu). Tento problém je nutné explicitně nějak ošetřit.

Abych eliminoval výše uvedené problémy, vymyslel jsem způsob, jak měnit jazyky aniž by došlo k refreshi stránky nebo přechodu na stránku jinou. Princip je následující. Obsah je na HTML stránce uložen ve všech jazycích, které aplikace podporuje. Pomocí panelu jazyků (kapitola: „Panel jazyků“) se aktivují pouze řetězce v požadovaném jazyce. Podrobný popis funkce bude popsán v implementační části práce.

Výhodou tohoto provedení je, jak již bylo řečeno, že nedochází k sebemenší ztrátě kontextu. Dále při změně jazyka nedochází k posílání dat na server a vše se provádí pouze na klientovi. Což může přispět ke zvýšení rychlosti. To je pouze relativní, neboť je o to více dat posíláno při prvním načtení této stránky, avšak místo dvou spojení je provedeno pouze jedno. Pokud si návštěvník stránek stáhne z internetu celou stránku, bude mu přepínání mezi jazyky fungovat i v offline režimu na jeho počítači. Otázkou zůstává vhodnost z pohledu SEO optimalizace.



Obrázek 3.12: Změna jazyka stránky

3.12.2 Modifikace jazykových řetězců

V administrační části prostředí je možné upravovat textové řetězce všech v aplikací používaných jazyků. Řetězce jsou seřazeny primárně podle typů a sekundárně podle abecedy. Je-li potřeba v aplikaci změnit nějaký řetězec, stačí si všimnout v jaké části aplikace se nachází. V administrační části přepnout na stránku s požadovaným jazykem a nalézt ho v seznamu řetězců v příslušné kategorii. Jako identifikátor řetězce je použita jeho stávající hodnota. Po uložení se stará hodnota nahradí hodnotou novou na všech místech aplikace.

3.13 Audit

Nedílnou součástí větších interaktivních aplikací je audit. Záznam všech relevantních událostí velmi usnadňuje práci, dojde li k nějaké závažné chybě, nějakému bezpečnostnímu incidentu nebo chceme li z nějakého důvodu zmapovat chováni některého z uživatelů. Je také dobré, když je možné implicitně nastavit jaké události se mají zaznamenávat. Typicky to může být přidání dat do databáze, modifikace dat, mazání dat, chybové události, přihlášení a odhlášení uživatele, zobrazení privátních sekci, ale také hlasování v anketě, přidání příspěvku ve fóru a další. Velikost takových auditních dat s časem velmi roste, proto je nutné mít nějaké filtrovací a řadící mechanizmy. Je také možné občas auditní data z databáze vyexportovat nebo auditní seznam vyprázdnit.

3.14 Zamčení a odemčení aplikace

Měla by také existovat možnost aplikaci zamknout či odemknout, bude-li nutné provést nějakou změnu zásadnějšího rázu. Takovou změnou je vetšinou změna systémových parametrů databázových tabulek, úprava rezistence dat databáze a další podobné změny týkajíce se většinou práce s databází.
Tuto možnost není dobré používat bez rozmyslu. Vždy je to volba mezi dvěma zly, a je nutné pečlivě zvážit které z nich je menší.

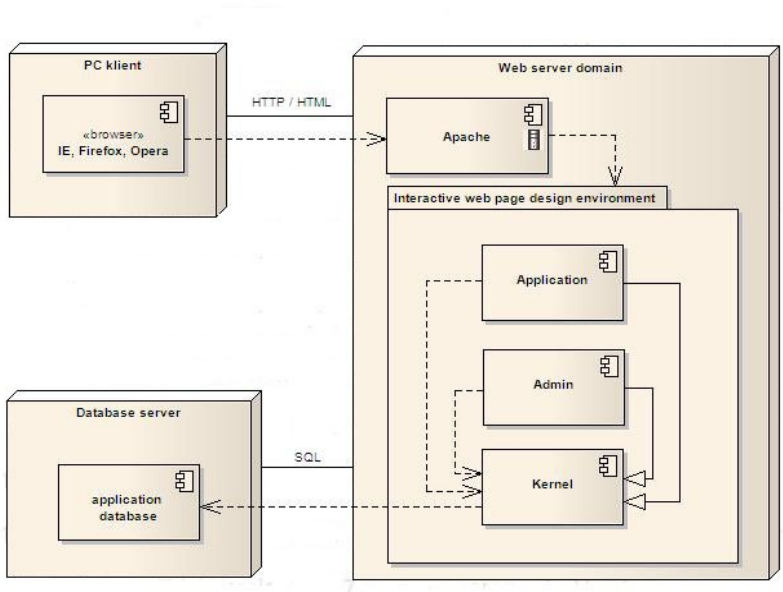
4 Implementace

Doposud jsme se zabývali pouze ideovým návrhem funkcionality, schopnostmi a dovednostmi prostředí pro tvorbu a návrh interaktivních webových stránek. Nyní tedy přikročme k vlastní implementaci aplikace. Toto prostředí je tvořeno výhradně objektovým a modulárním způsobem. Není ale možné zde popisovat všechny funkce, metody, objekty ba ani moduly. Aplikace je na tolik rozsáhlá (cca 30 tisíc řádků kódu), že by to zabralo několik set stránek. Tvorba jakési programátorské dokumentace není ani cílem této diplomové práce.

Je ale nutné, alespoň rámcově přiblížit návrh struktury, návrh databáze a objektový návrh toho prostředí pro tvorbu interaktivních stránek. Protože si myslím, že jeden obrázek řekne více než deset stran textu, bude převážná část vysvětlena pomocí diagramu v jazyce UML2. Složitější a rozměrnější diagramy budou v přílohách práce.

4.1 Základní rozvržení prostředí

Celé toto prostředí je možné rozdělit na tři základní části. První část je samotná webová aplikace, která je přístupná všem u návštěvníkům internetu. Je to vlastně výsledek naší snahy. Dále nesmí chybět administrační aplikace, pomocí které tvoříme a spravujeme část předchozí. Třetí, a však nejdůležitější, část, která je skryta očím všech návštěvníků, uživatelů, webmastrů a administrátorů, je jádro prostředí. Zde jsou uloženy všechny moduly a třídy společné pro obě předchozí části. To vše pracuje nad jednou databází, ve stejném souborovém systému a doménovém prostoru (tzn. není možná správa aplikace pomocí vzdáleného přístupu nebo z jiného serveru, aspoň prozatím).



Obrázek 4.1: Architektura prostředí pro webový server (UML diagram)

4.2 Návrh databáze

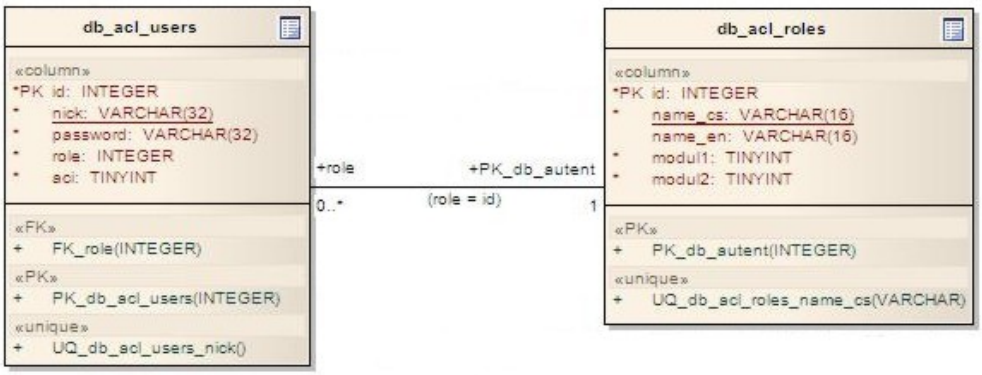
Jak už bylo řečeno celé prostředí (ač se jedná o více aplikací) pracuje nad jedinou databází. Databáze a systém souborů jsou zde jediná shromaždiště dat. Nyní se nabízí otázka co je lepší uložené v databázi a co nikoli. Zde jsem zvolil takovýto přístup: strukturovaná data se ukládají do databáze(metadata, uživatelé, atd.), kdežto data mající blokový charakter se ukládají v souborovém systému (vygenerované php skripty, obrázky, atd.). Při použití výkonnějšího databázového systému než je MySQL server, by toto kriterium šlo přehodnotit.

V předchozích kapitolách jsem se zmínil o tom, že toto prostředí je zároveň i správce databáze. To ovšem je jen poloviční pravda, neboť lze spravovat jen některé tabulky. Tabulky v databázi se dělí na dvě kategorie: systémové a uživatelské. Systémové tabulky obsahují data důležitá pro správu a chod celého prostředí, tudíž není možné je přímo editovat pomocí databázového správce. Editace je možná pomocí rozhraní modulů k nimž konkrétní tabulka přináleží. Uživatelské tabulky je možné editovat způsobem popsaným v kapitole „Administrace databázové části“. Tyto tabulky nemají pevně danou strukturu a v prostředí nemusí ani žádná existovat (pokud ji administrátor nevytvoří), tudíž s návrhem databáze v tomto pojetí nemají nic společného. Nebudeme se jimi tedy nadále zabývat.

V následujících kapitolách stručně popíši strukturu a význam systémových databázových tabulek. Princip jako takový, bude vysvětlen u třídy nebo modulu k němuž tabulka náleží. Celkový diagram databáze včetně relací mezi tabulkami je v příloze D) Diagram struktury databáze.

4.2.1 Tabulky pro řízení přístupu

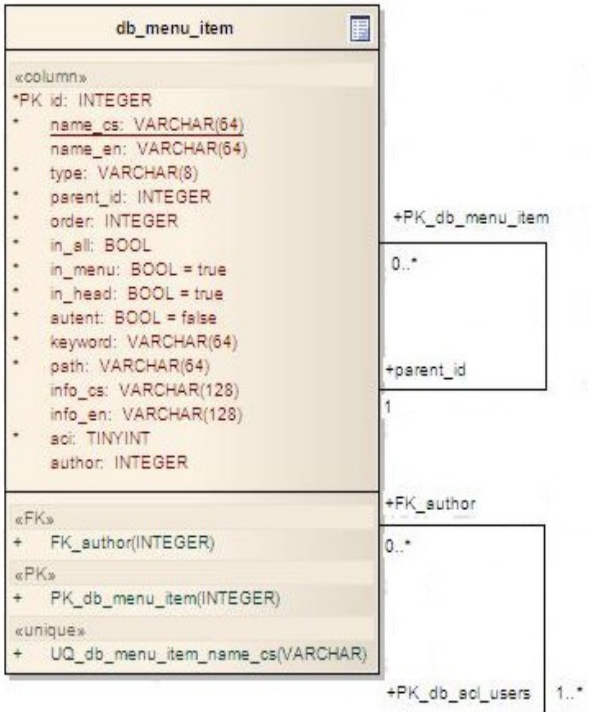
Pro řízení přístupu jsou potřeba dvě tabulky *db_acl_roles* a *db_acl_users*. Tabulka *db_acl_roles* obsahuje uživatelské role (řádky) a příslušné přístupové hodnoty k modulům aplikace (sloupce). Tyto hodnoty jsou v databázi uloženy jako číselný typ *tinyint(3)*. To je kvůli šetření paměti, neboť modulů a rolí může být velmi mnoho (velikost tabulky pak roste kvadraticky). Tabulka *db_acl_roles* obsahuje informace o uživatelích, včetně příslušnosti k roli a ACI.



Obrázek 4.2: UML diagram tabulek pro řízení přístupu

4.2.2 Tabulka pro práci s menu

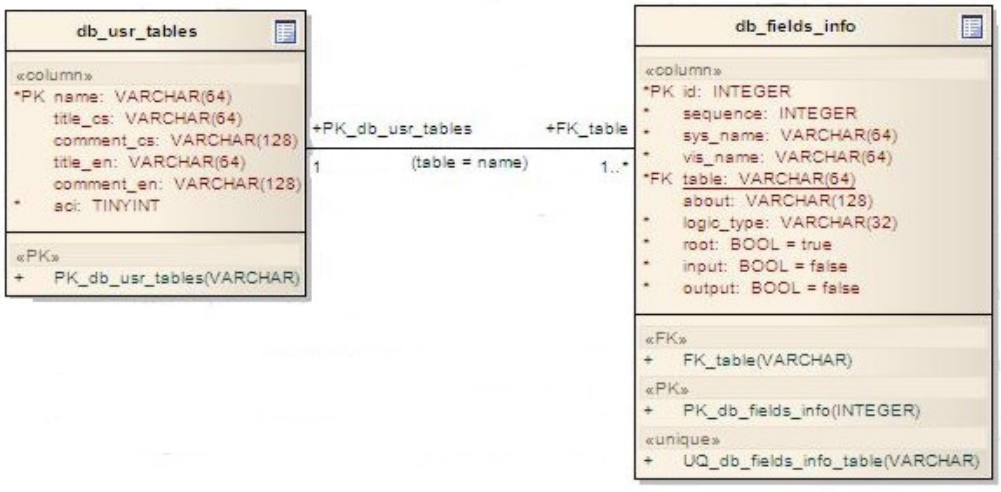
Databázová tabulka *db_menu_item* je z pohledu struktury aplikace tou nejdůležitější tabulkou. Určuje strukturu všech menu v aplikaci, pořadí položek (stránek), viditelnost a údaje potřebné pro řízení přístupu k obsahu stránek. Obsahuje ale také názvy a popisky stánek v aplikace ve všech podporovaných jazycích. Přesnější popis byl vylíčen v kapitole „Menu a systém stránek“. Co však v této tabulce není, je vlastní obsah stránky. Nachází se zde pouze cesta k souboru, v němž je uložen php skript pro zobrazení obsahu dané stránky. Neboť obsah stránky nemusí být pouze statický text, ale i libovolný prvek stránky (vis. předchozí kapitoly).



Obrázek 4.3: UML diagram tabulky pro práci s menu

4.2.3 Tabulky pro práci s databází

Pro práci s uživatelskými databázovými tabulkami jsou zapotřebí tyto dvě tabulky db_usr_tables db_fields_info. Jsou zde uloženy popisná data, komentáře a také informace potřebné k řízení přístupu k uživatelským tabulkám (tb: db_usr_tables) a sloupcům (tb:db_fields_info). U sloupců jsou navíc ještě mody viditelnosti, pořadí sloupců a logický typ. Podrobnější vysvětlení je v kapitole: „Administrace databázové části“.



Obrázek 4.4: UML diagram tabulek pro práci s databází

4.2.4 Tabulka s daty prvků stránek

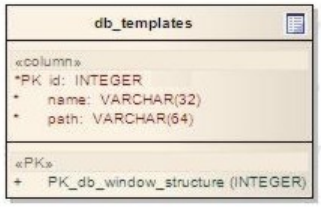
V databázi je také spousta tabulek obsahující data jednotlivých modulů. Jsou to například tabulky:

- *db_articles (modul article)*
- *db_messages (modul admin message)*
- *db_forum (modul forum)*
- *db_extern (modul pictures, files)*
- *db_opin_answ (modul opinios)*
- *db_opin_question (modul opinios)*

Více informací najdete v kapitole: „Moduly“, kde jsou některé moduly blíže specifikovány. Diagramy těchto tabulek lze nanést v příloze D) Diagram struktury databáze, kde jsou zobrazeny i relace mezi tabulkami.

4.2.5 Tabulka se šablonami

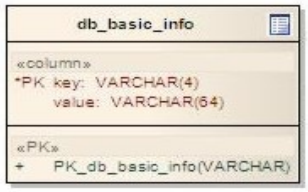
Jak již bylo řečeno v kapitole „Webové šablony“, aplikace může používat vícero webových šablon. Seznam těchto šablon je uložen v databázové tabulce *db_templates*. Ve sloupci *name* je jméno šablony a ve sloupci *path* je cesta k souboru *index.html* příslušné šablony.



Obrázek 4.5: Tabulka šablon (UML)

4.2.6 Tabulka základních informací

V tabulce *db_basic_info*, jsou uloženy základní informace o aplikaci. Ve sloupci je *key* je unikátní klíč informace a ve sloupci *value* je její hodnota. Těmi to informacemi mohou být například: jméno stránek, autor, copyright, datum poslední aktualizace, úvodní stránka, aktivní šablona, volba ikon obsažených v základní kostře aplikace atd.



Obrázek 4.6: Tabulka se základními daty (UML)

4.2.7 Tabulka záznamu událostí

Tabulka *db_events_log* složí pro ukládání událostí odehrávajících se v celém prostředí. V sloupci *type* je uložen typ události, ve sloupci *user* je *id* uživatele který byl v té době (sloupec *time*) přihlášen. V neposlední řadě zde musí být uložena vlastní hláška událostí (sloupec *message*). Více informaci v kapitola: „Audit“.

Obrázek 4.7: Tabulka záznamů událostí (UML)

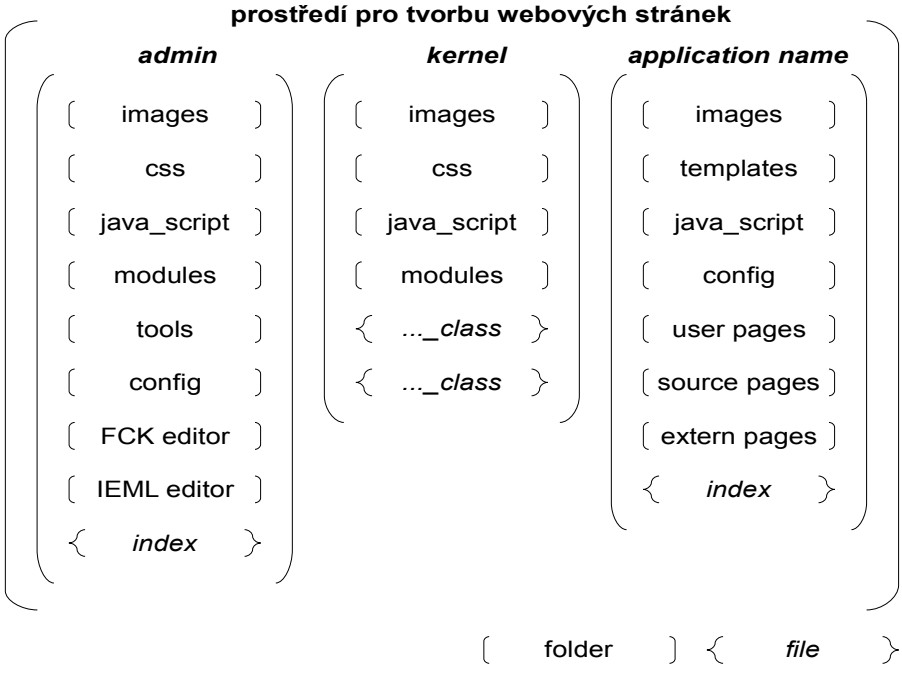
4.3 Kostra prostředí a abstraktní třídy

Jak již bylo uvedeno výše, prostředí pro tvorbu internetových aplikací se skládá ze tří hlavních částí: vlastní aplikace, administrační části a jádra. Byla snaha toto prostředí programovat modulárně za pomoci objektového přístupu, proto je zde spousta modulů a tříd. Některé moduly jsou v části jádra jiné v adminu, některé třídy jsou rodičovské jiné dceřiné, abstraktní nebo obyčejné, to vše vyplývá z podstaty jejich funkce a použití. Hlavní myšlenku této struktury se pokusím nastínit v následujících kapitolách.

4.3.1 Adresářová struktura

Podstatným faktorem zpřehlednění vnitřní struktury aplikace je uspořádání systému souborů. Moduly a třídy, které jsou využívány jak aplikací tak adminem, jsou umístěny v adresáři *kernel*. Moduly a třídy používané pouze adminem jsou v adresáři *admin*. V tomto adresáři jsou také obrázky, kaskádové předlohy, konfigurační soubory, soubory s java scriptem a ostatní soubory potřebné pro chod administrační části prostředí.

Jelikož toto prostředí může obsahovat i více aplikací, je každá taková aplikace ve vlastním adresáři. Zde je kladen důraz na to, aby se pokud možno žádná funkčnost, data nebo zdrojové kódy nevyskytovali duplicitně. Proto je obsah adresáře aplikace omezován na co nejmenší objem dat. Nachází se zde pouze indexový soubor, obrázky, konfigurační soubory a všechny administrátorem vytvořené a konfigurovatelné části. Vše je na následujícím obrázku.



Obrázek 4.8: Adresářová struktura

4.3.2 Abstraktní rodičovské třídy

Jako v každé aplikaci tak i zde se nachází sekvence kódu, které se používají stále dokola. Toto velmi dobře řeší objektově orientovaný přístup k programování. Jelikož je i toto prostředí psáno objektově, vyskytují se i zde rodičovské třídy i třídy abstraktní. Ty nejdůležitější budou krátce zmíněny v této kapitole.

4.3.2.1 Třída jm_object

Třída *jm_objec*t je abstraktní rodičovskou třídou pro všechny třídy v celém prostředí pro tvorbu webových aplikací. Tato třída obsahuje metody:

- pro úpravu řetězců
- pro načítaní dat z proměnných *post*, *gets* a *session*, s kontrolou typovosti a ochranou proti nebezpečným znakům
- pro úpravu a převody mezi formáty data
- pro práci s časem
- pro odlaďování aplikace a hledání chyb
- některé další

4.3.2.2 Třída jm_module

Tato třída je společná všem třídám typu modul. Je to dceřiná třída třídy *jm_object*. Třída obsahuje některé základní proměnné (pole) obsahující řetězce, přístupová práva, ikony, session, konfiguraci a aktuální cestu. Dále obsahuje metody které zjišťují přístupová práva uživatele k modulu, vkládají předefinované HTML elementy, zobrazují jazykový panel, nápovědu k modulu a mnohé další.

4.3.2.3 Třída jm_loading

Třída jm_loading je dodatečná třída k předchozí třídě. Jak už název napovídá, tato třída zjišťuje načítání informací. Konkrétně se jedná o načítání konfigurace, načítání přístupových práv, zajišťuje autentizaci uživatele, odhlášení uživatele a zabezpečení. Tyto hodnoty jsou používány i ve třídě *jm_module*, ale zde dochází při inicializaci pouze k předání hodnot nikoli načtení.

4.3.2.4 Třída jm_admin_page

Tuto třídu mají za rodičovskou všechny kořenové třídy administrační části prostředí. Třída je potomkem třídy *jm_loading*, neboť v kořenových třídách (*index.php*) je nutné načítání výše uvedených parametrů. Moduly použité v této administrační části jsou zděděné od třídy *jm_module*.

4.3.2.5 Třída jm_application_page

Třída jm_application_page je jakousi obdobou předchozí třídy pro vlastní aplikační část. Třída je také potomkem třídy *jm_loading*.

4.3.3 Třída pro práci s databází

Takovou trochu zvláštní třídou je třída pro práci z databázi, *jm_mysqli*. Třída je odvozena od třídy *mysqli* v knihovně php. Její vznik byl inicializován touhou co nejvíce usnadnit práci s SQL dotazy a pokusit se pracovat s databází pomocí předdefinovaných metod. Nejde ovšem o žádný pokus o objektově orientovanou databázi.

Nachází se zde velmi mnoho metod, proto bude lepší uvést je v tabulce s analogii v SQL.

Metoda	***Popis***	***SQL dotaz***
clear()	resetovaní dotazu	
coll(name)	přidání sloupce	`*name*`
data(name, value)	přidání dat	`*name*` = '*value*'
delete(table1, table2, ...)	mazaní dat	DELETE FROM `*table*`
end_transaction(ok)	konec transakce	COMMIT; ROLLBACK;
insert(table1, table2, ...)	vkládaní dat	INSERT INTO `*table*`
inserted_id()	vrátí id vloženého řádku	
limit(from , to)	úprava počtu výsledku dotazu	LIMIT *from*, *to*
order(col, asc)	určení pořadí	ORDER BY `*col*` *asc*
query(sql_string)	libovolný SQL dotaz	
run(clear)	spuštění dotazu	
set_false(string)	nastavení hlásky při neúspěchu	
set_true(string)	nastavení hlásky při úspěchu	
show_columns(table1,...)	zjištění sloupců tabulky	SHOW COLUMNS `*table*`
start_transaction()	zahájení transakce	START TRANSACTION;
update(table1, table2, ...)	změna dat	UPDATE `*table*` t1, `*table2*` t2
where(string)	podmínka dotazu	WHERE string

Tabulka 4.1: Metody třídy pro práci s databází

Příklad: zobrazení článku v databáze

```
$this->sql->set_false_trstr( "Článek se nepodařilo načíst!" );
$this->sql->select( "db_articles" );
$this->sql->coll("title");
$this->sql->coll("text");
$this->sql->whehe("id='1'");
$this->sql->limit( 0, 1);
$this->sql->run();
Odpovídající SQL dotaz:
SELECT `title`, `text` FROM `db_articles` WHERE `id` = '1' LIMIT 0,1;
```

Na první pohled se může zdát, že je tato konstrukce delší a složitější, než kdyby se použil prostý SQL dotaz s potřebnými knihovními funkcemi. Je to proto, že tento příklad je velmi jednoduchý. Velkou výhodou je snadné použití v cyklech, procházení polí a současné požití metod *coll()* nebo *data()*.

4.3.4 Třída pro práci s řetězci

Jak už bylo zmíněno v kapitole „Modifikace jazykových řetězců“ v aplikaci je možné používat vícero jazyků. K tomu slouží třída *jm_tr_str*. Je ji možné používat ve všech třídách a modulech. V instanci třídy jsou načteny všechny překladatelné řetězce. Z tohoto důvodu je vhodné, aby třída měla během celého vykonávání skriptu právě jednu instanci. Výčet potřebných metod je v tabulce.

Metoda	***Popis***
add_to_defstr(cs, en, group)	přidání nové dvojce slov a určení příslušnosti ke skupině
build(group, key1, key2, key3)	inicializace slov před použitím v kontextu stránky, *key1*=hodnota elementu, *key2*=hodnota titulku elementu, *key3*=hodnota atributu *value* elementu *input*
build_title(group, key)	inicializace titulku stránky v hlavičce prohlížeče
get(group, key, lang)	použití slova(jazykového páru) v kontextu stránky
get_active_language()	zjištění aktivního jazyka
change_active_language(new)	změna aktivního jazyka
printf_js_comands()	funkce vytvoří náležité kód v Java Scriptu nakonci stránky

Tabulka 4.2: Metody třídy pro práci s řetězci

Příklad: zobrazení článku v databáze

```
$key = $this->trstr->build("jadm", "eval", "tval");      // inicializace
echo '<div id="$key" title="';
echo $this->trstr->get("jadm", "tval");          // zobrazeni retezce
echo "' >";
echo $this->trstr->get("jadm", "eval");          // zobrazeni retezce
echo "</div>\n";
printf_js_comands();                             // aktivace Java Scriptu
```

Jelikož se může použití této třídy jevit složitě, uvedu zde zjednodušené schéma, jak tato třída funguje a jak se co odehrává v jednotlivých fázích. Schéma je v příloze E) Schéma funkce třídy tr_str.

4.4 Moduly

Modul je část prostředí, která se stará o nějakou konkretní funkcionalitu nebo entitu. V tomto případě to jsou převážně prvky stránek, ale jsou i jiné moduly. Výdou modulárního přístupu programování je možnost měnit a přidávat moduly, aniž by bylo třeba upravit zbytek aplikace. Je ovšem nutné zachovat komunikační rozhraní modulu s okolím.

Toto rozhraní je jak vstupního, tak výstupního charakteru z pohledu modulu. Na vstupu modul obvykle potřebuje znát konfigurační data, seznam používaných multijazyčných řetězců (třída: *tr_str*), připojení k databázi (třída: *jm_mysqli*), seznam ikon používaných v aplikaci, seznam přístupových práv a aktuální cestu v odkazu. To vše je zajišťováno dědičností, proto každá třída v modulu musí být potomkem třídy *jm_module*. Výstup je definován seznamem veřejných metod jednotlivých tříd.

Změna modulu je v prostředí podporovaná refaktorizace. Přidání nebo smazání modulu není aplikací podporovaná refaktorizace (kapitola: „Podporované změny aplikace“).

4.4.1 Seznam a vlastnosti modulů

V prostředí se nachází mnoho modulů a do budoucna se tento seznam bude stále rozšiřovat. Některé zajímavé moduly přiblížím více v následujících kapitolách. V tabulce je seznam všech modulů, které jsou aktuálně v prostředí, jejich popis a případná vazba na prvek stránky.

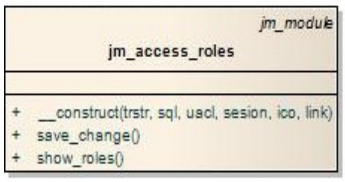
Modul	***Specifikace***	***Vazba na prvek***
m_admin_mess	modul pro práci se zprávami administrátora	Administrátorské zprávy
m_articles	modul pro práci se články a knihami	Článek, Kniha
m_audit	modul pro záznam událostí v prostředí	-
m_basic_info	modul pro úpravu a základních informací	-
m_counter	modul pro práci s počítadlem návštěv	Počítadlo návštěv
m_db_elements	modul pro práci s databází	všechny databázové p.
m_files	modul pro práci se soubory	-
m_forum	modul umožňující diskuse	Fórum
m_galery	modul pro práci s obrázky a fotkami	Obrázek, Galerie
ieml_editor	modul pro editaci šablon a obsahu stránek	-
m_lock	modul pro uzamčení aplikace	-
m_menu	modul pro práci s menu	Menu
m_opinion	modul pro práci s anketami	Anketa
m_search	modul umožňující vyhledávání	Stránka vyhledávání
m_tamplates	modul pracující s webovými šablonami	-
m_uacl	modul přestupových práv	-

Tabulka 4.3: Seznam a vlastnosti modulů

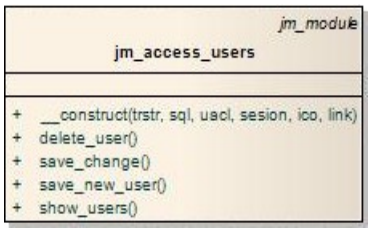
4.4.2 Modul pro řízení přístupu

Modul pro řízení přístupu je jeden z nejdůležitějších modulů vůbec. Stará se o přidělování uživatelských práv a správu účtů a rolí (kapitola: „Přihlašování, správa účtů, řízení přístupu“). Přístupové udaje jsou uloženy v databázových tabulkách *db_acl_roles* a *db_acl_users*. V modulu jsou obsaženy následující třídy:

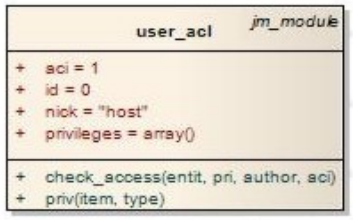
- *jm_access_roles* (správa uživatelských rolí)
- *jm_access_users* (správa uživatelských účtů)
- *jm_uacl_class* (kontrola přístupových práv)



Obrázek 4.9: Třída jm_access_roles (UML)



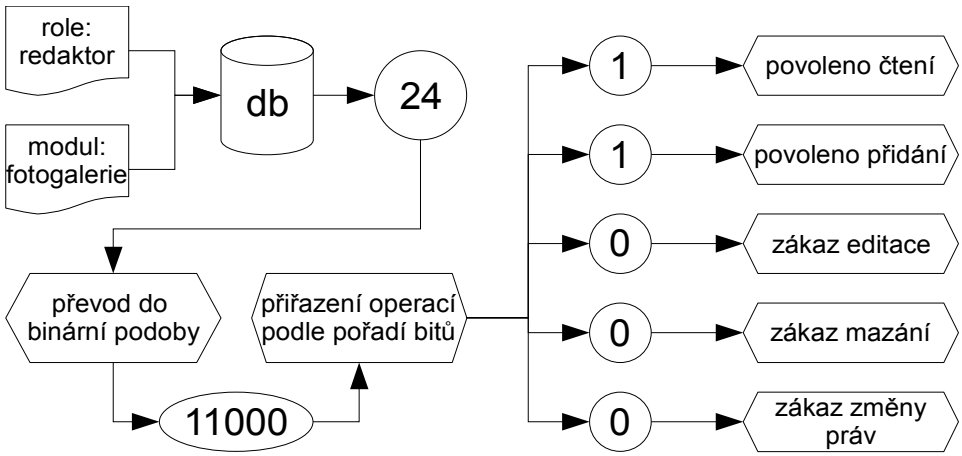
Obrázek 4.11: Třída jm_access_users (UML)



Obrázek 4.10: Třída jm_uacl_class (UML)

Asi nejzajímavější třídou tohoto modulu je třída *jm_uacl_class*. Ta prakticky řídí veškerá přístupová práva v celém prostředí pro tvorbu interaktivních webových stránek. Velkou zásluhu má na tom funkce *check_access()*. Prvním parametrem je jméno modulu, druhým je požadovaná operace (čtení, zápis, přidání, smazaní, změna práv), dále autor a *aci* aktuálního objektu.

V této třídě jsou také tyto přístupové informace uloženy. Ty se načtou z databáze při vytvoření instance třídy. Tato data se musí dále upravit neboť jsou uloženy jako *integer*. Hodnota se převede do binární podoby a rozloží se na jednotlivé bity. Ty určují, které operace může role s tímto modulem provádět. Vše bude lépe vidět na následujícím obrázku.



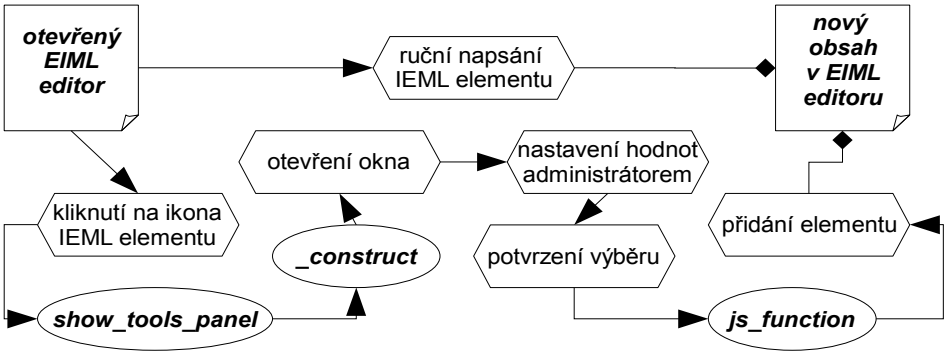
Obrázek 4.12: Načítání a zpracování přístupových dat z databáze

4.4.3 Modul IEML editor

EEML editor je modul, který je používán výhradně administrátorskou částí prostředí. Slouží pro editaci šablony aplikace (kapitola „IEML editor šablona) a pro editaci obsahu stránek (kapitola: IEML a prvky stránky). Tento modul je velmi specifický a neplatí pro něj obecná pravidla jako pro ostatní moduly. Třídy Modul nepracuje s žádnou databázovou tabulkou a obsahuje následující třídy a jeden Java Scriptový soubor:

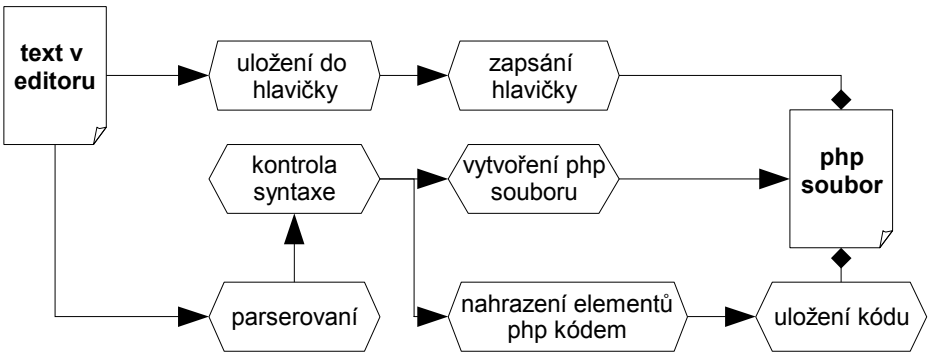
- *ieml_editor* (vlastní IEML editor)
- *ieml_panel* (okno pro vkládání IEML elementů)
- *ieml_parser* (parser upravovaného dokumentu)
- *ieml_tools_window.js* (soubor s Java Scriptem pro *ieml_panel*)

UML diagram tříd je v příloze F) Příloha – schéma tříd modulu *IEML editor* v UML. Základní třídou tohoto modulu je třída *ieml_editor*. Je to jednoduchý HTML editor s možností vkládání IEML elementů (screen shot editoru je v příloze G) Příloha – screen shot IEML editoru). Výsledek stránky je uložen, parserován a znovu uložen jako php skript. Průběh právě těchto dvou událostí blíže vysvětlím na následujících obrázcích. Vysvětlivky k oběma obrázkům jsou v příloze E) úplně dole.



Obrázek 4.13: Přídání elementu v IEML editor

Pří uložení obsahu IEML editoru se vytvoří aktivní php skript. Je nutné ovšem ukládat i původní obsah v IEML, aby bylo možné obsah v budoucnu opět editovat bez nutnosti zpětného parserování. Proto se kód v IEML jazyce ukládá do hlavičky php souboru.

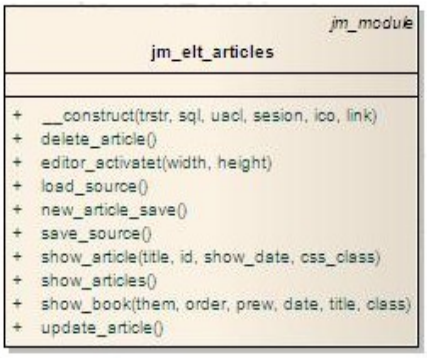


Obrázek 4.14: Uložení stránky v IEML editoru

4.4.4 Modul pro práci s články

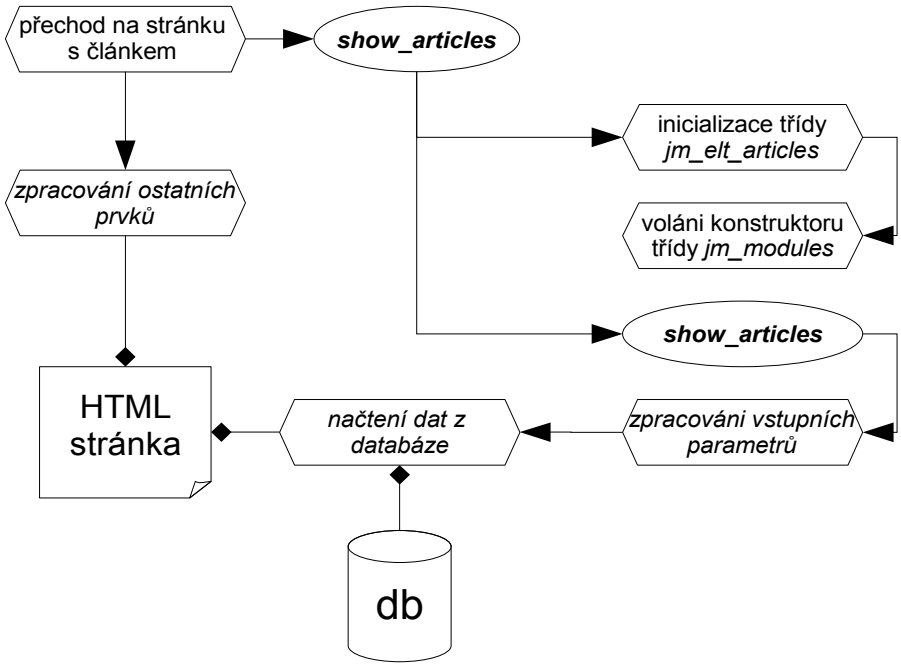
Práce se články je velmi důležitá, pro zde uvedu tento modul. Na modul jsou vázány obsahové prvky stránky: článek a kniha (kapitola „Obsahové prvky stránky). Data jsou ukládána v databázové tabulce *db_article*. Obsažena je pouze následující třída:

- jm_elt_articles



Obrázek 4.15: Třída jm_elt_articles (UML)

Na tomto obrázku je stručně popsán princip přístupu a práce s metodami třídy pro práci se články. Vysvětlivky k obrázku jsou v příloze E) úplně dole.



Obrázek 4.16: Práce se články

5 Závěr

K vytvoření prostředí pro tvorbu webových aplikací mě vedla potřeba vytvářet webové stránky stále rychleji a snadněji. Není ale možné vytvořit prostředí pomocí kterého by se aplikace dala vytvořit pouze uživatelsky (naklikáním možností a vyplněním formulářů), a i kdyby takové robustní prostředí bylo vytvořeno, jeho ovládání by bylo podle mého názoru složitější, než samotné programování. Je proto nutné volit kompromis mezi těmito dvěma extrémy. S využitím objektového přístupu a za použití modulů si myslím, že je toto prostředí velmi povedené a má předpoklady se nadále rozvíjet a rozrůstat.

Toto prostředí v současnosti spravuje několik webových aplikací, od nejjednodušších prezentačních stránek až po internetový obchod (přílohy L - N). Možnost přidávání modulů a podpora webových šablon rozšiřuje hranice použití do takových mezí, které jsou pro dnešní web postačující. Teprve praxe ukázala, které věci se vyplatí provádět automaticky a které je podstatně jednodušší programovat ručně, proto hranice mezi introspekcí, automatickou, podporovanou a nepodporovanou refaktorizací je velmi tenká a dá se vždy lehce posouvat podle aktuálních potřeb.

Myslím že do budoucna budou přidávány stále nové moduly a možná by se dalo uvažovat nad dalšími změnami zásadnějšího rázu. Jednou z nich je použití objektově orientované databáze, což by vlastní návrh velmi zpřehlednilo a usnadnilo další rozšiřování aplikace. Také by bylo možné základní jádro prostředí pracující s webovými prvky stránky neimplementovat v Perlu jako modul pod Apache. Nelze ani vyloučit možnost popisu aplikace ve schématech Relax NG a generovat ji do potřebných programovacích jazyků (jak jsem popisoval již ve své Bakalářské práci).

Závěrem bych chtěl ještě říci pár slov. Programováním a vývojem tohoto prostředí jsem strávil velmi mnoho času. Mnohokrát jsem vešel do slepé uličky a zjistil, že vývoj tím směrem není to pravé, mnohokrát jsem měl pocit, že vyvíjet takto robustní aplikaci o desítkách tisíc řádků je ztrátou času, ale myslím, že nakonec se toto úsilí vyplatilo. Opravdovou radost mám především z toho, že se prostředí v praxi dobře osvědčilo a vložte úsilí se začíná vracet zpět v podobě ušetřených peněz, času a lidských sil.

Literatura

[1] Larry Ullman. *PHP pokročilé programování pro world wide web*, SoftPress 2003, ISBN: 80-86497-36-4

[2] Luke Welling, Laura Thomson. *PHP a MySQL, rozvoj webových aplikací*, SoftPress 2003, ISBN: 80-86497-60-7

[3] W. Jason Gilmore. *Velká kniha PHP 5 a MySQL,* ZonerPress 2005, ISBN: 80-86815-20-2

[4] Jiří Kosek: *PHP – tvorba internetových aplikací*, Grada 1999, ISBN: 80-81326-21-4

[5] *Wikipedia.org* [online]. 2007. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Webov %C3%BD_server>.

[6] Marek Brazina.*Builder.cz* [online]. 2007. Dostupný z WWW: <http://www.builder.cz/serial1.html>.

[7] MySQL AB.*MySQL Reference Manual* [online]. 2007. Dostupný z WWW: <http://dev.mysql.com/doc/refman/5.1/en/>.

[8] Vebloud. *Manualy.net* [online]. 2006. Dostupný z WWW: <http://www.manualy.net/article.php?articleID=17>.

Seznam obrázků

Seznam tabulek

Seznam příloh

A) Příloha – seznam odpovědí serveru

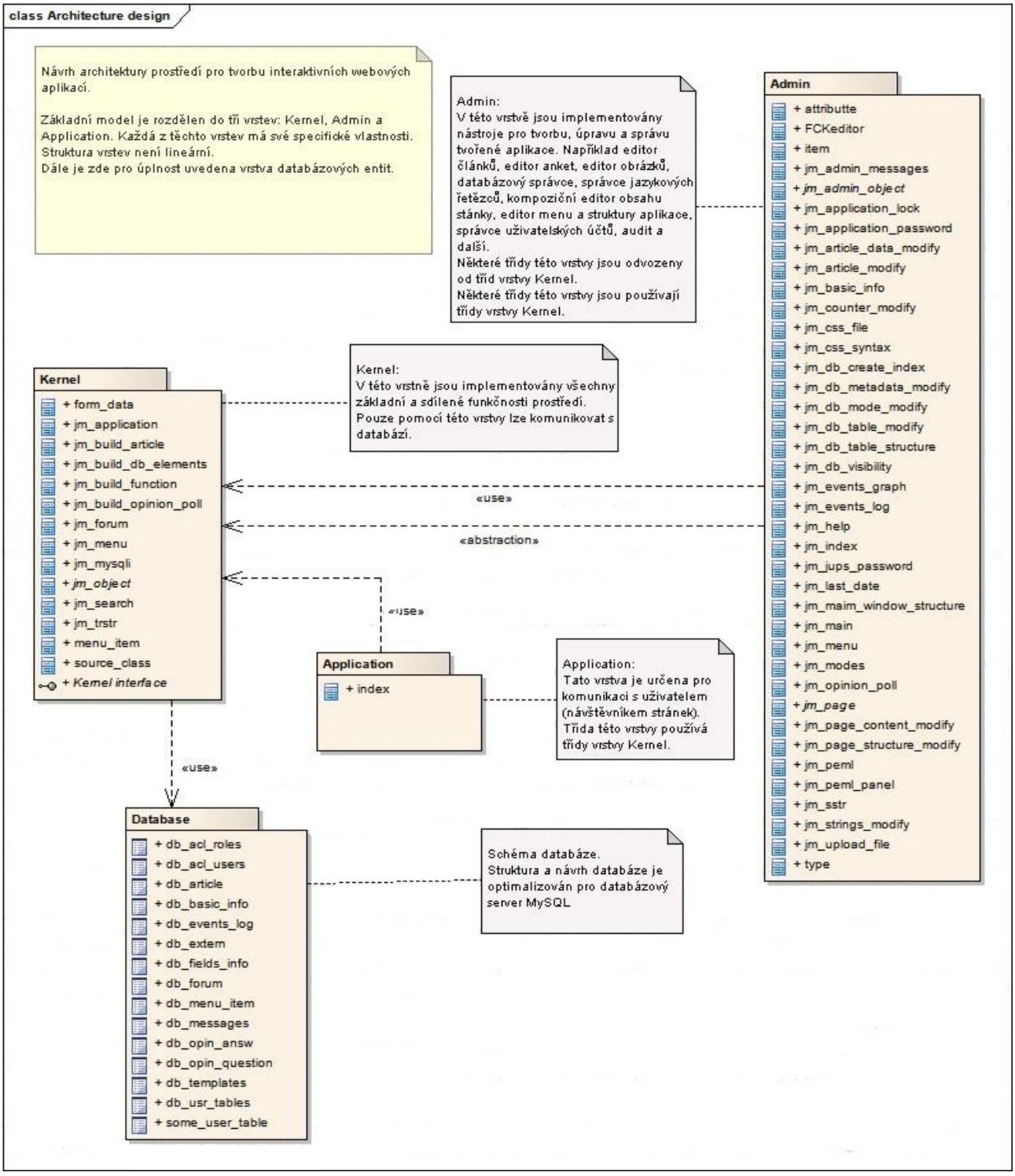
Successful Client Requests
200 OK
201 Created
202 Accepted
203 Non-Authorative Information
204 No Content
205 Reset Content
206 Partial Content
Client Request Redirected
300 Multiple Choices
301 Moved Permanently
302 Moved Temporarily
303 See Other
304 Not Modified
305 Use Proxy
Client Request Errors
400 Bad Request
401 Authorization Required
402 Payment Required (not used yet)
403 Forbidden
404 Not Found
405 Method Not Allowed
406 Not Acceptable (encoding)
407 Proxy Authentication Required
408 Request Timed Out
409 Conflicting Request
410 Gone
411 Content Length Required
412 Precondition Failed
413 Request Entity Too Long
414 Request URI Too Long
415 Unsupported Media Type
Server Errors
500 Internal Server Error
501 Not Implemented
502 Bad Gateway
503 Service Unavailable
504 Gateway Timeout
505 HTTP Version Not Supported

B) Příloha – šablona externího kódu

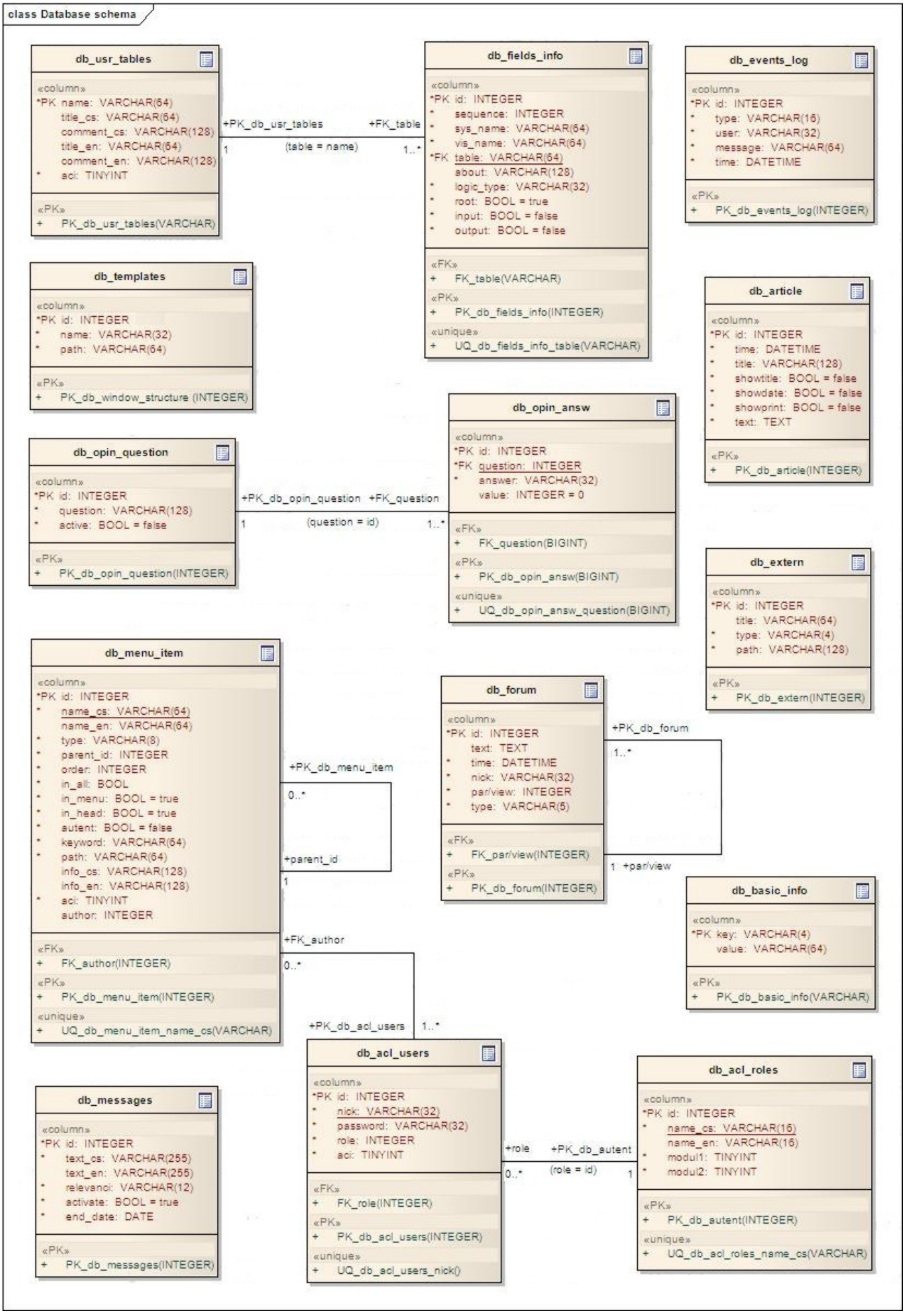
(více informací v kapitole „3.5.10 Programátorský modul“)

```
<?php
/*****************************************************************************

           file: source_class.php
         author: Jaroslav Moravec
          about: sablona
*****************************************************************************/
// hlavni rodicovska trida
include_once( KERNEL_DIR."jm_module_class.inc.php");
/*****************************************************************************
 hlavni trida
*****************************************************************************/
class source_class extends jm_module
{
/*****************************************************************************
 tridni promenne
*****************************************************************************/


/*****************************************************************************
 konstruktory a destruktory
*****************************************************************************/
  function __construct(&$trstr, &$sql, &$uacl, &$sesion, &$ico, &$link )
  {
    // volani rodicovskeho konstruktoru
    parent::__construct($trstr, $sql, $uacl, $sesion, $ico, $link );
    // ziskani promenich z GET
    $this->project_id = $this->from_get( "project", 0, 4);
    $this->page = $this->from_get( "page", 0, 4);
  }
/*****************************************************************************
 metody
*****************************************************************************/
  public function run($par1, $par2)
  {
      /* telo metody */
  }

}

?>
```

C) Příloha – diagram vrstev prostředí

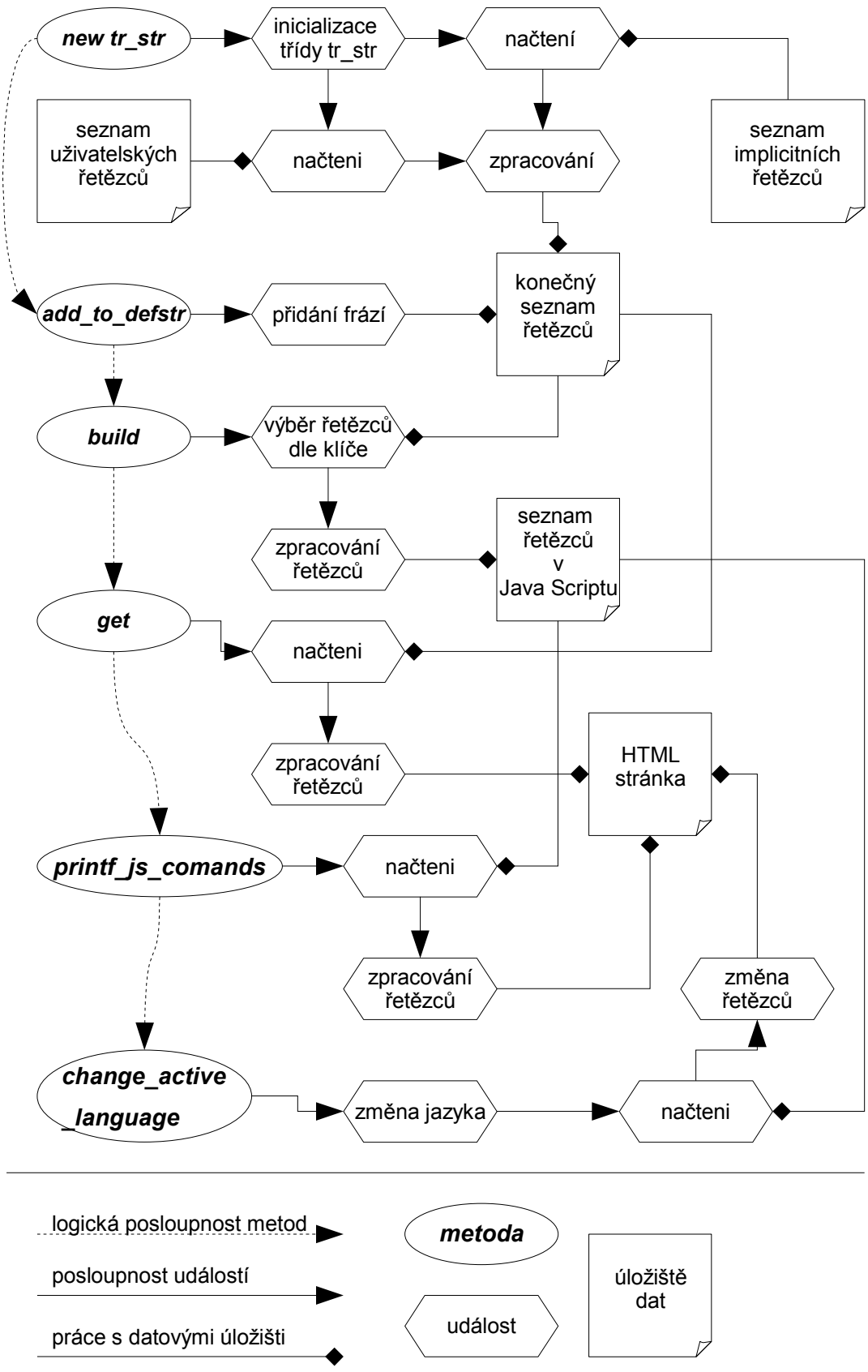


D) Příloha – diagram struktury databáze

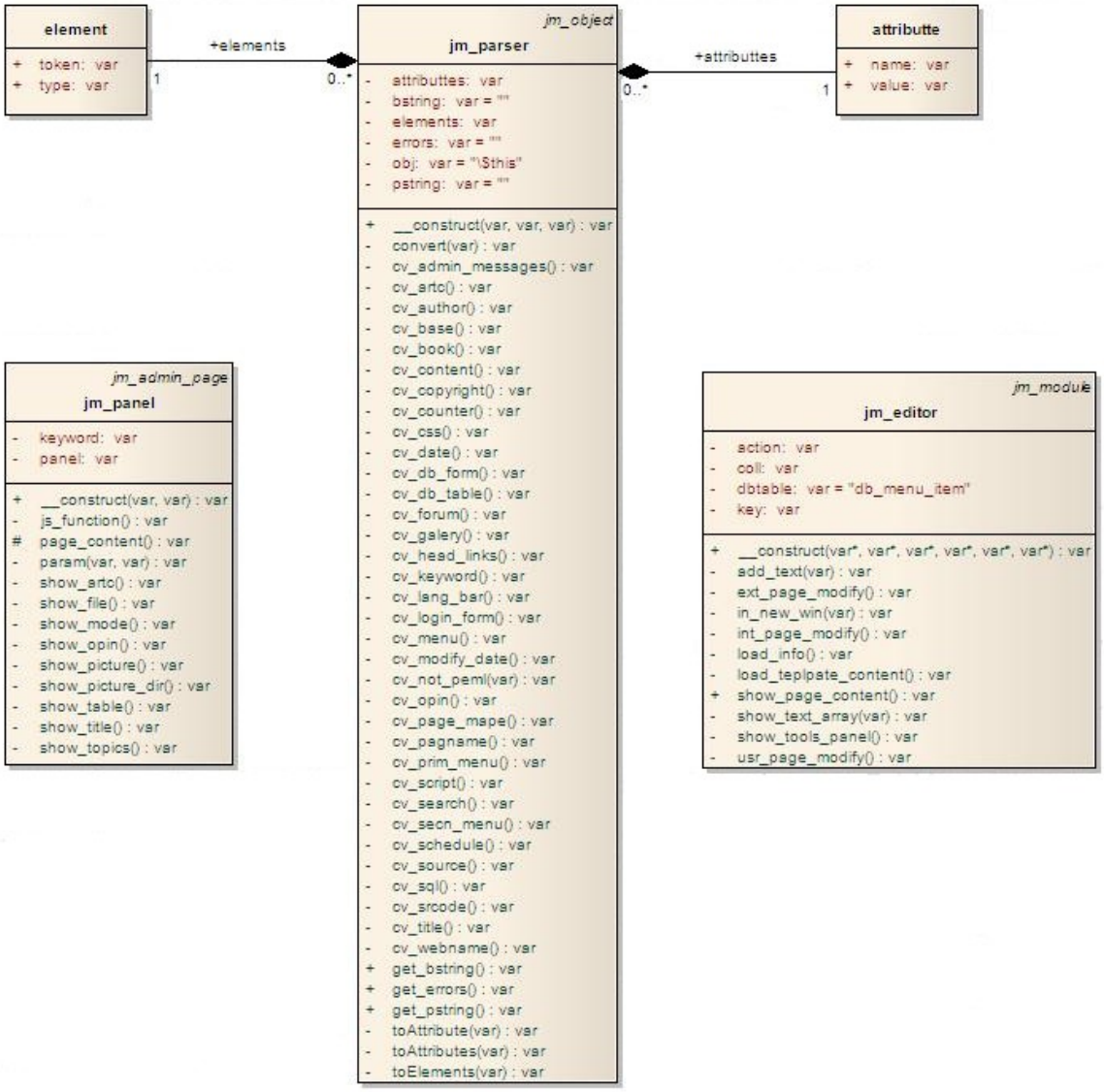


E) Příloha – schéma funkce třídy tr_str

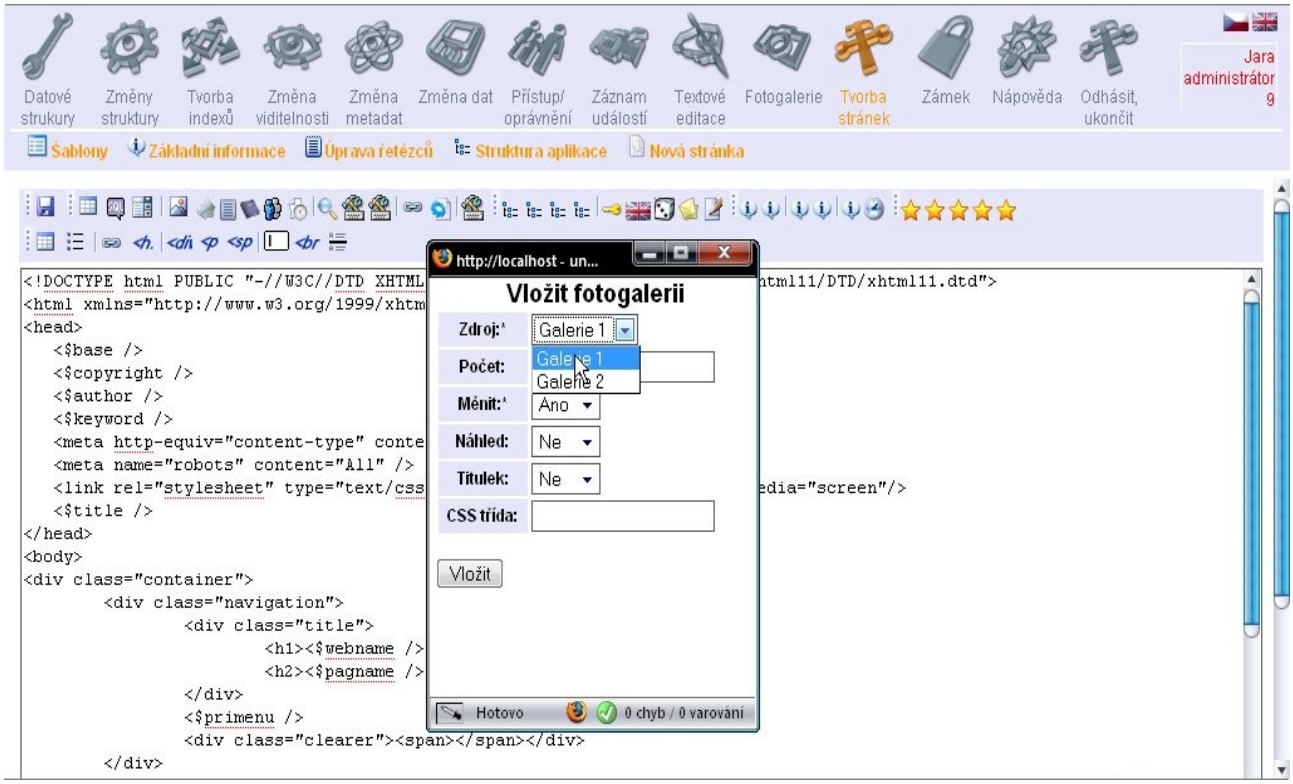
(více informací v kapitole „4.3.4 Třída pro práci s řetězci“)



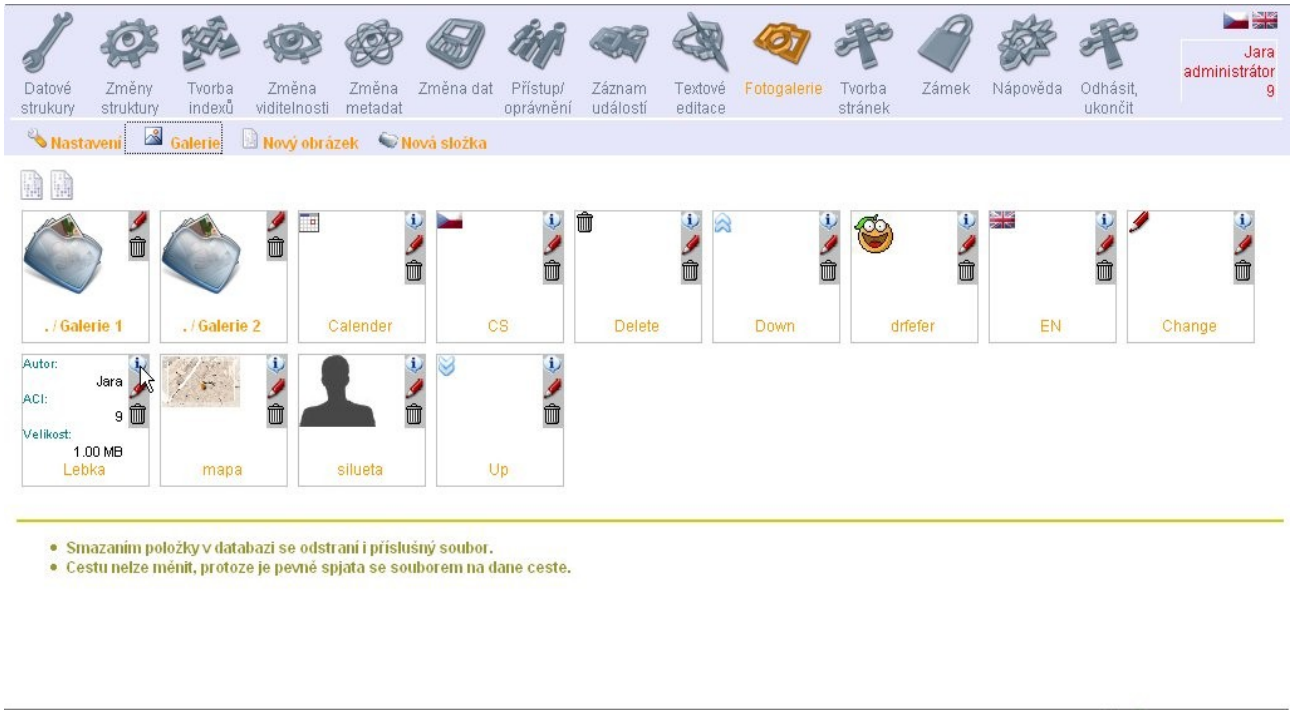
F) Příloha – schéma tříd modulu *IEML editor* v UML



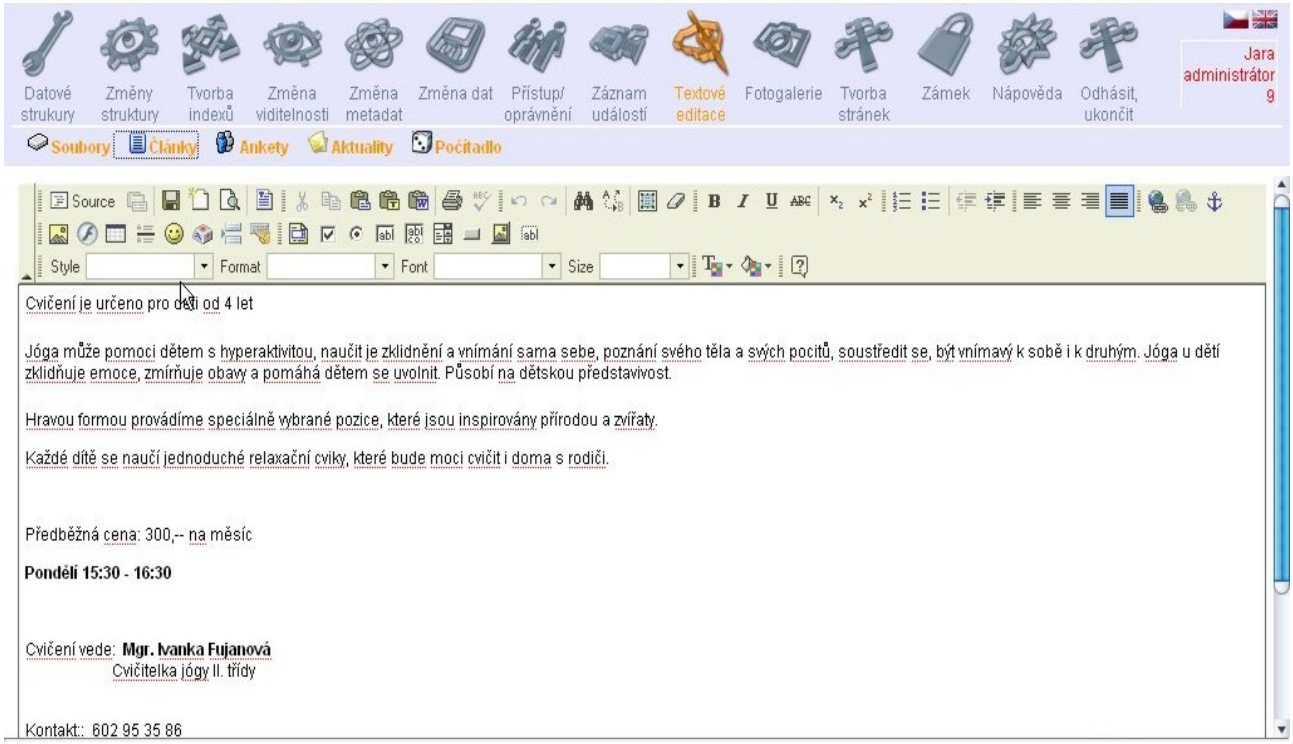
G) Příloha – screen shot IEML editoru



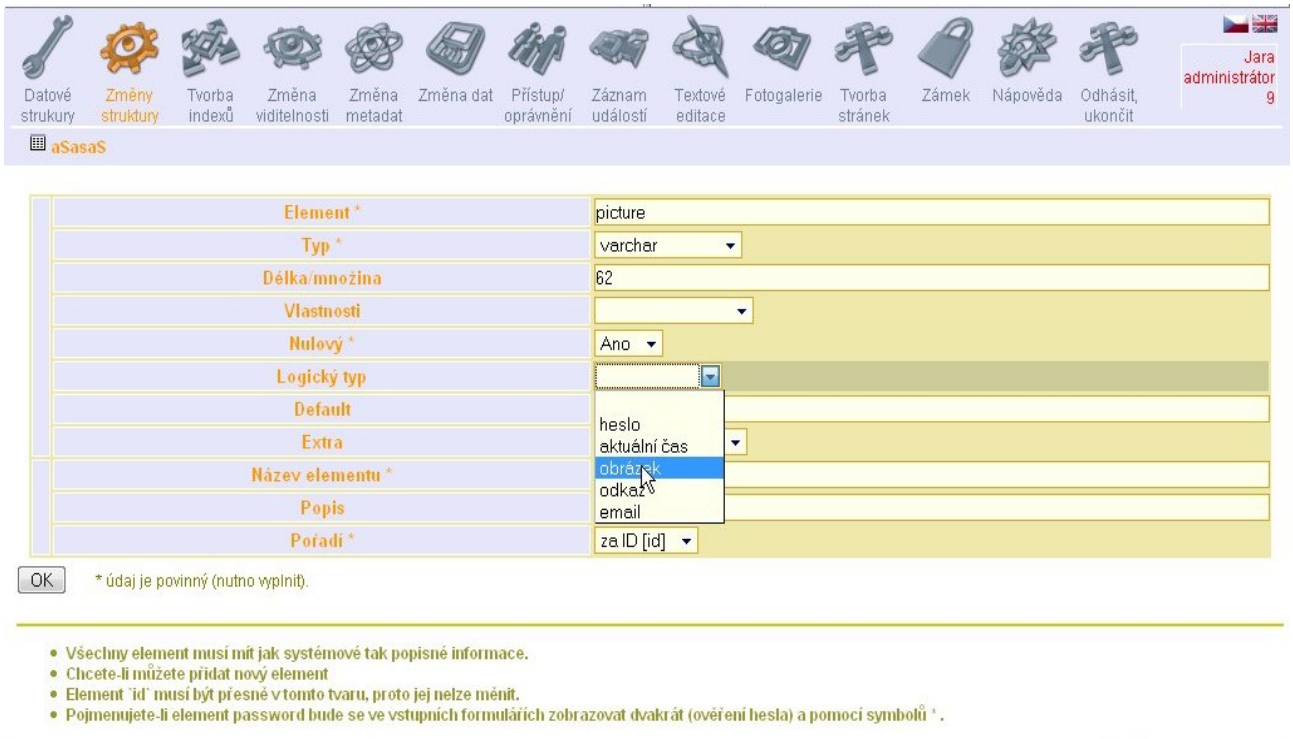
H) Příloha – screen shot fotogalerie



I) Příloha – screen shot editoru článků



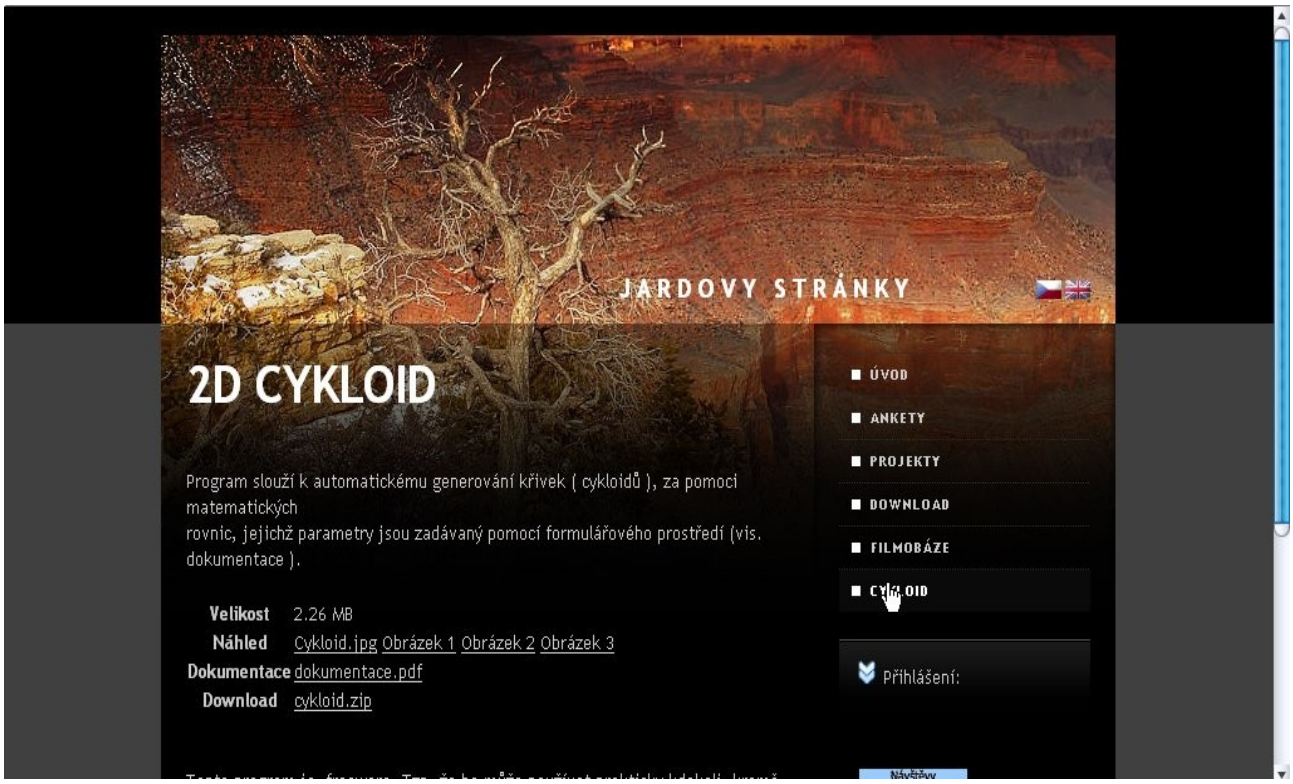
J) Příloha – screen shot přidání sloupce uživatelské tabulky



K) Příloha – screen shot správa uživatelských rolí

L) Příloha – screen shot vytvořená aplikace 1

M) Příloha – screen shot vytvořená aplikace 2



N) Příloha – screen shot vytvořená aplikace 3