Univerzita Tomáše Bati ve Zlíně

Elektronická podpora výuky předmětu AI

Electronical support for education in subject AI

Petr Látal

Bakalářská práce

2009

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2008/2009

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Petr LÁTAL**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Elektornická podpora výuky předmětu AI**

Zásady pro vypracování:

1. **Vypracujte literární rešerši na dané téma.**
2. **Navrhněte a vytvořte dynamickou webovou prezentaci předmětu AI.**
3. **Vytvořte administrační rozhraní včetně jednoduchého uživatelského fóra.**
4. **Vytvořte aplikace pro ověření správnosti výsledků zadaných samostatných prací pomocí SW Web Mathematica nebo JavaScriptu, databázi samostatných prací a rozšíření této databáze o nové úlohy.**
5. **Umístěte webovou prezentaci na server UTB.**

Rozsah práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **ULLMAN, Larry. HTML : PHP a MySQL. Brno : Computer Press, 2004. ISBN 80-251-0063-4.**
2. **HENK C.A : van TILBORG. Fundamentals of cryptology. Norwell : Kluwer Academic, 2000. ISBN 0-7923-8675-2.**
3. **ZELINKA, Ivan. Aplikovaná informatika. Zlín : Ediční středisko, UTB, 1999. ISBN 80-214-1423-5.**
4. **SCHLOSSNAGLE, Georgie. Pokročilé programování v PHP 5. Computer Press, 2004. ISBN 80-86815-14-5.**
5. **LACKO, Luboslav. PHP5 a MYSQL5 Hotová řešení. Computer Press, 2007. ISBN 978-80-251-1695-1.**
6. **KOLEKTIV AUTORŮ. Mistrovství v PHP 5. Computer Press, 2007. ISBN 978-80-251-1519-0.**
7. **CASTRO, Elizabeth. HTML : XHTML a CSS -- Názorný průvodce tvorbou WWW stránek. Computer Press, 2007. ISBN 978-80-251-1531-2.**
8. **RESIG, John. Javascript a AJAX : Moderní programování webových aplikaci. Computer Press,2007. ISBN 978-80-251-1824-5.**

Vedoucí bakalářské práce: **Ing. Roman Šenkeřík, Ph.D.**
Ústav aplikované informatiky

Datum zadání bakalářské práce: **20. února 2009**

Termín odevzdání bakalářské práce: **1. června 2009**

Ve Zlíně dne 13. února 2009

prof. Ing. Vladimír Vašek, CSc.
děkan

L.S.

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Bakalářská práce byla vypracována jako webové řešení programové podpory výuky předmětu Aplikovaná Informatika. V teoretické části jsem se zaměřil na popis webového redakčního systému CMS Made Simple, webMathematicy a programovacích jazyků spojených s těmito systémy. V praktické části pak bylo třeba vyřešit konverzi zdrojových kódů z Mathematicy do formátu JSP pro webMathematicu a jejich začlenění do CMS systému CMS Made Simple.

Klíčová slova:

JSP,CMS, HTML, PHP, MySQL, webMathematica, Mathematica, WYSIWYG, SMARTY, Open Source Software

ABSTRACT

This Graduation Thesis has been created as a web solution of programme support for the education subject Applied Informatics. I concentrated myself on a description of the content management system CMS Made Simple and web services for webMathematica and programming languages connected with these systems in theoretical part of this thesis. Then I had to solve the conversion of source codes from Mathematica to JSP format and their implementation in the practical part.

Keywords:

JSP,CMS, HTML, PHP, MySQL, webMathematica, Mathematica, WYSIWYG, SMARTY, Open Source Software

Rád bych zde poděkoval panu Ing. Romanu Šenkeříkovi, Ph.D. za zadání této bakalářské práce, za téměř okamžitou podporu při řešení nejrůznějších problémů s konfigurací na serveru a také za velmi vstřícný přístup při tvoření bakalářské práce . Dále bych rád poděkoval panu Ing. Jiřímu Hološkovi za potřebnou podporu při konfiguraci některých služeb na serveru a ochotu řešit i zdánlivě neřešitelné problémy při konfiguraci.

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval.

V případě publikace výsledků budu uveden jako spoluautor.

Ve Zlíně …….

podpis diplomanta

OBSAH

ÚVOD

Tato bakalářská práce bude vytvořena pro podporu výuky předmětu AI (Aplikované Informatiky). V tomto předmětu se studenti budou učit programovat šifry (Caesarova šifra, šifra autokláv, transpoziční šifra a další) v programu Wolfram matematica 7. Cílem této práce bude vytvořit webové rozhraní, ve kterém by bylo možné zobrazit postup řešení jednotlivých šifer a také možnost vyzkoušet si funkčnost svých naprogramovaných šifer.

Webové rozhraní bude realizováno na serveru pomocí redakčního systému CMS Made Simple (využívající programovací jazyky HTML,PHP,MySQL) a Wolfram webMathematica (komunikující se zdrojovým kódem ve formátu JSP). Funkce a moduly redakčního systém CMS Made Simple budou podrobně zdokumentovány. Důležitý faktor zde sehraje modul HTML bloky, který zajistí pomocí tagu `iframe` bezproblémový přechod mezi rozhraním JSP na straně webMathematicy a HTML.

Celá práce bude koncipována jako manuál s obrázky pro použití tohoto systému. Uživatel by tedy měl být schopný vkládat obsah společně s obrázky a měl by také být schopný vložit soubory JSP do tohoto systému.

I. TEORETICKÁ ČÁST

1 POUŽITÉ TECHNOLOGIE A SYSTÉMY

Pro realizaci bakalářské práce bylo využito objektového programování v systému pro správu obsahu CMS Made Simple. Ke zpracování šifer bylo potřeba konvertovat soubory Mathematica (*.nb) do formátu JSP, který zpracovává webMathematica.

1.1 CMS

Zkratka CMS pochází z anglického termínu *Content Management System* a označují se tak různé systémy pro správu obsahu. CMS je vlastně složitou webovou aplikací, používanou pro vytváření a úpravy obsahu webu bez znalostí tvorby WWW stránek. Výhodou CMS je možnost rychlé reakce, web si můžete aktualizovat sami kdykoliv, ať je třeba neděle nebo svátek.

Pro internetové použití existuje celá řada nejrůznějších CMS, lišících se podle svých schopností. K předchůdcům CMS patří různá administrační rozhraní pro vkládání novinek a aktualit. Složitější CMS umožňují vkládání různých článků a spotů (včetně tabulek a obrázků) do předem určených kategorií, případně úpravu všech textů na webu. Používají se k tomu tzv. WYSIWYG editory. Podobné CMS bývají také součástí intranetů a extranetů.

Častou funkcí CMS je i správa obrázků a celých fotogalerií. Mnohé umí spravovat i audio a video soubory a další multimediální obsah. Výjimkou není ani možnost spravovat diskuse a komentáře. Takový CMS je pak snadné přebudovat na dokonalý weblog. Ty nejlepší CMS dokáží přímo manipulovat se strukturou webu.

Okolo obecné terminologie systémů pro správu obsahu dnes existuje spousta nepřesností. Kromě CMS se rozlišují i další systémy – nejčastěji WCM (zaměřené pouze na správu webu) a DMS (správa dokumentů). Jako synonyma k CMS se používají i termíny **redakční systém** a **publikační systém**.[1]

Mezi základní funkce CMS systému patří například:

- Tvorba, modifikace a publikace dokumentů (článků) zpravidla prostřednictvím webového rozhraní, často s využitím jednoduchého online WYSIWYG editoru nebo jednoduchého systému formátování textu
- správa diskusí či komentářů
- správa souborů
- správa obrázků či galerií
- kalendářní funkce
- statistika přístupů
- správa anket
- správa registrovaných uživatelů

1.1.1 Výhody CMS

Velkou výhodou je, že osoba obstarávající vkládání a úpravy obsahu nemusí znát jazyk HTML.

Většina CMS systémů je Open Source Software (OSS), což znamená, že zdrojové kódy jsou otevřené. Otevřenost zde znamená jak technickou dostupnost kódu, tak legální dostupnost - licenci software, která umožňuje, při dodržení jistých podmínek, uživatelům zdrojový kód využívat, například prohlížet a upravovat. V nepřesném, ale poměrně běžném vyjadřování, se označení open source používá i pro mnoho vlastností, které s otevřeností zdrojového kódu nesouvisí, ale vyskytují se u mnoha open source programů. Například může jít o bezplatnou dostupnost software, vývoj zajišťovaný úplně nebo z podstatné části dobrovolnickou komunitou nebo „nekomerčnost“. [2]

1.1.2 CMS Made Simple

CMS Made Simple odděluje obsah od webového návrhu nebo struktury. Jednoduše řečeno, obsah je oddělen od programové části systému, což dovoluje návrháři zaměřovat se na design a editorovi na stránky s obsahem. V administrační části je vytvořena správa, která umožňuje uživatelům konat úkony na webu pomocí omezujících oprávnění. Administrátor může doinstalovat další rozšíření, která dodávají rozšířenou funkcionalitu na stránkách. Typickým případem může být například Frontend User Management, který umožňuje registrovat nové uživatele na webu, zasílat zapomenutá hesla, doplňovat či měnit personální informace a spoustu dalších věcí.

1.2 HTML

HyperText Markup Language, označovaný zkratkou **HTML**, je značkovací jazyk pro hypertext. Je jedním z jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci dokumentů na Internetu.

Jazyk je aplikací dříve vyvinutého rozsáhlého univerzálního značkovacího jazyka SGML (*Standard Generalized Markup Language*). Vývoj HTML byl ovlivněn vývojem webových prohlížečů, které zpětně ovlivňovaly definici jazyka.

1.2.1 Vývoj jazyka

V roce 1989 spolupracovali Tim Berners-Lee a Robert Caillau na propojeném informačním systému pro CERN, výzkumné centrum fyziky poblíž Ženevy ve Švýcarsku. V té době se pro tvorbu dokumentů obvykle používaly jazyky TeX, PostScript a také SGML. Berners-Lee si uvědomoval, že potřebují něco jednoduššího a v roce 1990 byl tedy navržen jazyk HTML a protokol pro jeho přenos v počítačové síti – HTTP (*HyperText Transfer Protocol – přenosový protokol hypertextu*). Zároveň také Tim Berners-Lee napsal první webový prohlížeč, který nazval WorldWideWeb.

V roce 1991 CERN zprovoznil svůj web. Současně organizace NCSA (*National Center for Supercomputer Applications*) vybídla Marca Andreessena a Erica Binu k vytvoření prohlížeče Mosaic; ten vznikl v roce 1993 ve verzích pro počítače IBM PC a Macintosh a měl obrovský úspěch. Byl to první prohlížeč s grafickým uživatelským rozhraním.

Následoval rychlý rozvoj webu, takže bylo nutné pro HTML definovat standardy.

1.2.2 Verze jazyka

Verze 0.9

Byla vydána zhruba v roce 1991. Nepodporuje grafický režim (verze, kterou vytvořil Tim Berners-Lee).

Verze 2.0

Zachycuje stav jazyka v polovině roku 1994. Standard vydala komunita IETF (*Internet Engineering Task Force*). Je to první verze, která odpovídá syntaxi SGML. Přidává k původní specifikaci interaktivní formuláře a podporu grafiky.

Verze 3.2

Byla vydána 14. ledna 1997 a zachycuje stav jazyka v roce 1996. Připravovaná verze HTML 3.0 nebyla nikdy přijata jako standard, protože byla příliš složitá a žádná firma nebyla schopna naprogramovat její podporu. Standard už vydalo W3C, stejně jako následující verze. Přidává k jazyku tabulky, zarovnávání textu a stylové elementy pro ovlivňování vzhledu.

Verze 4.0

Byla vydána 18. prosince 1997. Do specifikace jazyka přibyly nové prvky pro tvorbu tabulek, formulářů a nově byly standardizovány rámy (*frames*). Tato verze se snaží dosáhnout původního účelu – prvky by měly vyznačovat význam (sémantiku) jednotlivých částí dokumentu, vzhled má být ovlivňován připojovanými styly. Některé prezentační elementy byly zavrženy.

Verze 4.01

Byla vydána 24. prosince 1999. Tato verze opravuje některé chyby verze předchozí.[1] Podle původního předpokladu se mělo jednat o poslední verzi, po které by se přešlo na XHTML.

Verze 5

7. března 2007 byla založena nová pracovní skupina HTML, jejíž cílem je vývoj nové verze HTML. V květnu 2007 bylo odhlasováno, že základem nové specifikace se

stanou Web Applications 1.0 a Web Forms 2.0 ze specifikace WHATWG. Jako název nové specifikace bylo odhlasováno HTML 5. Specifikace by měla být hotova v letech 2010-2012 (odkdy ji začnou vývojáři webových aplikací používat), ukončení vývoje specifikace po vyřešení problémů a opravení všech chyb se odhaduje až na rok 2022).

1.2.3 Popis jazyka

1.2.3.1 Koncepce

Jazyk HTML je od verze 2.0 aplikací SGML (připravovaná verze HTML5 ale již závislost na SGML obsahovat nebude). Je charakterizován množinou značek a jejich atributů definovaných pro danou verzi. Mezi značky se uzavírají části textu dokumentu a tím se určuje význam (*sémantika*) obsaženého textu. Názvy jednotlivých značek se uzavírají mezi úhlové závorky (< a >). Část dokumentu tvořená otevírací značkou, nějakým obsahem a odpovídající ukončovací značkou tvoří tzv. *element* (prvek) dokumentu. Například `<strong>` je otevírací značka pro zvýraznění textu a `<strong>červená Karkulka</strong>` je element obsahující zvýrazněný text. Součástí obsahu elementu mohou být další vnořené elementy. Atributy jsou doplňující informace, které upřesňují vlastnosti elementu.

Značky (zvané tagy) jsou obvykle párové (v XHTML jsou párové všechny), přičemž koncové značka je shodná se značkou počáteční, jen má před názvem znak lomítko. Příklad pro označení odstavce:

```
<p>Text odstavce</p>
```

Některé značky jsou nepárové – nemají žádný obsah a nepoužívají koncovou značku. Příklad pro vykreslení vodorovné čáry:

```
<hr>
```

Tagy mohou obsahovat atributy, které popisují jejich vlastnosti nebo nesou jinou informaci. Příkladem může být odkaz (tag `a`), jehož atribut `href` říká, kam se uživatel po kliknutí na něj dostane (v tomto příkladu na stránku `http://example.com`):

```
<a href="http://example.com">text odkazu</a>
```

Nebo jiné možnosti zápisu odkazu - odkaz, který se otevře v novém okně/panelu:

```
<a href="http://example.com" target="_blank">text odkazu</a>
```

Nebo jiné možnosti zápisu odkazu - odkaz, který se otevře v novém okně/panelu, obsahuje titulek (zobrazí se po najetí myší) a bude přirazen ke třídě CSS (CSS - vyložení jazyka dle WIKI):

```
<a href="http://example.com" target="_blank" title="titulek"
class="nazev_tridy">text odkazu</a>
```

Od SGML zdědil jazyk HTML i jiné, méně známé konstrukce pro tvoření elementů. Jedná se o tzv. *zkrácené HTML zápisy*:

- `<element/Obsah/` je totéž jako `<element>Obsah</element>`
- `<element>Obsah</>` je totéž jako `<element>Obsah</element>`
- `<el<el2>Obsah` je totéž jako `<el><el2>Obsah`
- `<ul><li>Adam<>Božena</ul>` je totéž jako `<ul><li>Adam</li><li>Božena</li></ul>`

Všechny tyto zápisy jsou sice podle normy validní a zcela ekvivalentní, ale žádný ze známých prohlížečů zkrácené verze nepodporuje, takže se nedoporučuje je používat.

Pro každou verzi existuje definice pravidel DTD (*Document Type Definition*). Od verze 4.01 musí být odkaz na deklaraci DTD v dokumentu uveden pomocí klíčového slova `DOCTYPE`. DTD definuje pro určitou verzi elementy a atributy, které lze používat.

Dokument může mimo značkování obsahovat další prvky:

- Direktivy – začínají znaky `<!`, jsou určeny pro zpracovatele dokumentu (prohlížeč).
- Komentáře – pomocné texty pro programátora, nejsou součástí obsahu dokumentu a nezobrazují se (prohlížeč je ignoruje). Příklad komentáře je uveden níže.
- Kód skriptovacích jazyků.
- Definice událostí a kód pro jejich obsluhu.

1.2.3.2 Struktura dokumentu

Dokument v jazyku HTML má předepsanou strukturu:

- Deklarace DTD – je povinná až ve verzi 4.01, je uvedena direktivou `<!DOCTYPE`.

- Kořenový element – element `html` (značky `<html>` a `</html>`) reprezentuje celý dokument. Kořenový element je povinný, ale otevírací a ukončovací značka samotná povinná není (pokud tyto značky nebudou v těle dokumentu uvedeny, prohlížeč si je sám doplní podle kontextu).

- Hlavička elementu – obsahuje metadata, která se vztahují k celému dokumentu. Definují např. název dokumentu, jazyk, kódování, klíčová slova, popis, použitý styl zobrazení. Hlavička je uzavřena mezi značky `<head>` a `</head>`. Element `head` je opět povinný, ale jeho otevírací a koncová značka povinná není, prohlížeč ji sám doplní podle kontextu.

- Tělo dokumentu – obsahuje vlastní text dokumentu. Vymezuje se značkami `<body>` a `</body>`. Element `body` je povinný, ale jeho otevírací a koncová značka povinná není, prohlížeč ji sám doplní podle kontextu.

1.2.3.3 Příklad zdrojového kódu

Příklad HTML dokumentu ve verzi 4.01:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
   <!-- toto je komentář -->
   <head>
      <title>Titulek stránky</title>
   </head>
   <!-- tělo dokumentu -->
   <body>
      <h1>Nadpis stránky</h1>
      <p>Toto je tělo dokumentu</p>
   </body>
</html>
```

1.2.3.4 Druhy značek

Značky lze z hlediska významu rozdělit na tři základní skupiny:

Strukturální značky

rozvrhují strukturu dokumentu, příkladem jsou odstavce (`<p>`), nadpisy (`<h1>`, `<h2>`). Dodávají dokumentu formu.

Popisné (sémantické) značky

popisují povahu obsahu elementu, příkladem je nadpis (`<title>`) nebo adresa (`<address>`). Současný trend je orientován právě na sémantické značky, které usnadňují automatizované zpracovávání dokumentů a vyhledávání informací v záplavě dokumentů na webu. Vyvrcholením této snahy je v současné době jazyk XML.

Stylistické značky

určují vzhled elementu při zobrazení, typickým příkladem je značka pro tučné písmo (`<b>`). Od tohoto druhu značek se postupně upouští, trendem je používání kaskádových stylů, které vzhled popisují odděleně od obsahu dokumentu. Mezi důvody pro neužívání těchto značek patří především to, že tyto značky jsou orientovány na prohlížení na obrazovce počítače, příliš se však nepočítá s používáním dokumentu jiným způsobem – alternativní prohlížeče pro postižené (čtečky pro slepce), v mobilních zařízeních a podobně. Kaskádové styly umožňují definovat rozdílné zobrazení pro různá zařízení.

1.2.4 Parsování v prohlížečích

Webové prohlížeče jsou programy, jejichž účelem je prezentovat dokument na zobrazovacím zařízení – nejčastěji monitoru počítače. Dokument je prohlížečem načítán a prováděna jeho rozklad (syntaktická analýza) na jednotlivé elementy. Prohlížeč obsahuje tabulku značek, které podporuje. Moderní prohlížeče (Opera, Mozilla Firefox) dokonce umožňují „vytvářet“ vlastní tagy a elementy a umožňují také jejich stylování pomocí kaskádových stylů. Jelikož však nejrozšířenější prohlížeč, Windows Internet Explorer, toto nedovoluje a obsah neznámých elementů zobrazí zcela normálně, bez stylu, vlastní elementy se prakticky nepoužívají.

Každému elementu je poté přiřazen *styl* (způsob zobrazení). Styly mohou být uvedeny ve stylovém předpisu. Vlastnosti stylů, které nejsou předepsány, doplní prohlížeč podle implicitního stylu, který má zabudován. Některé prohlížeče umožňují uživateli implicitní styly definovat.

Novější prohlížeče pracují obecně ve dvou základních režimech:

- *Standardní režim* – režim snažící se dodržovat definované standardy;
- *Quirk mód* – režim zaměřený na zpětnou kompatibilitu, i pokud takové chování není v souladu se standardy.

Tyto režimy chování zavedl Internet Explorer ve své páté verzi z důvodu zpětné kompatibility. Microsoft při vytváření nové verze prohlížeče chtěl, aby se v něm zobrazovaly správně stránky, které již existují, ale také nové stránky, které jsou psány podle standardů. To, jaký režim (a chování) prohlížeč zvolí, závisí především na uvedení direktivy `<!DOCTYPE`, protože většina starších stránek ji vůbec neobsahuje. Později toto chování částečně přebrala Mozilla i Opera, kde však rozdíly mezi režimy nejsou natolik markantní. Existence nestandardního režimu je výsledkem vývoje, kdy si výrobci jednotlivých prohlížečů přizpůsobovali definici HTML podle svých potřeb a prohlížeče podporovaly nestandardní elementy a syntaxi. Řada těchto „vylepšení“ byla následně přejímána do standardů, některé však byly zase v dalších verzích vyřazeny.

1.2.5 Alternativní zařízení

Vzhledem k rozvoji jak na poli software, tak i hardware přibylo mnoho nových možností zpracování HTML dokumentů. Proto se pojem „prohlížeč“ stal zavádějícím (jsou např. i hlasové čtečky, či tiskárny) a W3C začalo používat termín *user agent*. V tuto chvíli (konec roku 2007) ještě neexistuje jednotný český termín, používá se buď doslovný překlad – „uživatelský agent“, nebo např. „interpret“, „interpretr“, „zařízení“ či jiné.

1.2.6 Editory HTML

Editory HTML jsou programy pro snadnou tvorbu webových stránek. Například PSPad, Macromedia Dreamweaver a další.

1.2.6.1 Textové editory

Editorem HTML může být ve své podstatě jakýkoliv program pracující s textem. V praxi se však používají mnohem sofistikovanější programy. Běžný editor HTML zvládá barevnou syntaxi (barevně rozlišuje jednotlivé části kódu jako například HTML značky či atributy a

prostý text), dokáže napovídat značky, zná chytré tabulátory nebo zvládá validovat dokument podle předepsané DTD. Mezi takové nejrozšířenější editory HTML u nás určitě patří PSPad.

1.2.6.2 WYSIWYG editory

WYSIWYG je zkratka od anglického „What you see is what you get“, v překladu „Co vidíš, to dostaneš“. Tyto WYSIWYG editory pracují na opačném principu než textové editory – ve WYSIWYG editoru pracujete z již hotovou stránkou a obecně neplatí, že by uživatel tohoto editoru musel znát jazyk HTML. Ve WYSIWYG editoru si může uživatel poskládat stránku jak se mu zlíbí a program následně vygeneruje požadovaný kód HTML. Mezi nejznámější takovéto editory patří *Adobe Dreamweaver* nebo *Expression Web* (novější verze *Microsoft Frontpage*). [3]

1.3 PHP

PHP (rekurzivní zkratka *PHP: Hypertext Preprocessor*, „PHP: Hypertextový preprocesor“, původně *Personal Home Page*) je skriptovací programovací jazyk, určený především pro programování dynamických internetových stránek. Nejčastěji se začleňuje přímo do struktury jazyka HTML, XHTML či WML, což lze využít při tvorbě webových aplikací. PHP lze použít i k tvorbě konzolových a desktopových aplikací.

PHP skripty jsou většinou prováděny na straně serveru, k uživateli je přenášen až výsledek jejich činnosti (interpret PHP skriptu je možné volat pomocí příkazové řádky). Syntaxe jazyka je inspirována několika programovacími jazyky (Perl, C, Pascal a Java). PHP je nezávislý na platformě, skripty fungují bez větších úprav na mnoha různých operačních systémech. Podporuje mnoho knihoven pro různé účely - např. zpracování textu, grafiky, práci se soubory, přístup k většině databázových systémů (mj. MySQL, ODBC, Oracle, PostgreSQL, MSSQL), podporu celé řady internetových protokolů (HTTP, SMTP, SNMP, FTP, IMAP, POP3, LDAP…)

PHP se stalo velmi oblíbeným především díky jednoduchosti použití a tomu, že kombinuje vlastnosti více programovacích jazyků a nechává tak vývojáři částečnou svobodu v syntaxi. V kombinaci s operačním systémem Linux, databázovým systémem (obvykle MySQL nebo PostgreSQL) a webovým serverem Apache je často využíván k tvorbě webových aplikací.

Pro tuto kombinaci se vžila zkratka LAMP – tedy spojení Linux, Apache, MySQL a PHP nebo Perl.

Pomocí technologie PHP je naprogramovaná Wikipedie.

1.3.1 Ukázka kódu

Takto v PHP vypadá skript Hello world:

```
<?php
        echo "Ahoj, světe!";
?>
```

1.3.2 Některé vlastnosti jazyka PHP

- Jazyk PHP je dynamicky typový, tzn. že datový typ proměnné se určí v okamžiku přiřazení hodnoty.
- Kvůli tomu má PHP dva operátory porovnání - '==' a '==='; při použití prvního dochází před porovnáním ke konverzi, při použití druhého je výraz pravdivý jen když jsou oba dva operandy stejného datového typu a jejich obsah má stejnou hodnotu.
- Pole se dají indexovat číselnými indexy (jako v jazyce C), nebo mohou fungovat jako hash-mapa. Stejné pole může obsahovat oba typy indexů.
- Pole jsou heterogenní (stejné pole může obsahovat prvky různých typů) a řídká.
- Řetězce lze uzavírat do uvozovek (při vyhodnocení se provede nahrazení proměnných uvnitř) nebo do apostrofů (nahrazuje se jen escape sekvence \').

```
// Zde je v proměnné string (tečka je operátor spojování řetězců)
$retez = "Ahoj, světe" . ', mám se dobře' . " a nevadí, že střídám oddělovače";

// Zde je v proměnné číslo (int)
$cislo = 100;

// Do proměnné je možné dát pole, které obsahuje jak čísla, tak znaky či další pole
$pole = array('a', 'b', 1, 2, array('první' => 'podpole', 'vytištěno'));

// Nenahlásí chybu (jenom varování) a vytiskne 'Array'
```

```
print($pole);

// Vytiskne obsah proměnné pole
print_r($pole);

// Test porovnání
$cislo = 100;
$retez = '100';

// Toto porovnání ('==') platí díky automatické typové konverzi
if ($retez == $cislo)
{
  echo 'Jsou stejné';
}

// Ale porovnání pomocí '===' neplatí, neboť nejsou stejné typy
if ($retez === $cislo)
{
  echo 'To by nešlo';
}
```

- PHP do verze 4.2.0 ve výchozím nastavení automaticky přejímalo veškeré proměnné poslané jakoukoliv metodou (HTTP POST, HTTP GET, HTTP cookie, ale i ze zabudovaného mechanismu sessions) a umožňovalo s nimi dále pracovat jako s globálními - tato možnost představovala bezpečnostní riziko.

- Od verze 4.2.0 lze hodnotu získat z tzv. superglobálních proměnných s garancí původu informace - tedy že data byla odeslána požadovanou metodou. Používání globálních proměnných je stále možné pomocí konfigurační direktivy `register_globals` povolit, ale z bezpečnostních důvodů je to silně nedoporučováno.

```
// odešlu formulář metodou POST, kde do pole s názvem jmeno vepíšu 'Tom'
echo $jmeno; // vrátí 'Tom', funguje pouze v případě povolených globálních proměnných
echo $_POST['jmeno']; // vrátí 'Tom', superglobální proměnné fungují i při vypnutých globálních proměnných
echo $_GET['jmeno'];  // vypíše se chybové hlášení o neexistenci proměnné a vrátí NULL
// NULL je zvláštní hodnota libovolného typu proměnných pro stav 'nedefinováno'
```

1.3.3 Významné projekty implementované v PHP

- software MediaWiki — např. Wikipedie, tato webová encyklopedie
- phpBB — balík pro provoz webového fóra
- WordPress — publikační systém pro provoz blogů a podobných aplikací
- phpMyAdmin —webová aplikace pro správu databázového systému MySQL
- Texy! — překladač intuitivní syntaxe pro formátování textu na HTML

1.3.4 Výhody a nevýhody PHP

1.3.4.1 Výhody PHP

- PHP je specializované na webové stránky
- rozsáhlý soubor funkcí v základní knihovně PHP + dalších z PECL
- nativní podpora mnoha databázových systémů
- multiplatformost (zejména Linux, Windows)
- možnost využití nativních funkcí operačního systému (možná nekompatibilita s jiným OS)
- strmá křivka učení
- obrovská podpora na hostingových službách – PHP je fakticky standardem, který najdeme všude
- obrovské množství projektů a kódů, které lze zdarma využít (WordPress, phpBB a další)
- poměrně slušná dokumentace
- vynikající, velmi svobodná licence – svobodnější, než GPL

Nevýhody PHP

- jazyk PHP není nikde definován, je popsán pouze jeho implementací
- mírně nekonzistentní vývoj v minulosti, který si sebou PHP nese dosud (někdy často měnící se příkazy atd…)
- nekonzistentní pojmenování funkcí a nejednotné pořadí parametrů
- ač jazyk výborně podporuje výjimky, jeho knihovna je používá jen zřídka
- slabší podpora Unicode, pouze přes PHP knihovnu (v PHP 6 má být Unicode řetězec jako základní typ)
- neumožňuje překlad do byte kódu, PHP skript se při každém požadavku překládá znovu
- ve standardní distribuci chybí ladící (debugovací) nástroj
- po zpracování požadavku neudržuje kontext aplikace, vytváří jej vždy znovu (oslabuje výkon)
- nepodporuje jmenné prostory (v nejbližší verzi 5.3 v PHP budou)

1.3.5 Historický vývoj PHP

PHP bylo původně označení pro Personal Home Page, tedy osobní domácí stránky. Vše začalo v roce 1994, kdy byla napsána binární část Common Gateway Interface (CGI) v programovacím jazyku C. Tuto prvotní část napsal Dánský/Grónský Programátor Rasmus Lerdorf. Lerdorf zpočátku vytvořil tyto nástroje pro osobní domácí stránky (Personal Home Page) za účelem možné záměny s malou skupinou skriptů v Perlu, které chtěl používat pro údržbu osobní domovské stránky. Nástroje měly zajistit běh úloh jako například zobrazení résumé (obsahu) a zaznamenávání návštěvnosti jeho stránek. Tento binární kód ještě tentýž rok skloubil s jiným programem, který sám napsal. Po spojení s Form Interpreter tak vznikla kombinace PHP/FI, která měla mnohem větší funkčnost. PHP/FI obsahovala širokou implementaci pro programovací jazyk C a navíc tato verze mohla komunikovat s databázemi, což umožnilo tvorbu prvních jednoduchých dynamických webových aplikací. Lerdorf veřej-

ně vydal PHP 8. června 1995, aby mohl najít co nejvíce chyb a tak zdokonalil kód. Tato verze byla pojmenována jako PHP verze 2 a již měla základní funkčnost jako má dnešní PHP. To zahrnuje například proměnné ve stylu Perlu, zpracování formulářů a možnost vložit HTML kód. Syntaxe byla obdobná jako u Perlu, ale byla omezenější, jednodušší a méně konzistentní.

Zeev Suraski a Andi Gutmans, dva Izraelští vývojáři na Technion IIT, přepsali parser v roce 1997, vytvořili tak základ PHP 3 a změnili název jazyka na rekurzivní zkratku PHP = PHP: Hypertext Preprocessor. Tým vývojářů oficiálně vydal PHP/FI 2 v Listopadu 1997 po měsíčním testování beta verze. Poté začalo veřejné testování PHP 3, a její oficiální uvolnění přišlo v červnu 1998. Zeev Suraski a Andi Gutmans poté začali opětovné přepisování jádra PHP a vydali Zend Engine v roce 1999. Založili firmu Zend Technologies v Ramat Gan, Izrael.

Dne 22. května 2000 byla vydána verze PHP 4 postavená na Zend Engine 1.0. Dne 13. června 2004 byla představena verze PHP 5, která již stojí na novém Zend Engine II. PHP 5 obsahuje nové rysy jako je vylepšená podpora pro objektově orientované programování, PHP Data Objects extension (ta definuje lehké a konzistentní rozhraní pro napojení k databázím) a nesčetné množství výkonových vylepšení. PHP 4 se již dále nevyvíjí a pro tuto verzi se nebudou vydávat ani žádné bezpečnostní aktualizace.

V roce 2008 se stává PHP 5 jedinou stabilní verzí, která se vyvíjí. Později se zjistilo, že zde chybí static binding a bude přidáno v PHP 5.3. PHP 6 se bude zároveň vyvíjet s verzí PHP 5. Mezi hlavní změny patří odebrání register_globals, magické uvozovky a safe mode.

PHP ještě plně nepodporuje Unicode nebo multibyte strings; podpora unicode bude zahrnuta až do verze PHP 6. Spousta kvalitních open sourcových projektů pozastavilo podporu PHP 4 ve svých nových kódech od 5. února 2008. Aby jim konsorcium vývojářů PHP usnadnilo přechod na PHP 5 poskytlo jim přechodovou verzi z PHP 4 na PHP 5.

PHP 5 běží jak ve 32-bitovém tak i v 64-bitovém prostředí, ale jedinou oficiální verzí pro Windows je 32-bitová verze, vyžadující mód kompatibility Windows 32-bit při použití IIS v 64-bitovém prostředí Windows. K dispozici je verze třetí strany, která je určena pro 64-bitové Windows. [4]

1.4 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.

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 na v současné době 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 poněkud scházet.

Přehled podporovaných vlastností:

- 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 a časových pásem v datech (od verze 4.1)
- poddotazy (od verze 4.1)
- uložené procedury (od verze 5.0)
- triggery (od verze 5.0)
- pohledy (od verze 5.0)
- práce s metadaty (od verze 5.0)

1.4.1 Uložiště dat

MySQL nabízí několik typů databázových tabulek (storage engine), které se liší svými možnostmi, použitím a způsobem ukládání dat do souborů:

- MyISAM - nejpoužívanější, bez podpory transakcí
- InnoDB - podpora transakcí
- BerkeleyDB (BDB)
- MEMORY - práci s daty v paměti
- NDB Cluster - úložiště pro clusterované databáze (od verze 5.0)
- ARCHIVE - komprimované tabulky, bez podpory indexů
- CSV - ukládání dat v prostých textových souborech

1.4.2 Správa MySQL přes PhpMyAdmin

Pro jednoduchou správu MySQL databází se používá nástroj PhpMyAdmin. PhpMyAdmin je Open Source program napsaný v PHP, který umožňuje zálohování, vytváření tabulek, vkládání, editaci a mazání záznamů v tabulkách, vytváření databází apod. PhpMyAdmin je pokročilý nástroj pro kompletní správu MySQL systému přes webové rozhraní. [5]

1.4.3 Příklady MySQL a PHP

1.4.3.1 Vybrání databáze

Abychom mohli s určitou databází pracovat, musíme ji vybrat. Každý databázový server může spravovat mnoho databází. Proto je dobré si hned po připojení databázi vybrat. K tomu slouží funkce **mysql_select_db**:

mysql_select_db (název_databáze, identifikátor_spojení)

1.4.3.2 Poslání SQL příkazu

K zaslání SQL příkazu můžeme použít funkci **mysql_query**:

```
mysql_query ( sql_příkaz, identifikátor_spojení )
```

Funkce mysql_query vrací buď identifikátor výsledku, pokud SQL příkaz vrací nějaká data, nebo pouze hodnotu true, pokud SQL dotaz nevrací žádná data. Při chybě vrací funkce mysql_query vždy hodnotu false.

1.4.3.3 Čtení výsledků SQL dotazu

Pro čtení výsledků SQL dotazu existují dvě šikovné funkce, a to **mysql_fetch_row** a **mysql_fetch_array**. Obě se nejčastěji používají s jedním parametrem:

```
mysql_fetch_row ( identifikátor_výsledku )
```

```
mysql_fetch_array ( identifikátor_výsledku )
```

Obě funkce načtou další řádek z výsledků SQL dotazu a vrací buď pole s hodnotami, a nebo false, pokud bylo už všechno přečteno. Liší se od sebe jen tím, že mysql_fetch_row načte hodnoty do pole s číselnými indexy, zatímco funkce mysql_fetch_array přidá i indexy s názvy sloupců.

Použití funkce `MySQL_Fetch_Row();` pro výpis všech hodnot tabulky:

```
$vysledek = MySQL_DB_Query("php", $sql, $spojeni);
while($zaznam = MySQL_Fetch_Row($vysledek)):
   echo $zaznam[0]."<br>";
   echo $zaznam[1]."<br>";
   echo $zaznam[2]."<br>";
   echo $zaznam[3]."<br><br>";
endwhile;
```

Použití funkce MySQL_Fetch_Array(); pro výpis všech hodnot tabulky:

```
$vysledek = MySQL_DB_Query("php", $sql, $spojeni);
while($zaznam = MySQL_Fetch_Array($vysledek)):
```

```
   echo $zaznam[rc]."<br>";
   echo $zaznam[jmeno]."<br>";
   echo $zaznam[adresa]."<br>";
   echo $zaznam[plat]."<br><br>";
endwhile;
```

Použití funkce MySQL_Fetch_Object(); pro výpis všech hodnot tabulky:

```
$vysledek = MySQL_DB_Query("php", $sql, $spojeni);
while($zaznam = MySQL_Fetch_Object($vysledek)):
   echo $zaznam->rc."<br>";
   echo $zaznam->jmeno."<br>";
   echo $zaznam->adresa."<br>";
   echo $zaznam->plat."<br><br>";
endwhile;
```
[6]

1.5 Wolfram Mathematica

Wolfram Mathematica je software pro počítání čehokoliv vás napadne. Vypsat všechny funkce, které program obsahuje je nad moje schopnosti (u 99% bych ani nepochopil co mají dělat, nicméně pro představu mu nedělají problém integrály, derivace, maticové počty, sumy a spousta jiných věcí. Dále samozřejmě není problém s tvorbou grafů.

Program disponuje obsáhlou nápovědou. V ní je seznam všech příkazů, včetně mnoha ukázek použití.

Za zmínku taktéž stojí, že jeden výpočet lze rozdělit mezi více počítačů.

K dispozici je zdarma pouze 30 denní trial verze, kde není možno ukládat. Cena plné verze je poněkud vysoká. Studenti mají cenu samozřejmě nižší, navíc vysokoškoláci mají možná to štěstí, že jejich fakulta má zakoupenou celoškolní licenci (UTB má například licenci, kdy člověk může používat Mathematicu ve škole.

1.5.1 Co je webMathematica?

web*Mathematica* je produkt, který prostřednictvím integrace *Mathematica* s nejnovější technologií webových serverů rozšiřuje možnosti webových stránek o interaktivní výpočty a vizualizace.

1.5.2 Čím se liší webMathematica od Mathematica?

web*Mathematica* a *Mathematica* mají stejný "engine", ale poskytují zásadně rozdílná uživatelská rozhraní a jsou zaměřené na různé typy uživatelů.

web*Mathematica* nabízí přístup ke specifickým aplikacím programu *Mathematica* prostřednictvím webového prohlížeče nebo jiných webových klientů. Stačí pouze malý zácvik k tomu, abyste standardní rozhraní mohli efektivně používat. Ve většině případů uživatelé nemusí *Mathematica* znát, dokonce ani nemusí vědět, že ji používají.

V určitém slova smyslu lze *Mathematica* považovat za vývojové prostředí pro stránky web*Mathematica*. *Mathematica* je například vhodná pro práci na kódu, který modeluje určitý fyzikální proces - kódu, který se následně dá umístit do stránky web*Mathematica*, aby se lidem umožnilo fungování modelu a využití jeho výsledků při běžné práci. [7]

1.5.3 Co je formát JSP?

JSP je zkratka pojmu Java Server Pages. Jedná se o technologii pro tvorbu dynamických webových stránek na první pohled podobnou dobře známému PHP. Instrukce se zapisují do dokumentu mezi značky **<?** a **?>**. Dokument může vypadat například takto:

```
<html>
  <body>
    Náhodné číslo:
    <?= java.lang.Math.random() ?>
  </body>
</html>
```

Když dáte tento text do souboru a uložíte ho do adresáře servírovaného webovým serverem do světa a někdo napíše jeho adresu, server tento text přeloží na zdrojový kód servletu (.java), ten dále přeloží do bytecode (.class) a s tím už zachází jako se servletem. To zna-

mená - načte ho a vzniklému objektu předá HTTP požadavek. Vyřízení prvního dotazu zabere nějaký čas (řekněme sekundu), ale objekt pak zůstane v paměti a čeká na další dotazy, takže zpracování dalších zpráv už je dílem okamžiku. [8]

1.6 SMARTY

Smarty je šablonovací systém pro PHP, který umožňuje oddělení aplikační (samotný kód PHP) a zobrazovací (klasické HTML) logiky. To se hodí především pro větší projekty, kdy programátor PHP a kodér HTML není tatáž stejná osoba. Díky Smarty tak kodér úpravami HTML nezničí práci programátora. Kodér tedy vždy pracuje jen s HTML a programátor jen s PHP.

Smarty funguje tak, že kodér upravuje klasickou stránku HTML (šablonu) a v ní jsou určité značky, které jsou dynamicky vytvářené pomocí PHP. Taková šablona může např. vypadat takto (index.tpl):

```
<html>
<body>
 Vítejte {$jmeno}
</body>
</html>
```

Právě značka `{$jmeno}` bude zajímat programátora. Ten v jiném skriptu zjistí (třeba z databáze) jméno návštěvníka, které pak pomocí metod třídy Smarty zamění za `{$jmeno}` v šabloně. Příklad:

```
<?php
/*Skript vybere z databáze jméno
návštěvníka a přiřadí ho proměnné $name */

  $smarty->assign('jmeno','$name');
  $smarty->display('index.tpl');
?>
```

Metoda `$smarty->assign` zamění proměnnou `$name` za značku `{$jmeno}` v šabloně. Druhá metoda pak načte a zobrazí šablonu (tento skript by nefungoval, museli bychom ještě zahrnout třídu Smarty, ale pro zjednodušení je to myslím dostačující).

Velkou předností je kompilace šablon. Pokud zavoláme předchozí skript v prohlížeči, Smarty vytvoří klasický skript PHP i se zobrazovací logikou do specialního adresáře. Při příštím volání funkce již není nutné znova značky převádět, zavolá se zkompilovaná verze. Proč to Smarty dělá? Je to především rychlejší.

Ještě mnohem rychlejší než kompilace je cachování obsahu. Cachování je v podstatě kompilace, kdy se nevytváří PHP skripty, ale klasické HTML. Cachování je vysvětleno velmi podrobně později. [9]

II. PRAKTICKÁ ČÁST

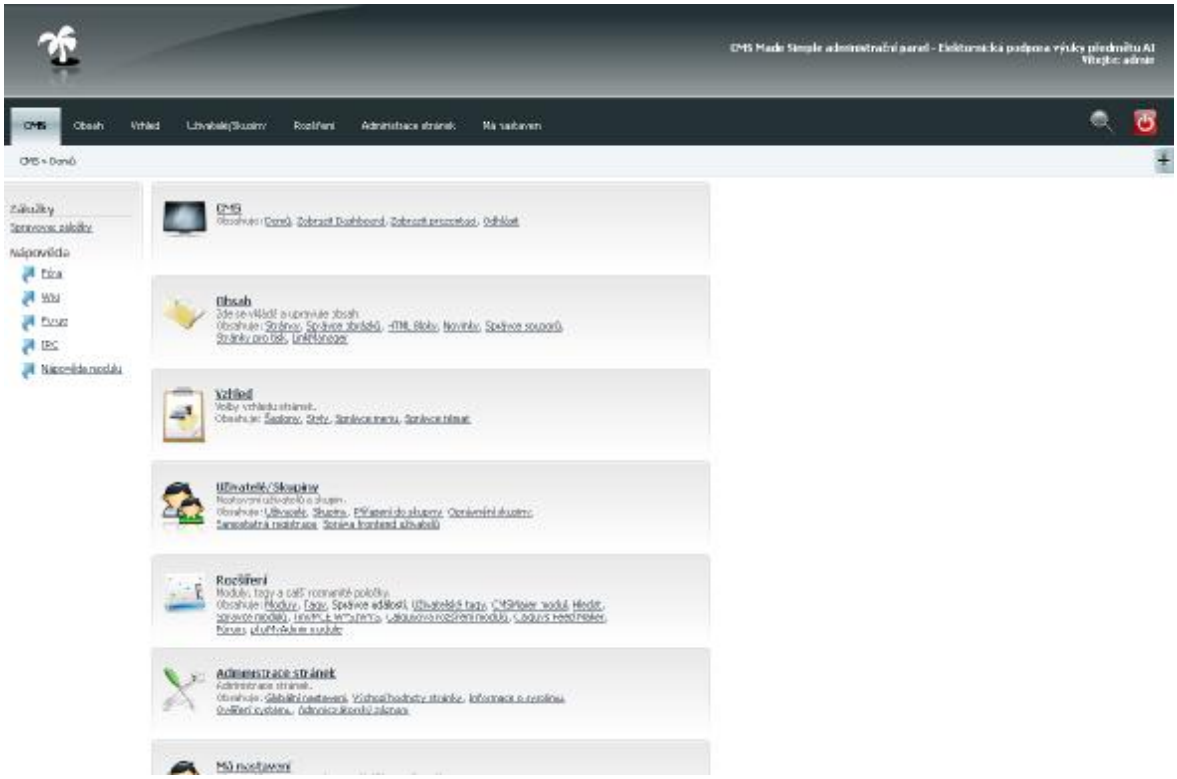
2 SYSTÉM PRO SPRÁVU

2.1 Funkce CMS Made Simple

Funkce CMS bychom mohli rozdělit do těchto kategorií:

- Vytvoření obsahu
- Správa obsahu
- Publikování
- Prezentace

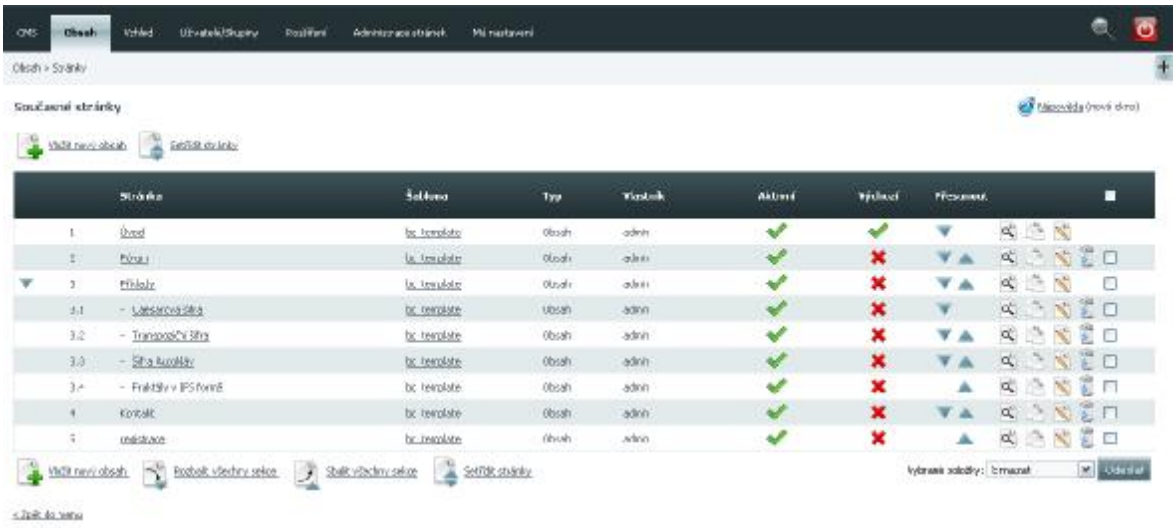
Každá oblast bude rozebrána níže.



Obr. 1 : Vzhled administrace

2.1.1 Vytváření obsahu

Vytváření obsahu v CMS Made simple je velmi jednoduchou záležitostí. V našem případě byla vytvořena stromová struktura menu zobrazená na *Obr. 2*.

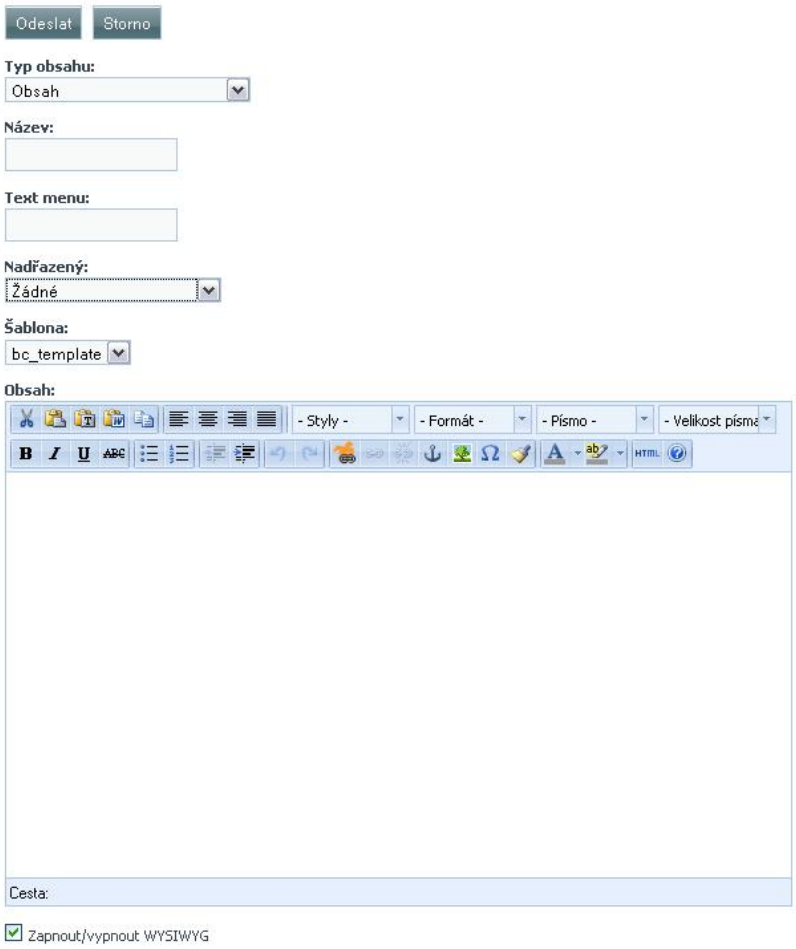


Obr. 2 : Ukázka administrace správy obsahu

Obsah je jednoduše upravitelný, mazatelný a v případě potřeby přesunovatelný výše či níže. Na *Obr. 3* je zobrazena šablona, ve které se má daný obsah zobrazit. Další vlastností je typ obsahu, vlastník, zda je stránka aktivní a výchozí stav stránky. Šablona bývá ve většině případů pro celý obsah stránek stejná, protože stránky svůj vzhled nemění.

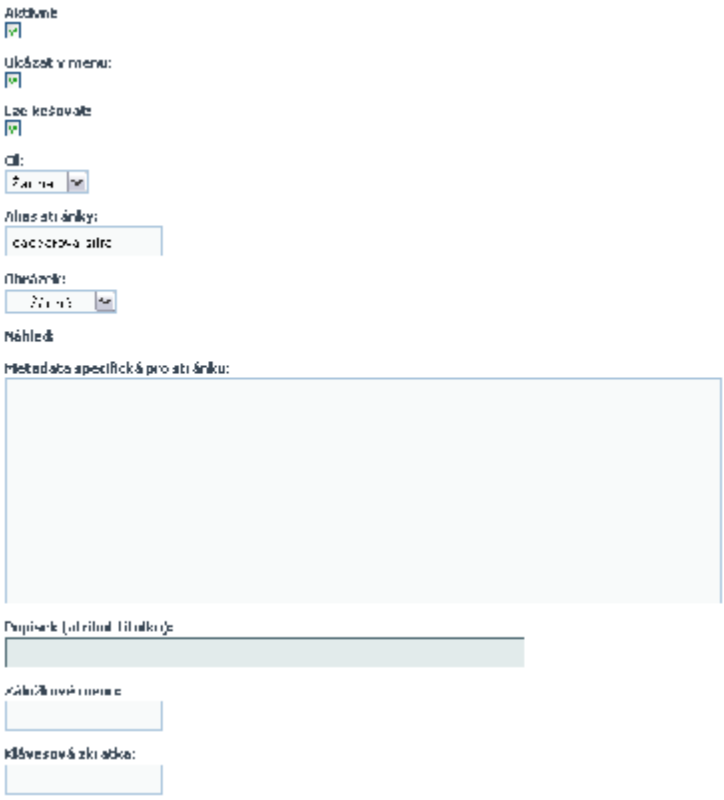
Samotné vložení obsahu má tato pole:

- typ obsahu
- název
- text menu
- nadřazenost
- šablonu
- obsah



Obr. 3 : Vkládání nového obsahu

V záložce „Volby“ je spousta dodatečného nastavení obsahu, jako například zda je obsah aktivní, zda se má zobrazit v menu, zda se má zobrazit na stejné stránce, nebo na nové. Alias stránky je název, který se předává jako parametr proměnné index.php?page=“Alias stránky“.

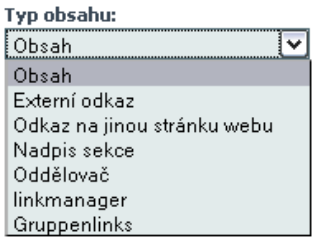


Obr. 4 : Nabídka voleb při vkládání obsahu

Text menu je v podstatě název s tím rozdílem, že název je zobrazen jen v administraci, ale text menu je zobrazeno na stránkách webové prezentace v menu.

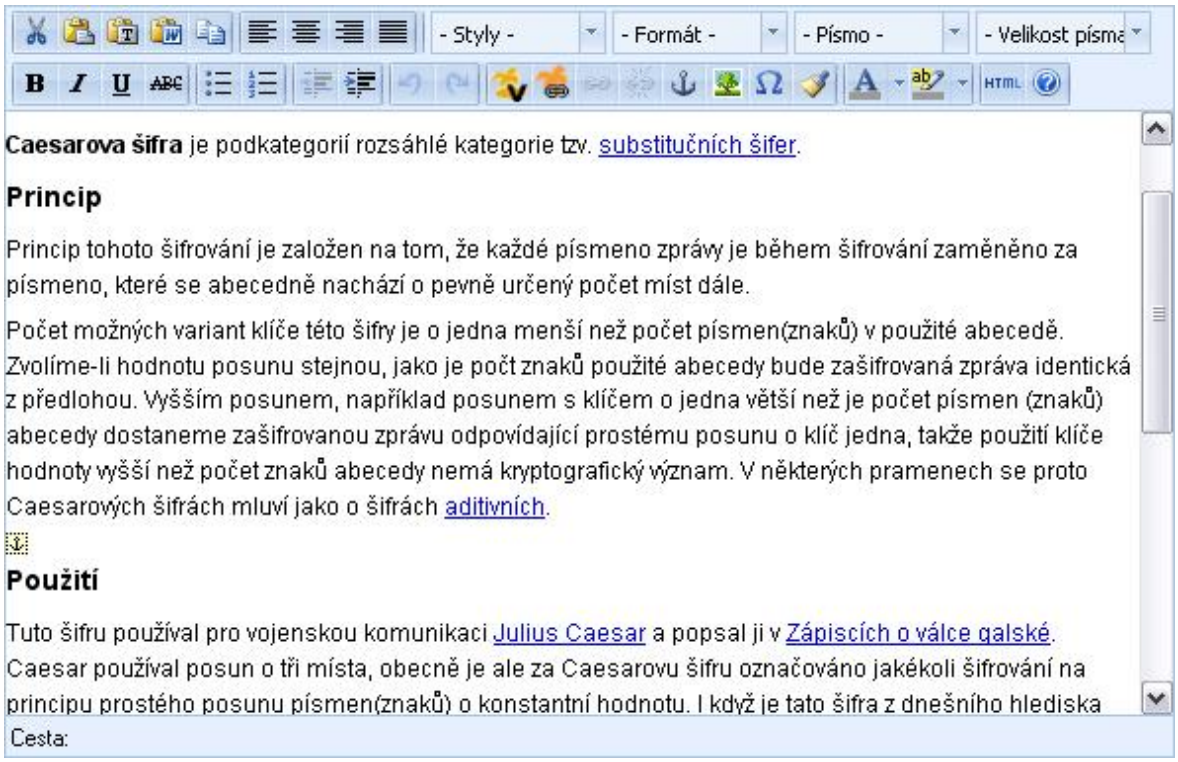
Nadřazenost je velmi důležitým prvkem nového obsahu. V případě že chcete, aby obsah byl umístěn úplně nahoře ve stromové struktuře menu, v `select` boxu necháte volbu „Žádné“, v opačném případě jednoduše vyberete menu položku, pod kterou chcete aby menu spadalo. Dále vyplníte obsah a uložíte.

Při vkládání nového obsahu se může jednat o klasický typ obsahu, externí odkaz, ale i další typy. Některé moduly nabízí přímo v této nabídce odkaz na vlastní strukturu stránky, jak je tomu například u fotoalb. Na *Obr. 5* je zobrazen seznam základních voleb.



Obr. 5 : Typ obsahu

Mezi obrovské výhody z pohledu vytváření patří jednoduché prostředí, které disponuje velmi podobnými funkcemi jako Word. CMS Made Simple využívá WYSIWYG plugin TinyMCE s předdefinovaným layoutem Office 2007.



Obr. 6 : Prostředí pro vytváření obsahu

Další vlastností je provázanost odkazů mezi stránkami na webu. Tato vlastnost u CMS Made Simple je v editoru zobrazena jako „Palma s řetízkem“.

Obr. 7 : Odkaz na stránky CMS

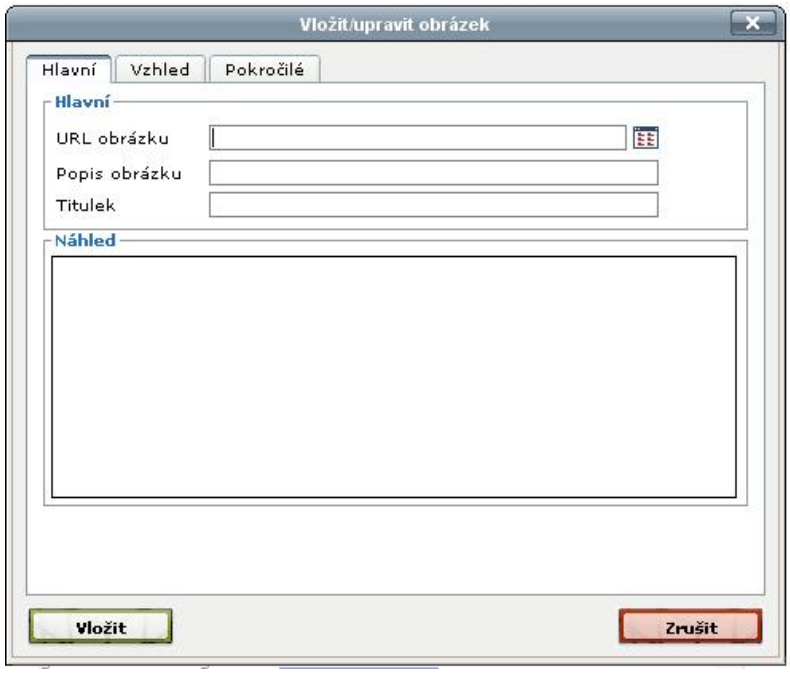
Po kliknutí na ikonku se zobrazí menu, které ukáže kompletní stromovou strukturu menu a po vybrání požadované stránky se na místě kurzoru objeví kompletně vygenerovaný html kód s odkazem.

2.1.1.1 Vkládání obrázků

Práce s obrázky je velmi jednoduchou záležitostí, díky propracovanosti pluginu TinyMCE. K vyvolání nabídky slouží ikonka „Vložit/Upravit Obrázek“. Některé další modifikace pluginu dokáží nahrát obrázek přímo z okna na *Obr. 9*. Tuto vlastnost bohužel tato verze nepodporuje. Jestliže chcete vložit obrázek na stránky, musíte je nejdříve nahrát na server. Postup pří nahrávání obrázků bude popsán níže.

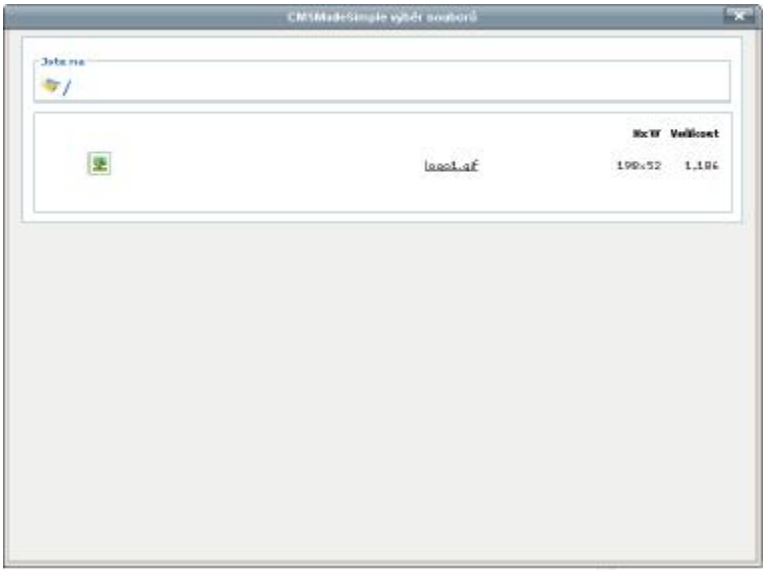
Obr. 8 : Vložit/Upravit Obrázek

Po kliknutí na ikonku se zobrazí dialogové okno s několika záložkami. Pro jednoduchost a rychlost vkládání obrázku stačí pouze hlavní záložka.



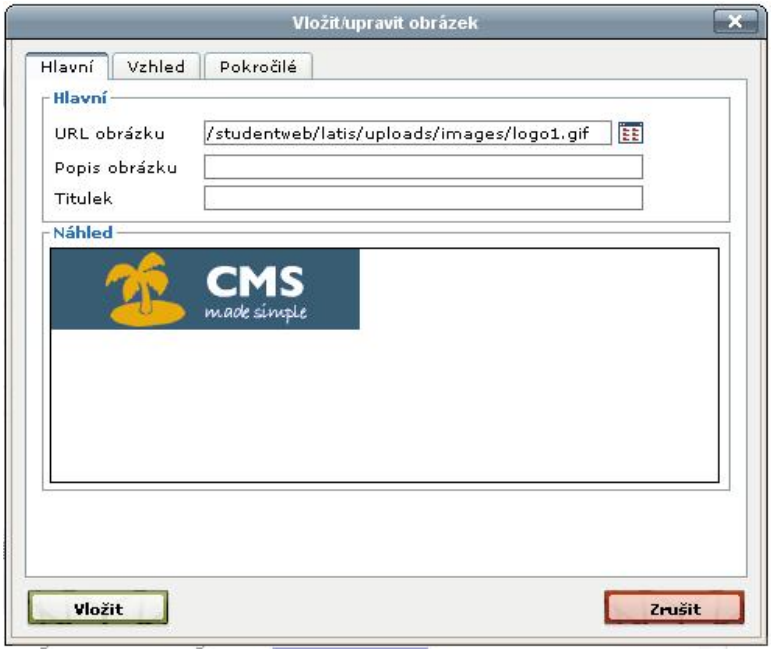
Obr. 9 : Vložit/Upravit Obrázek

Kliknutím na tlačítko „Procházet“ se zobrazí další dialogové okno, kde si jednoduše vyberete požadovaný obrázek a kliknutím na něj zavřete okno a vrátíte se do prvního dialogového okna.



Obr. 10 : Výběr obrázku

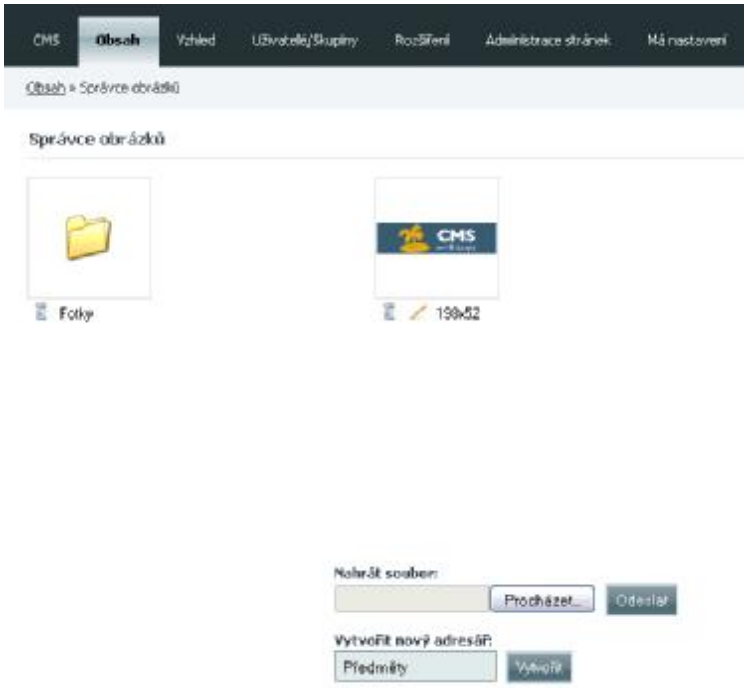
Tentokráte již bude URL obrázku vyplněna s relativní cestou k hlavní složce (documentRoot) na serveru a v náhledu bude zobrazena skutečná velikost obrázku. Další úpravy jsou již na Vás, nicméně stačí kliknout na tlačítko vložit a obrázek se zobrazí na požadovaném místě.



Obr. 11 : Vložit/Upravit Obrázek

2.1.1.2 Nahrávání obrázků na server

Nahrávání obrázku je specifickou záležitostí podle verze a konfigurace pluginu. V hlavním menu administrace si nalistujeme záložku „Obsah-Správce obrázků“. Zobrazí se list všech obrázků ve složce /images/. Pokud chcete své obrázky rozdělovat do adresářů, není nic jednoduššího než vyplnit formulář pro vytváření adresářů (viz *Obr. 12*). Adresáře můžete libovolně vnořovat pod sebe, mazat a modifikovat.



Obr. 12 : Správce obrázků

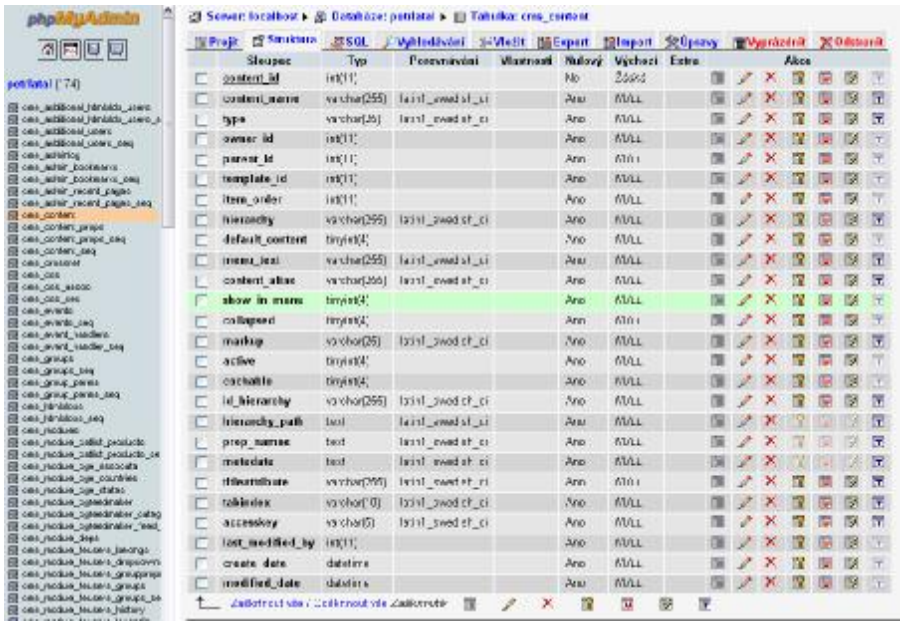
2.1.1.3 Editace obrázků na serveru

Správce má integrovaný editor obrázků, který Vám umožní s obrázkem na webové úrovni udělat základní úpravy jako například ořezání, otočení a změna velikosti. Po úpravách lze obrázek na servru uložit. Obrázek se uloží ve dvou variantách. Vám se však zobrazí pouze upravená verze.

Obr. 13 : Editor obrázků

2.1.2 Správa obsahu

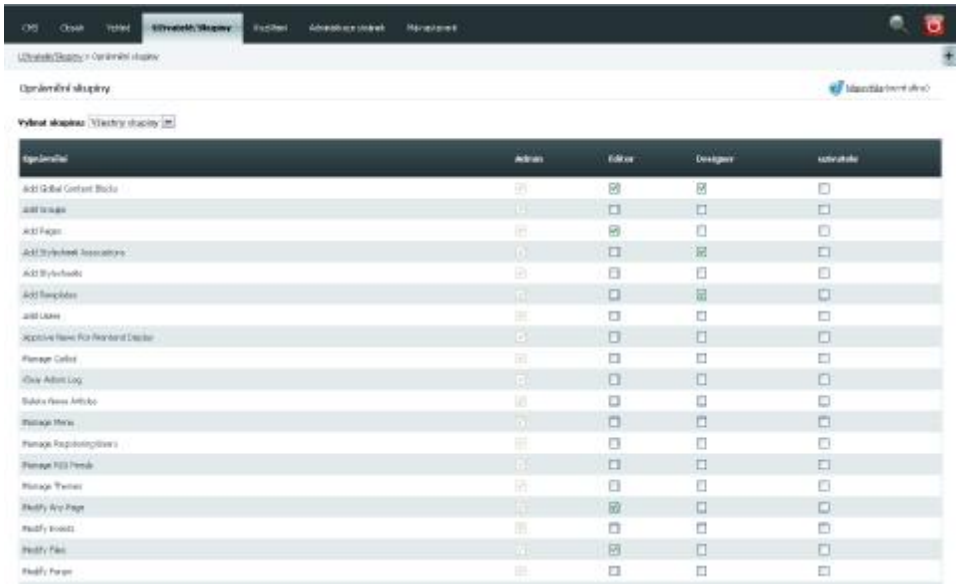
Po vytvoření obsahu stránky budou data uložena do MySQL databáze CMS systému. V databázi je uložen veškerý obsah společně s dalšími údaji jako například historie úprav stránek, kdo tyto úpravy provedl a kdy. Databáze také obsahuje položku, ve které je uloženo oprávnění pro přístup a úpravy obsahu.



Obr. 14 : Data uložená v databázi

2.1.2.1 Workflow procesy

CMS Made simple umožňuje rozsáhlou podporu tzv. workflow procesů. To znamená, že je zde hierarchie, do které spadá každá uživatelem vytvořená stránka. Stránka je předána ke schválení příslušnému redaktorovi a teprve po schválení je zobrazena na webových stránkách. Tento proces schvalování může být několikanásobný, podle hierarchie schvalovacích práv. V každém schvalovacím kroku řídí CMS Made simple status stránky a upozorňuje uživatele systému na potřebné kroky.

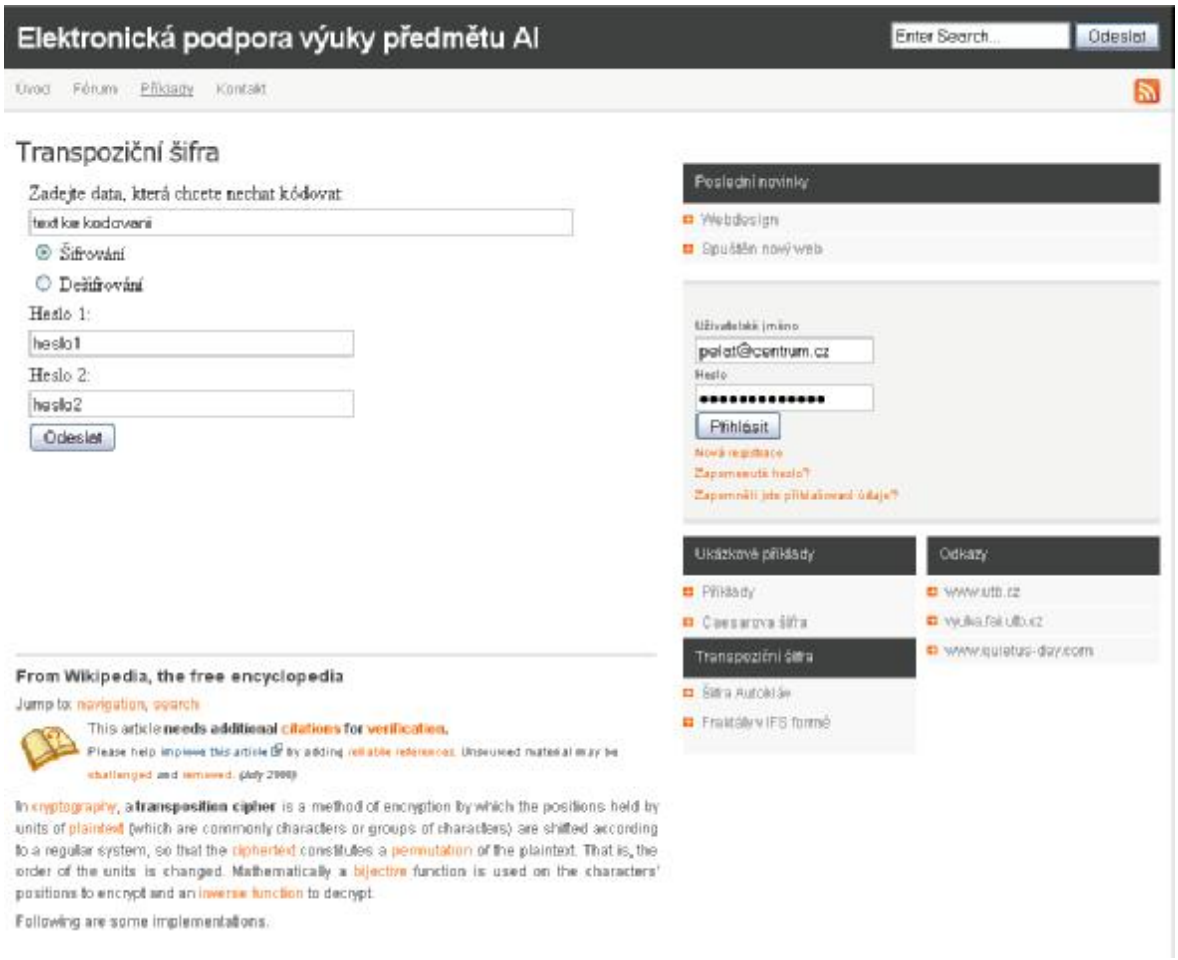


Obr. 15 : Přístupová práva uživatelských skupin

Díky tomu může více autorů řídit obsah stránek při zachování přísné kontroly nad kvalitou, správností a soudržností webu. Pravidla pro workflow přináší pořádek do chaosu při manuálním způsobu zpracování informací.

2.1.3 Publikování a prezentace

Po úspěšném vložení dat do databáze, může být obsah publikován na stránkách.



Obr. 16 : Publikace stránek na webu

Velkou výhodou CMS systému je možnost zobrazovat obsah se stejným vzhledem. Tím pádem může být stejný obsah zobrazen na různých webových stránkách. Samozřejmě obvykle má každý web jiný vzhled, proto CMS odděluje obsah od vzhledu a ten aplikují nezávisle na obsahu. To umožňuje autorům soustředit se na obsah a vzhled nechat na CMS.

V neposlední řádě existuje spousta nástrojů pro zlepšení efektivnosti stránek. Například navigace stránek „sitemap“. Tento nástroj indexuje veškeré stránky na webu do souboru (například sitemap.xml) a díky tomuto souboru může být web lépe indexovatelný a snáze

hledatelný google roboty. Tuto vlastnost však musíte mít aktivní přímo na stránkách www.google.com/webmastertools.

2.2 Parsování zdrojového kódu mathematicy

Zdrojový kód Mathematica bylo třeba parsovat do formátu JSP. Pomocí editoru PSPad jsem vytvořil potřebnou hlavičku ve formátu:

```
<%@ page language="java"  %>
<%@ taglib uri="/webMathematica-taglib" prefix="msp" %>
```

Tato hlavička udává jazyk, kterým se bude zdrojový kód kódovat. Druhý řádek pak udává cestu k hlavičce s požadovaným prefixem. Na dalších řádcích je pak definována XHTML hlavička:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
  <meta http-equiv="content-type" content="text/html; charset=windows-
1250">

  <title>Transpozicni Šifra</title>
  </head>
```

Tato hlavička hlavně definuje typ kódování stránky se znakovou sadou windows-1250 a standart 4.01 Transitional. Tento typ programátorovi udává, jakým způsobem jsou definovány HTML tagy a dovoluje mu používat i některé zastaralé tagy (například tag `<menu>`).

Na dalších řádcích je definována samotná struktura JSP, kde je celý kód obalen v tagu :

```
<msp:allocateKernel>
</msp:allocateKernel>
```

Tento tag je nesmírně důležitý pro alokaci kernelu webMathematicy. Kdybychom spouštěli kód bez uzavření mezi tímto tagem, proces by nám uvízl na serveru a pravděpodobně bychom nemohli pokračovat ve výpočtech.

Důležitou částí pak byla sekvence, která nám udává míru zabezpečení a potřebné knihovny. V mém případě jsem potřeboval knihovnu MSP pro práci s vykreslováním obrázků,grafů a manipulate ze serveru na výstup.

```
<msp:evaluate>
Needs[ "MSP`" ];
SetSecurity[];
</msp:evaluate>
```

Dále následoval HTML formulář pro vstupní data. Většinou bylo třeba tagů `<input>` a `<select>`. Zde je ukázka formuláře pro transpoziční šifru a také způsob jakým se přebírají hodnoty:

```
<form action="latis-transpozice.jsp" method="post">
<table>
       <tr>
         <td>Zadejte data, která chcete nechat kódovat:</td>
       </tr>
       <tr>
         <td><input type="text" value="<msp:evaluate>MSPValue[ $$data,
"text ke kodovani" ]</msp:evaluate>" name="data" size="70" /></td>
       </tr>
       <tr>
         <td><input type="radio" name="desifrovani" value="False" chec-
ked="checked" /> Šifrování </td>
       </tr>
       <tr>
         <td><input type="radio" name="desifrovani" value="True" /> De-
šifrování </td>
       </tr>
```

```
      <tr>
        <td>Heslo 1:</td>
      </tr>
      <tr>
        <td><input type="text" value="<msp:evaluate>MSPValue[ $$heslo1,
"heslo1" ]</msp:evaluate>" name="heslo1" size="40" /></td>
      </tr>
      <tr>
        <td>Heslo 2:</td>
      </tr>
      <tr>
        <td><input type="text" value="<msp:evaluate>MSPValue[ $$heslo2,
"heslo2" ]</msp:evaluate>" name="heslo2" size="40" /></td>
      </tr>

      <tr>
        <td><input type="submit" name="odeslat" value="Odeslat" /></td>
      </tr>
</table>
</form>
```

Tělo JSP je obaleno tagem `<msp:evaluace>` a obsahuje podmínku, která ověřuje zda byl odeslán formulář. Zde je ukázka kódu :

```
<msp:evaluate>
If[ ValueQ[ $$odeslat ],

DvojitaTranspozice[data_, heslo_, heslo2_, desifrovani_] :=
  Module[{delkaDat, delkaHesla, delkaHesla2, zaokrouhleni,
    zaokrouhleni2, text1, zbytek, zbytek2, znak, znak2, matice,
    matice1, matice2, delkaZasifrovanehoTextu, osetrenyText,
    osetreneHeslo, osetreneHeslo2, heslo2Upravene, serazeneHeslo2,
```

```
    testvystup},
      .
      .
      .
     Print[" Heslo2:",heslo2];
     Print[typsifrovani];
]
</msp:evaluate>
```

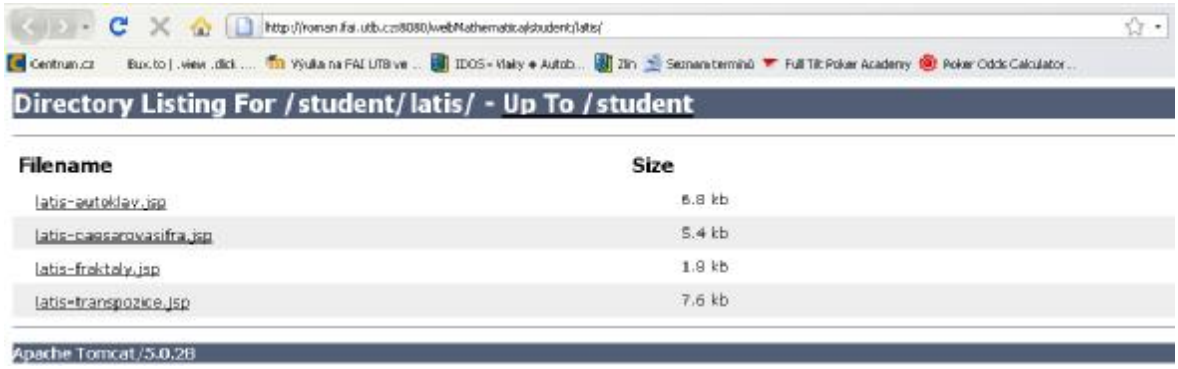
Na konci zdrojového kódu je funkce, která v případě problému na servru vypíše systémovou chybu, ale jinak slouží k vypsání konsolového výstupu, který jsem byl nucen využít, protože funkce Manipulate, kterou jsem chtěl obejít potřebné HTML formuláře je dostupná až ve vyšších verzích webMathematicy.

```
<msp:evaluate>
ColumnForm[ MSPGetPrintOutput[]]
</msp:evaluate>
```

Práce s proměnnými byla velmi jednoduchá, protože se předávaly ve formátu $$promenna.

2.3 Kompletace redakčního systému

Poté co byly naprogramovány jednotlivé typy šifer, bylo třeba je nahrát na server. Spouštění těchto skriptů bylo realizováno na portu 8080 a to z důvodu blokování standartního portu 80 pro webové rozhraní službou Apache.



Obr. 17 : Zdrojové kódy Webmathematicy na portu 8080

webMathematika běží pod službou Tomcat, ta však nepodporuje službu PHP, která je nezbytná pro správnou funkci redakčního systému CMS Made Simple.

K tomu aby bylo možné propojit služby Apache a Tomcat, bylo využito HTML tagu `<iframe></iframe>`. Jedná se o párový tag, který se dnes již nevyužívá tak, jak tomu bylo před pár lety. Tento tag s příslušnými parametry `src`, `width`,`height`, `frameborder` umožňuje zobrazit stránku i z jiných serverů nebo jen jiného umístění na serveru. K tomu slouží právě parametr `src`, do kterého byla vložena stejná adresa serveru, na kterém systém běží, ale s portem 8080 a jinou cestou ke zdrojovým kódům.

```
<iframe src="http://roman.fai.utb.cz:8080/webMathematica/student/latis/latis-caesarovasifra.jsp"
```

Obr. 18 : parametr src u tagu iframe

2.3.1 Příprava JSP kódu pro vložení do obsahu

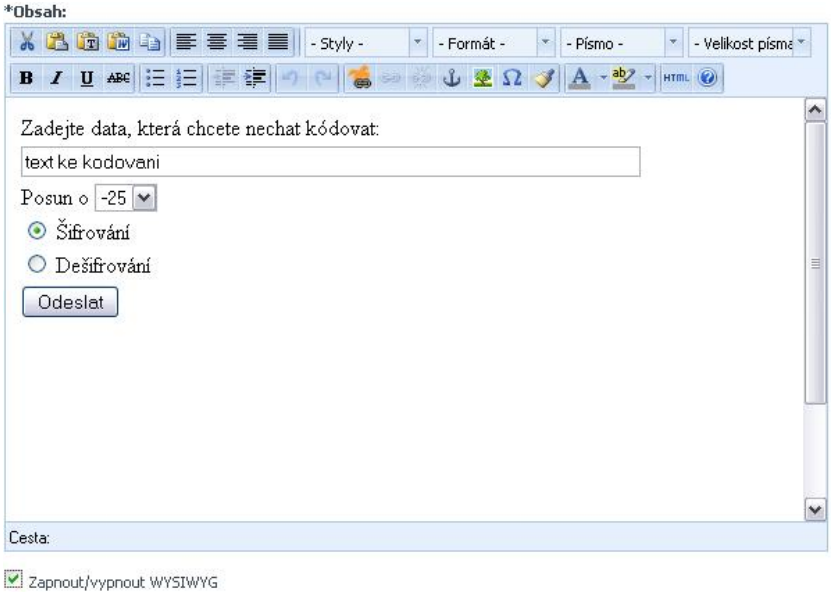
Vkládání JSP zdrojových kódů bylo realizováno pomocí modulu HTML bloky, kde si jako uživatel můžete napsat určitou část HTML kódu a pak se jednoduše na daný kód odkazovat v libovolném obsahu pomocí SMARTY tagu.

Uživatelské jméno	Tag pro použití tohoto bloku
Caesarova sifra	{global_content name='Caesarova sifra'}
Fraktaly	{global_content name='Fraktaly'}
Transpozicni sifra	{global_content name='Transpozicni sifra'}
Šifra autoklav	{global_content name='Šifra autoklav'}

Obr. 19 : HTML bloky se SMARTY tagy

Na prvním obrázku je vidět zdrojový kód pro zobrazení caesarovy šifry a příslušnými parametry. Ve spodní části obsahu je tlačítko, které obsluhuje stav WYSIWYG pluginu. Jeho velikou nevýhodou je to, že pokud přepínáte mezi jednotlivými stavy zapnuto/vypnuto, zdrojový kód se již nezobrazí a i při uložení je ztracen. Uživatel je tak nucen HTML kód znovu vložit nebo ručně vepsat což může být otravné a časově ztrátové.

Obr. 20 : HTML blok s WYSIWYG ve stavu vypnuto

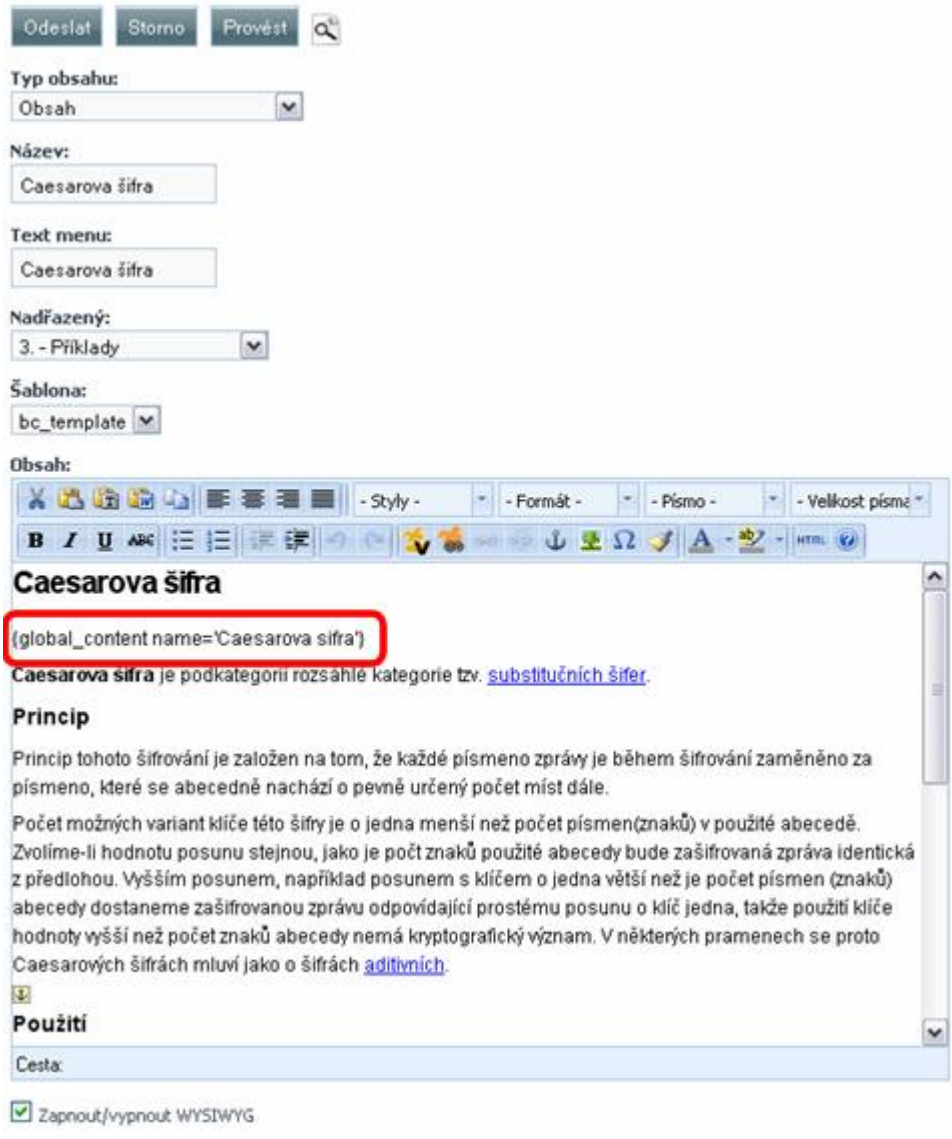


Obr. 21 : HTML blok s WYSIWYG ve stavu zapnuto

Výsledkem je tedy výstup, který pro běžného uživatele nijak nenaznačuje, že by na pozadí byly data jiné služby nebo serveru.

2.3.2 Vložení JSP kódu do obsahu

Vložením SMARTY tagu do obsahu je uzavřena cesta, jakým způsobem byl vložen rádoby zdrojový kód mathematicy do redakčního systému CMS Made Simple. Důležitým parametrem je tag name obsahující odkaz na jednotlivé šifry.



Obr. 22 : Vložení SMARTY tagu do obsahu s popisem šifer

2.3.3 Zobrazení na stránkách

Na *Obr. 23* je zobrazen vzorový příklad caesarovy šifry s popisem jak šifra funguje. Dále je zde vidět výstup dat po zašifrování a obecné informace o zpracování.

Obr. 23 : Ukázka caesarovy šifry

ZÁVĚR

Cílem této bakalářské práce bylo vytvořit webové stránky pro podporu výuky předmětu Aplikovaná Informatika.

V teoretické části jsem se zaměřil na jazyky a nástroje potřebné k realizaci celé bakalářské práce. Mezi základní prvky tak patřil redakční systém CMS Made Simple, který běží na PHP, MySQL a HTML. Dále bylo třeba popsat co je to formát JSP, co je to SMARTY a jakým způsobem se používá. Na závěr bylo velmi jednoduše popsáno prostředí webMathematicy.

V praktické části jsem se snažil popsat CMS Made Simple z hlediska vkládání, editace a celkových úprav obsahu. Uživatel by měl být schopný po přečtení této práce vkládat libovolný obsah a obrázky do tohoto systému. Dále by měl pochopit způsob, jakým se parsoval zdrojový kód Mathematicy v editoru PSPad do formátu JSP a na závěr byl popsán způsob, jak hotový JSP soubor vložit do obsahu stránek. V příloze jsou přidány zdrojové kódy 4 šifer, které jsem programoval v hodinách Aplikované Informatiky.

Závěrem bych rád podotknul, že tento systém je pilotním projektem, který bych rád dokončil v diplomové práci na toto téma. Mělo by se jednat o to, že uživatel, který dostane jistá oprávnění na webu, bude mít administraci webové rozhraní přímo na import souborů Mathematicy (*.nb), které budou následně konvertovány skriptem do formátu JSP, uloženy na server a umístěny do databáze systému. Uživatel si tak jednoduše vybere kód, který následně vloží do obsahu stránek. Celý systém pak poběží pod portem 80, čili již nebude třeba využívat `iframe`, jako přechodové prostředí.

CONCLUSION

This graduation thesis is focused on creation of web pages for education support to the subject Applied Informatics.

In the theoretical part I concentrated myself on languages and tools needed for realization of the whole graduation thesis. The basic stock was the content management system called CMS Made Simple, which runs on PHP,MySQL and HTML. The next part describes the source code format JSP, as well as the SMARTY language and the way how to use it. By way of ending, I tried to describe the development environment of the webMathematica.

In the practical part of this thesis, I tried to describe the content management system in the aspect of editing and adding content to the webpages. The user should be able to add the content with pictures to this system. The user should understand the way of parsing the source code of Mathematica, using the editor PSPad to source code JSP. The last step was the adding of JSP code to the content of the webpages. Added in the supplement are 4 source codes of ciphers that were programmed in the lessons in the subject Applied Informatics.

I would like to note, that this system is a pilot project, which I would like to finish in the diploma paper. The users who have the necessary permissions will be given a web interface for importing files (*.nb), which will be converted by the script to JSP format and than stored on server. The user will receive the code containing JSP data and add it to the content. The system will run under port 80 so that no `iframe` code will be necessary.

SEZNAM POUŽITÉ LITERATURY

[1] Co je CMS [online]. [cit. 2009-05-27]. Dostupný z WWW: http://www.voxcafe.cz/clanky/optimalizace-stranek/co-je-cms.html

[2] Open Source Software [online]. [cit. 2009-05-27]. Dostupný z WWW: http://cs.wikipedia.org/wiki/Open_source_software

[3] HTML [online]. [cit. 2009-05-27]. Dostupný z WWW: http://cs.wikipedia.org/wiki/HyperText_Markup_Language

[4] PHP [online]. [cit. 2009-05-27]. Dostupný z WWW: http://cs.wikipedia.org/wiki/Php

[5] MySQL [online]. [cit. 2009-05-27]. Dostupný z WWW: http://cs.wikipedia.org/wiki/Mysql

[6] Příklady MySQL [online]. [cit. 2009-05-27]. Dostupný z WWW: http://vyuka.fai.utb.cz/mod/resource/view.php?id=602

[7] Wolfram Mathematica [online]. [cit. 2009-05-27]. Dostupný z WWW: http://www.abclinuxu.cz/software/veda/mathematica

[8] Co je to JSP [online]. [cit. 2009-05-27]. Dostupný z WWW: http://www.jsphosting.cz/co-je-to-jsp/

[9] Co je to SMARTY [online]. [cit. 2009-05-27]. Dostupný z WWW: http://smarty.ronnieweb.net/co-je-smarty.php

[10] ULLMAN, Larry. HTML : PHP a MySQL. Brno : Computer Press, 2004. ISBN 80-251-0063-4.

[11] HENK C.A : van TILBORG. Fundamentals of cryptology. Norwell : Kluwer Academic, 2000. ISBN 0-7923-8675-2.

[12] ZELINKA, Ivan. Aplikovaná informatika. Zlín : Ediční středisko, UTB, 1999. ISBN 80-214-1423-5.

[13] SCHLOSSNAGLE, Georgie. Pokročilé programování v PHP 5. Computer Press, 2004. ISBN 80-86815-14-5.

[14] LACKO, Luboslav. PHP5 a MYSQL5 Hotová řešení. Computer Press, 2007. ISBN 978-80-251-1695-1.

[15] KOLEKTIV AUTORŮ. Mistrovství v PHP 5. Computer Press, 2007. ISBN 978-80-251-1519-0.

[16] CASTRO, Elizabeth. HTML : XHTML a CSS – Názorný průvodce tvorbou WWW stránek. Computer Press, 2007. ISBN 978-80-251-1531-2.

[17] RESIG, John. Javascript a AJAX : Moderní programování webových aplikací. Computer Press,2007. ISBN 978-80-251-1824-5.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AI	Předmět aplikovaná informatika
HTML	Význam třetí zkratky.
tag	Část značek v kódu HTML jazyka
JSP	Zkratka pojmu Java Server Pages
plugin	Zásuvný modul
workflow	Způsob řízení procesu schvalování obsahu
sitemap	Mapa stránek
kernel	Jádro systému

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

PŘÍLOHA P I: CAESAROVA ŠIFRA VE FORMÁTU JSP

```
<%@ page language="java"  %>
<%@ taglib uri="/webMathematica-taglib" prefix="msp" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
  <meta http-equiv="content-type" content="text/html; charset=windows-1250">

  <title>Caesarova Šifra</title>
  </head>
  <body>

<msp:allocateKernel>

<msp:evaluate>
Needs[ "MSP`" ];
SetSecurity[];
</msp:evaluate>

<form action="latis-caesarovasifra.jsp" method="post">
<table>
       <tr>
         <td>Zadejte data, která chcete nechat kódovat:</td>
       </tr>
       <tr>
         <td><input type="text" value="<msp:evaluate>MSPValue[ $$data,  "text ke kodovani"
]</msp:evaluate>" name="data" size="70" /></td>
       </tr>
       <tr>
         <td>Posun o <select name="posun">
         <option value="-25" >-25</option>
         <option value="-24" >-24</option>
         <option value="-23" >-23</option>
         <option value="-22" >-22</option>
         <option value="-21" >-21</option>
         <option value="-20" >-20</option>
         <option value="-19" >-19</option>
         <option value="-18" >-18</option>
         <option value="-17" >-17</option>
         <option value="-16" >-16</option>
         <option value="-15" >-15</option>
         <option value="-14" >-14</option>
         <option value="-13" >-13</option>
         <option value="-12" >-12</option>
         <option value="-11" >-11</option>
         <option value="-10" >-10</option>
```

```
          <option value="-9" >-9</option>
          <option value="-8" >-8</option>
          <option value="-7" >-7</option>
          <option value="-6" >-6</option>
          <option value="-5" >-5</option>
          <option value="-4" >-4</option>
          <option value="-3" >-3</option>
          <option value="-2" >-2</option>
          <option value="-1" >-1</option>
          <option value="0" >0</option>
          <option value="1" >1</option>
          <option value="2" >2</option>
          <option value="3" >3</option>
          <option value="4" >4</option>
          <option value="5" >5</option>
          <option value="6" >6</option>
          <option value="7" >7</option>
          <option value="8" >8</option>
          <option value="9" >9</option>
          <option value="10" >10</option>
          <option value="11" >11</option>
          <option value="12" >12</option>
          <option value="13" >13</option>
          <option value="14" >14</option>
          <option value="15" >15</option>
          <option value="16" >16</option>
          <option value="17" >17</option>
          <option value="18" >18</option>
          <option value="19" >19</option>
          <option value="20" >20</option>
          <option value="21" >21</option>
          <option value="22" >22</option>
          <option value="23" >23</option>
          <option value="24" >24</option>
          <option value="25" >25</option>
          </select></td>
        </tr>
        <tr>
          <td><input type="radio" name="desifrovani" value="False" checked="checked" /> Šif-
rování </td>
        </tr>
        <tr>
          <td><input type="radio" name="desifrovani" value="True" /> Dešifrování </td>
        </tr>
        <tr>
          <td><input type="submit" name="odeslat" value="Odeslat" /></td>
        </tr>
</table>
```

```
</form>

<msp:evaluate>
If[ ValueQ[ $$odeslat ],

FceCaesear[data_, posuna_, desifrovani_] :=
Module[{},

   vystup={};
   abc = CharacterRange["A", "Z"];

   abc2 =
    If[desifrovani == "True",
     RotateLeft[CharacterRange["A", "Z"], posuna-(posuna*2)],
     RotateLeft[CharacterRange["A", "Z"], posuna]
    ];

   zdrojovytext = data;
   zdrojovytext = ToUpperCase[zdrojovytext];
   zdrojovytext =
    StringReplace[
     zdrojovytext, {"Á" -> "A", "Č" -> "C", "Ď" -> "D", "Ě" -> "E",
      "É" -> "E", "Í" -> "I", "Ó" -> "O", "Ú" -> "U", "Ů" -> "U",
      "Ň" -> "N", "Ř" -> "R", "Š" -> "S", "Ť" -> "T", "Ý" -> "Y",
      "Ž" -> "Z", ";" -> "", "+" -> "", "=" -> "", "\.b4" -> "",
      "," -> "", "?" -> "", "<" -> "", "." -> "", ":" -> "",
      ">" -> "", "-" -> "", "_" -> "", "*" -> "", "§" -> "",
      "!" -> "", "ß" -> "", "¨" -> "", "'" -> "", "¤" -> "",
      "/" -> "", "\[Divide]" -> "", ")" -> "", "(" -> "",
      "\[Times]" -> "", "#" -> "", "&" -> "", "{" -> "", "}" -> "",
      "đ" -> "", "Đ" -> "", "[" -> "", "]" -> "", "ł" -> "",
      "Ł" -> "", "|" -> "", "\[Euro]" -> "", "@" -> "", " " -> "",
      "0" -> "", "1" -> "", "2" -> "", "3" -> "", "4" -> "",
      "5" -> "", "6" -> "", "7" -> "", "8" -> "", "9" -> "",
      "~" -> "", "$" -> "", "%" -> "", "^" -> "", "+" -> "",
      ";" -> "", "\[Degree]" -> "", "˘" -> ""}];
   zdrojovytext = Characters[zdrojovytext];
   delkazdrojovehotextu = Length[zdrojovytext];



   For[i = 1, i <= delkazdrojovehotextu, i++,
    pozice = Position[abc, zdrojovytext[[i]]];
    AppendTo[vystup, abc2[[pozice[[1,1]]]]];
   ];

     If[delkazdrojovehotextu >= 6,
```

```
       StringJoin[{Riffle[vystup, " ", 6]}],
       StringJoin[{Riffle[vystup, " ", delkazdrojovehotextu + 1]}]
      ]
   ];

      data = ToString[$$data];
      posuna = ToString[$$posun];
      posuna = ToExpression[posuna];
      desifrovani = $$desifrovani;

     Print["Vystup :",FceCaesear[data,posuna,desifrovani]]

     If[desifrovani=="False",typsifrovani="Data   byla   SIFROVANA",typsifrovani="Data   byla
DESIFROVANA"];
     Print["Data:",data, " ===== Posun: ",posuna, " ===== ",typsifrovani]

]
</msp:evaluate>

<msp:evaluate>
ColumnForm[ MSPGetPrintOutput[]]
</msp:evaluate>

</msp:allocateKernel>


  </body>
</html>
```

PŘÍLOHA P II: TRANSPOZIČNÍ ŠIFRA VE FORMÁTU JSP

```
<%@ page language="java"  %>
<%@ taglib uri="/webMathematica-taglib" prefix="msp" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
  <meta http-equiv="content-type" content="text/html; charset=windows-1250">

  <title>Transpozicni Šifra</title>
  </head>
  <body>

<msp:allocateKernel>

<msp:evaluate>
Needs[ "MSP`" ];
SetSecurity[];
</msp:evaluate>

<form action="latis-transpozice.jsp" method="post">
<table>
       <tr>
         <td>Zadejte data, která chcete nechat kódovat:</td>
       </tr>
       <tr>
         <td><input type="text" value="<msp:evaluate>MSPValue[ $$data,  "text ke kodovani"
]</msp:evaluate>" name="data" size="70" /></td>
       </tr>
       <tr>
         <td><input type="radio" name="desifrovani" value="False" checked="checked" /> Šif-
rování </td>
       </tr>
       <tr>
         <td><input type="radio" name="desifrovani" value="True" /> Dešifrování </td>
       </tr>
       <tr>
         <td>Heslo 1:</td>
       </tr>
       <tr>
         <td><input   type="text"   value="<msp:evaluate>MSPValue[   $$heslo1,      "heslo1"
]</msp:evaluate>" name="heslo1" size="40" /></td>
       </tr>
       <tr>
         <td>Heslo 2:</td>
       </tr>
       <tr>
```

```
        <td><input   type="text"   value="<msp:evaluate>MSPValue[   $$heslo2,      "heslo2"
]</msp:evaluate>" name="heslo2" size="40" /></td>
       </tr>

       <tr>
        <td><input type="submit" name="odeslat" value="Odeslat" /></td>
       </tr>
</table>
</form>

<msp:evaluate>
If[ ValueQ[ $$odeslat ],

DvojitaTranspozice[data_, heslo_, heslo2_, desifrovani_] :=
  Module[{delkaDat, delkaHesla, delkaHesla2, zaokrouhleni,
    zaokrouhleni2, text1, zbytek, zbytek2, znak, znak2, matice,
    matice1, matice2, delkaZasifrovanehoTextu, osetrenyText,
    osetreneHeslo, osetreneHeslo2, heslo2Upravene, serazeneHeslo2,
    testvystup},

   delkaDat = StringLength[data];
   delkaHesla = StringLength[heslo];
   delkaHesla2 = StringLength[heslo2];
   If[delkaHesla2 > delkaHesla,
    Print["Druhé heslo je delší než první. Opravte to!"]];
   If[delkaHesla > delkaDat,
    Print["Heslo je delší než délka dat. Opravte to!"]];

   zaokrouhleni = Ceiling[delkaDat/delkaHesla];
   zaokrouhleni2 = Ceiling[delkaDat/delkaHesla2];
   zbytek = Mod[delkaDat, delkaHesla];
   zbytek2 = Mod[delkaDat, delkaHesla2];
   znak = delkaHesla - zbytek;
   znak2 = delkaHesla2 - zbytek2;

   osetrenyText = ToUpperCase[data];
   osetrenyText =
    StringReplace[
     osetrenyText, {"Á" -> "A", "Č" -> "C", "Ď" -> "D", "Ě" -> "E",
      "É" -> "E", "Í" -> "I", "Ó" -> "O", "Ú" -> "U", "Ů" -> "U",
      "Ň" -> "N", "Ř" -> "R", "Š" -> "S", "Ť" -> "T", "Ý" -> "Y",
      "Ž" -> "Z", ";" -> "X", "+" -> "X", "=" -> "X", "\.b4" -> "X",
      "," -> "X", "?" -> "X", "<" -> "X", "." -> "X", ":" -> "X",
      ">" -> "X", "-" -> "X", "_" -> "X", "*" -> "X", "§" -> "X",
      "!" -> "X", "ß" -> "X", "¨" -> "X", "'" -> "X", "¤" -> "X",
      "/" -> "X", "\[Divide]" -> "X", ")" -> "X", "(" -> "X",
      "\[Times]" -> "X", "#" -> "X", "&" -> "X", "{" -> "",
      "}" -> "X", "đ" -> "X", "Đ" -> "X", "[" -> "", "]" -> "X",
```

```
   "ł" -> "X", "Ł" -> "X", "|" -> "X", "\[Euro]" -> "X",
   "@" -> "X", " " -> "X", "0" -> "X", "1" -> "X", "2" -> "X",
   "3" -> "X", "4" -> "X", "5" -> "X", "6" -> "X", "7" -> "X",
   "8" -> "X", "9" -> "X", "~" -> "X", "$" -> "X", "%" -> "X",
   "^" -> "X", "+" -> "X", ";" -> "X", "\[Degree]" -> "X",
   "˘" -> "X"}];
text1 = osetrenyText;

osetreneHeslo = ToUpperCase[heslo];
osetreneHeslo =
 StringReplace[
  osetreneHeslo, {"Á" -> "A", "Č" -> "C", "Ď" -> "D", "Ě" -> "E",
   "É" -> "E", "Í" -> "I", "Ó" -> "O", "Ú" -> "U", "Ů" -> "U",
   "Ň" -> "N", "Ř" -> "R", "Š" -> "S", "Ť" -> "T", "Ý" -> "Y",
   "Ž" -> "Z", ";" -> "X", "+" -> "X", "=" -> "X", "\.b4" -> "X",
   "," -> "X", "?" -> "X", "<" -> "X", "." -> "X", ":" -> "X",
   ">" -> "X", "-" -> "X", "_" -> "X", "*" -> "X", "§" -> "X",
   "!" -> "X", "ß" -> "X", "¨" -> "X", "'" -> "X", "¤" -> "X",
   "/" -> "X", "\[Divide]" -> "X", ")" -> "X", "(" -> "X",
   "\[Times]" -> "X", "#" -> "X", "&" -> "X", "{" -> "",
   "}" -> "X", "đ" -> "X", "Đ" -> "X", "[" -> "", "]" -> "X",
   "ł" -> "X", "Ł" -> "X", "|" -> "X", "\[Euro]" -> "X",
   "@" -> "X", " " -> "X", "0" -> "X", "1" -> "X", "2" -> "X",
   "3" -> "X", "4" -> "X", "5" -> "X", "6" -> "X", "7" -> "X",
   "8" -> "X", "9" -> "X", "~" -> "X", "$" -> "X", "%" -> "X",
   "^" -> "X", "+" -> "X", ";" -> "X", "\[Degree]" -> "X",
   "˘" -> "X"}];

osetreneHeslo2 = ToUpperCase[heslo2];
osetreneHeslo2 =
 StringReplace[
  osetreneHeslo2, {"Á" -> "A", "Č" -> "C", "Ď" -> "D", "Ě" -> "E",
   "É" -> "E", "Í" -> "I", "Ó" -> "O", "Ú" -> "U", "Ů" -> "U",
   "Ň" -> "N", "Ř" -> "R", "Š" -> "S", "Ť" -> "T", "Ý" -> "Y",
   "Ž" -> "Z", ";" -> "X", "+" -> "X", "=" -> "X", "\.b4" -> "X",
   "," -> "X", "?" -> "X", "<" -> "X", "." -> "X", ":" -> "X",
   ">" -> "X", "-" -> "X", "_" -> "X", "*" -> "X", "§" -> "X",
   "!" -> "X", "ß" -> "X", "¨" -> "X", "'" -> "X", "¤" -> "X",
   "/" -> "X", "\[Divide]" -> "X", ")" -> "X", "(" -> "X",
   "\[Times]" -> "X", "#" -> "X", "&" -> "X", "{" -> "",
   "}" -> "X", "đ" -> "X", "Đ" -> "X", "[" -> "", "]" -> "X",
   "ł" -> "X", "Ł" -> "X", "|" -> "X", "\[Euro]" -> "X",
   "@" -> "X", " " -> "X", "0" -> "X", "1" -> "X", "2" -> "X",
   "3" -> "X", "4" -> "X", "5" -> "X", "6" -> "X", "7" -> "X",
   "8" -> "X", "9" -> "X", "~" -> "X", "$" -> "X", "%" -> "X",
   "^" -> "X", "+" -> "X", ";" -> "X", "\[Degree]" -> "X",
   "˘" -> "X"}];
```

```
If[zbytek != 0, For[i = 1, i <= znak, text1 = text1 <> "X"; i++]];
If[zbytek2 != 0,
 For[i = 1, i <= znak2, heslo2Upravene = heslo2Upravene <> "X";
  i++]];

heslo2Upravene = Characters[osetreneHeslo2];
serazeneHeslo2 = Sort[heslo2Upravene];


If[zbytek != 0, For[i = 1, i <= znak, text1 = text1 <> "x"; i++]];



If[desifrovani == "True",
 matice =
  Table[StringTake[text1, {(i - 1)*zaokrouhleni + j}], {i, 1,
    delkaHesla}, {j, 1, zaokrouhleni}];
 matice1 = Transpose[matice];
 matice1 = Join[{Ordering[Characters[heslo]]}, matice1];
 matice1 = Transpose[matice1];
 serazeneHeslo2 // MatrixForm;
 matice2 = Sort[matice1, serazeneHeslo2];
 matice2 = Transpose[matice2];
 matice2 = Delete[matice2, 1];
 matice2 = Flatten[matice2];
 ,
 matice =
  Table[StringTake[text1, {(i - 1)*delkaHesla + j}], {i, 1,
    zaokrouhleni}, {j, 1, delkaHesla}];
 matice = Join[{Characters[osetreneHeslo]}, matice];
 matice1 = Transpose[matice];
 matice1 // MatrixForm;
 serazeneHeslo2 // MatrixForm;
 matice2 = Sort[matice1, serazeneHeslo2];
 matice2 // MatrixForm;
 matice2 = Transpose[matice2];
 matice2 = Delete[matice2, 1];
 matice2 = Transpose[matice2];
 matice2 = Flatten[matice2];
 ];

If[desifrovani == "False",
 matice2 = StringJoin[matice2];
 StringReplace[matice2, {"X" -> " "}]
 ,
 If[delkaDat >= 6, StringJoin[{Riffle[matice2, " ", 6]}],
```

```
     StringJoin[{Riffle[matice2, " ", delkaDat + 1]}]]]

   ];

      data = ToString[$$data];
      heslo = ToString[$$heslo1];
      heslo2 = ToString[$$heslo2];

      desifrovani = ToString[$$desifrovani];

     Print["Vystup :",DvojitaTranspozice[data, heslo, heslo2, desifrovani]]

     If[desifrovani=="False",typsifrovani="Data   byla   SIFROVANA",typsifrovani="Data   byla
DESIFROVANA"];
     Print["Data:",data];
     Print[" Heslo1:",heslo1];
     Print[" Heslo2:",heslo2];
     Print[typsifrovani];




]
</msp:evaluate>

<msp:evaluate>
ColumnForm[ MSPGetPrintOutput[]]
</msp:evaluate>

</msp:allocateKernel>


  </body>
</html>
```

PŘÍLOHA P III: ŠIFRA AUTOKLÁV VE FORMÁTU JSP

```
<%@ page language="java"  %>
<%@ taglib uri="/webMathematica-taglib" prefix="msp" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
  <meta http-equiv="content-type" content="text/html; charset=windows-1250">

  <title>Šifra Autoklav</title>
  </head>
  <body>

<msp:allocateKernel>

<msp:evaluate>
Needs[ "MSP`" ];
SetSecurity[];
</msp:evaluate>

<form action="latis-autoklav.jsp" method="post">
<table>
       <tr>
         <td>Zadejte data, která chcete nechat kódovat:</td>
       </tr>
       <tr>
         <td><input type="text" value="<msp:evaluate>MSPValue[ $$data,  "text ke kodovani"
]</msp:evaluate>" name="data" size="70" /></td>
       </tr>
       <tr>
         <td><input type="radio" name="desifrovani" value="False" checked="checked" /> Šif-
rování </td>
       </tr>
       <tr>
         <td><input type="radio" name="desifrovani" value="True" /> Dešifrování </td>
       </tr>
       <tr>
         <td>Heslo:</td>
       </tr>
       <tr>
         <td><input   type="text"   value="<msp:evaluate>MSPValue[   $$heslo1,      "heslo1"
]</msp:evaluate>" name="heslo1" size="40" /></td>
       </tr>


       <tr>
         <td><input type="submit" name="odeslat" value="Odeslat" /></td>
       </tr>
```

```
</table>
</form>

<msp:evaluate>
If[ ValueQ[ $$odeslat ],

Autoklav[data_, heslo_, desifrovani_] :=
  Module[{vstupniHeslo, vstupniHeslo2, text, zdrojovaData, abeceda,
    abeceda2, vienereTable, otevrenytext, delkaOtevrenehoTextu,
    maximalniDelkaOtevrenehoTextu, vystupniSifra, vystupniSifra2,
    finalniSifra, delkaDat, osetreneHeslo, osetreneHeslo2,
    osetreneHesloS, osetreneHesloD, osetrenyText, delkaHesla,
    abeceda2D, zdrojovaData2, otevrenytext2, osetrenyText2, heslo2,
    pom1, pom2, pom3, sloupec, radek, souradnicex, souradnicey, pom1D,
     pom2D, souradnicex2, souradnicey2, i},

   sloupec = {"z" , "y", "x", "w", "v", "u", "t", "s", "r", "q", "p",
     "o", "n", "m", "l", "k", "j", "i", "h", "g", "f", "e", "d", "c",
     "b", "a" };
   radek = {"a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k",
     "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x",
     "y", "z" };

   abeceda = CharacterRange["a", "z"];
   abeceda = RotateRight[abeceda, 1];

   vienereTable = Table[abeceda = RotateLeft[abeceda, 1], {26} ] ;

   vstupniHeslo = heslo;
   vstupniHeslo = ToLowerCase[vstupniHeslo];
   vstupniHeslo =
    StringReplace[
     vstupniHeslo, {"á" -> "a", "č" -> "c", "ď" | "Ď" -> "d",
      "ě" | "Ě" -> "e", "é" -> "e", "í" -> "i", "ň" | "Ň" -> "n",
      "ó" -> "o", "ř" | "Ř" -> "r", "š" -> "s", "ť" | "Ť" -> "t",
      "ú" -> "u", "ů" | "Ů" -> "u", "ý" -> "y", "ž" | "Ž" -> "z",
      ";" -> "", "+" -> "", "=" -> "", "\.b4" -> "", "," -> "",
      "?" -> "", "<" -> "", "." -> "", ":" -> "", ">" -> "",
      "-" -> "", "_" -> "", "*" -> "", "§" -> "", "!" -> "",
      "ß" -> "", "¨" -> "", "'" -> "", "¤" -> "", "/" -> "",
      "\[Divide]" -> "", ")" -> "", "(" -> "", "\[Times]" -> "",
      "#" -> "", "&" -> "", "{" -> "", "}" -> "", "đ" -> "",
      "Đ" -> "", "[" -> "", "]" -> "", "ł" -> "", "Ł" -> "",
      "|" -> "", "\[Euro]" -> "", "@" -> "", " " -> "", "0" -> "",
      "1" -> "", "2" -> "", "3" -> "", "4" -> "", "5" -> "",
      "6" -> "", "7" -> "", "8" -> "", "9" -> "", "~" -> "",
      "$" -> "", "%" -> "", "^" -> "", "+" -> "", ";" -> "",
```

```
   "\[Degree]" -> "", " " -> "", "˘" -> "", "\\" -> ""}];
osetreneHeslo = vstupniHeslo;
osetreneHeslo2 = vstupniHeslo;
vstupniHeslo2 = vstupniHeslo;

zdrojovaData = data;
zdrojovaData = ToLowerCase[zdrojovaData];
zdrojovaData =
 StringReplace[
  zdrojovaData, {"á" -> "a", "č" -> "c", "ď" | "Ď" -> "d",
   "ě" | "Ě" -> "e", "é" -> "e", "í" -> "i", "ň" | "Ň" -> "n",
   "ó" -> "o", "ř" | "Ř" -> "r", "š" -> "s", "ť" | "Ť" -> "t",
   "ú" -> "u", "ů" | "Ů" -> "u", "ý" -> "y", "ž" | "Ž" -> "z",
   ";" -> "", "+" -> "", "=" -> "", "\.b4" -> "", "," -> "",
   "?" -> "", "<" -> "", "." -> "", ":" -> "", ">" -> "",
   "-" -> "", "_" -> "", "*" -> "", "§" -> "", "!" -> "",
   "ß" -> "", "¨" -> "", "'" -> "", "¤" -> "", "/" -> "",
   "\[Divide]" -> "", ")" -> "", "(" -> "", "\[Times]" -> "",
   "#" -> "", "&" -> "", "{" -> "", "}" -> "", "đ" -> "",
   "Đ" -> "", "[" -> "", "]" -> "", "ł" -> "", "Ł" -> "",
   "|" -> "", "\[Euro]" -> "", "@" -> "", " " -> "", "0" -> "",
   "1" -> "", "2" -> "", "3" -> "", "4" -> "", "5" -> "",
   "6" -> "", "7" -> "", "8" -> "", "9" -> "", "~" -> "",
   "$" -> "", "%" -> "", "^" -> "", "+" -> "", ";" -> "",
   "\[Degree]" -> "", " " -> "", "˘" -> "", "\\" -> ""}];
osetrenyText = zdrojovaData;
osetrenyText2 = zdrojovaData;
zdrojovaData2 = zdrojovaData;



If[desifrovani == "False",
 osetreneHesloS = Characters[osetreneHeslo];
 delkaHesla = Length[osetreneHesloS];
 otevrenytext = Characters[zdrojovaData];
 delkaOtevrenehoTextu = Length[otevrenytext];

 maximalniDelkaOtevrenehoTextu = 0;
 maximalniDelkaOtevrenehoTextu = delkaOtevrenehoTextu + 1;

 vystupniSifra = {};
 For[i = 1, i < maximalniDelkaOtevrenehoTextu, i++,
  pom1 = Take[osetreneHesloS, {i}][[1]];
  pom2 = Take[otevrenytext, {i}][[1]];
  souradnicex = Position[sloupec, pom2][[1]];
  souradnicey = Position[radek, pom1][[1]];
  pom3 = vienereTable[[souradnicex, souradnicey]][[1]];
  AppendTo[vystupniSifra, pom3[[1]]];
```

```
     AppendTo[osetreneHesloS, pom3[[1]]];
     ];

    If[delkaOtevrenehoTextu >= 6,
     finalniSifra =
      ToUpperCase[StringJoin[Riffle[vystupniSifra, " ", 6]]],
     finalniSifra =
      ToUpperCase[
       StringJoin[
        Riffle[vystupniSifra, " ", delkaOtevrenehoTextu + 1]]]];
    finalniSifra
    ,
    osetreneHeslo2 = Characters[vstupniHeslo2];
    otevrenytext2 = Characters[zdrojovaData2];

    vstupniHeslo2 = StringJoin[vstupniHeslo2, zdrojovaData2];
    vstupniHeslo2 = Characters[vstupniHeslo2];
    zdrojovaData2 = Characters[ zdrojovaData2];

    vystupniSifra2 = {};
    For[i = 1, i < 26, i++,
     pom1D = Take[vstupniHeslo2, {i}];
     pom2D = Take[zdrojovaData2, {i}];
     souradnicex2 = Position[radek, pom1D[[1]]][[1]];
     abeceda2D =
      RotateLeft[CharacterRange["a", "z"], souradnicex2 - 1];
     souradnicey2 = Position[abeceda2D, pom2D[[1]]];
     pom2D = Take[sloupec, souradnicey2[[1]]][[1]];
     AppendTo[vystupniSifra2, pom2D];
     ];
    vystupniSifra2 = StringJoin[vystupniSifra2];
    vystupniSifra2
    ]
   ];


      data = ToString[$$data];
      heslo = ToString[$$heslo1];


      desifrovani = ToString[$$desifrovani];


     Print["Vystup :",Autoklav[data, heslo, desifrovani]]


     If[desifrovani=="False",typsifrovani="Data   byla   SIFROVANA",typsifrovani="Data   byla
DESIFROVANA"];
     Print["Data:",data];
     Print[" Heslo1:",heslo1];
     Print[typsifrovani];
```

```
]
</msp:evaluate>

<msp:evaluate>
ColumnForm[ MSPGetPrintOutput[]]
</msp:evaluate>

</msp:allocateKernel>


  </body>
</html>
```

PŘÍLOHA P IV: FRAKTÁLY V IFS FORMĚ VE FORMÁTU JSP

```
<%@ page language="java"  %>
<%@ taglib uri="/webMathematica-taglib" prefix="msp" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
  <meta http-equiv="content-type" content="text/html; charset=windows-1250">

  <title>Fraktály v IFS forme</title>
  </head>
  <body>

<msp:allocateKernel>

<msp:evaluate>
Needs[ "MSP`" ];
SetSecurity[];
</msp:evaluate>

<form action="latis-fraktaly.jsp" method="post">
<table>
       <tr>
         <td><input type="submit" name="odeslat" value="Odeslat" /></td>
       </tr>
</table>
</form>

<msp:evaluate>
If[ ValueQ[ $$odeslat ],

Needs["ProgrammingInMathematica`AffineMaps`"]
Needs["ProgrammingInMathematica`IFS`"]
ifs0 = IFS[{AffineMap[0, 0, 0.5, 0.5, 1, 1],
    AffineMap[-1/3, -1/3, 1, 1, 2, 2],
    AffineMap[-3/4, -3/4, 1.5, 1.5, 3, 3]}];
gr1 = Show[Graphics[
   {
    Rectangle[{-1.0, 0.8}, {-0.5, 0.6}],
    Rectangle[{-0.85, 0.6}, {-0.65, -0.1}],
    Rectangle[{-0.5, 0.6}, {0.0, 0.5}],
    Rectangle[{-0.5, 0.3}, {0.0, 0.2}],
    Rectangle[{-0.5, 0.0}, {0.0, -0.1}],
    Rectangle[{-0.5, 0.6}, {-0.4, -0.1}],
    Rectangle[{0.1, 0.6}, {0.2, -0.1}],
    Rectangle[{0.1, 0.6}, {0.5, 0.5}],
```

```
    Rectangle[{0.5, 0.6}, {0.6, 0.2}],
    Rectangle[{0.1, 0.2}, {0.5, 0.3}],
    Line[{{0.2, 0.2}, {0.6, -0.1}}],
    Rectangle[{0.7, 0.6}, {0.8, -0.1}],
    Rectangle[{0.9, 0.6}, {1.0, -0.1}],
    Rectangle[{1.1, 0.6}, {1.2, -0.1}],
    Rectangle[{0.7, 0.6}, {1.2, 0.5}],
    Rectangle[{1.3, 0.5}, {1.4, -0.1}],
    Disk[{1.35, 0.6}, {0.09, 0.1}],
    Rectangle[{1.5, 0.8}, {2.0, 0.6}],
    Rectangle[{1.65, 0.6}, {1.85, -0.1}]
    }
   ]
  , AspectRatio -> Automatic, Axes -> False, Frame -> True,
  GridLines -> Automatic, PlotRange -> All]

  MSPShow[
    Show[ifs0[gr1]]
  ]


]
</msp:evaluate>

<msp:evaluate>
ColumnForm[ MSPGetPrintOutput[]]
</msp:evaluate>

</msp:allocateKernel>
  </body>
</html>
```