

Web-slovník pojmů z automatického řízení

Web-oriented dictionary of
automatic control terms

Roman Kalivoda

Bakalářská práce
2005



Univerzita Tomáše Bati ve Zlíně
Fakulta technologická

Vložit oficiální zadání diplomové/bakalářské práce

Chtěl bych poděkovat vedoucímu bakalářské práce Ing. Františku Gazdošovi, Ph.D. za velmi užitečné rady a poznámky při vývoji tohoto projektu.

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

Ve Zlíně, 02. 06. 2005

.....

Roman Kalivoda

Abstrakt

Tato práce se zabývá návrhem webového rozhraní, které má sloužit jako oboustranný česko-anglický slovník vybraných pojmů z oblasti automatického řízení. Cílem práce je vytvořit stránky, které by usnadnily překlad nejčastěji používaných výrazů v tomto oboru.

Celý slovník je naprogramován ve skriptovacím jazyku PHP a data jsou spravována pomocí databázového systému MySQL. Stránky byly navrženy tak, aby práce se slovníkem byla rychlá, jednoduchá a příjemná.

The goal of this Bachelor thesis is to develop a Czech-English/English-Czech web-oriented dictionary of automatic control terms. It enables easy-to-use translation of the most frequent words and phrases from the field. The dictionary is programmed using the PHP scripting language and the data are administered by the database system MySQL. The resultant web-based system is designed for quick and user-friendly work enabling also easy update by means of the feedback from users.

OBSAH

ÚVOD.....	6
1 TEORETICKÁ ČÁST.....	7
1.1 HTML.....	7
1.1.1 Historie HTML.....	7
1.1.2 Základy HTML.....	7
1.1.3 CSS.....	17
1.1.4 HTML aneb statické stránky.....	17
1.1.5 JAVA skripty.....	17
1.2 PHP.....	18
1.2.1 Historie PHP.....	18
1.2.2 Základy PHP.....	20
1.3 MYSQL.....	25
1.3.1 K čemu slouží MYSQL.....	25
1.3.2 Připojení MYSQL.....	26
1.3.3 Spolupráce PHP a MySQL s webovým serverem.....	27
1.3.4 Základní příkazy SQL.....	27
2 PRAKTICKÁ ČÁST.....	33
2.1 SLOVNÍKY NA INTERNETU.....	33
2.2 OBECNÁ CHARAKTERISTIKA VYTVOŘENÉHO ROZHRANÍ.....	33
2.3 PRÁCE SE SLOVNÍKEM.....	34
2.3.1 Rychlé vyhledávání.....	34
2.3.2 Zobrazení celého slovníku.....	36
2.3.3 Administrace slov.....	38
2.3.4 Ostatní funkce.....	40
2.4 ZDROJOVÝ KÓD.....	40
2.4.1 Rozdělení do souborů.....	40
2.4.2 Princip rychlého vyhledávání.....	41
2.4.3 Tabulka <i>poslední přidaná slova</i>	45
2.4.4 Funkce pro zobrazení celého slovníku.....	46
2.4.5 Princip administrace slov.....	48
2.5 GRAFICKÝ VZHLED.....	52
2.5.1 Barvy a pozadí.....	52
2.5.2 Návrh šablony.....	52
2.6 VÝVOJ PROJEKTU.....	53
2.6.1 Vývojové prostředky.....	53
2.6.2 Testování na Internetu.....	53
ZÁVĚR.....	54
SEZNAM POUŽITÝCH ZDROJŮ.....	55
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	56
SEZNAM OBRÁZKŮ.....	57
SEZNAM TABULEK.....	58

ÚVOD

Dnešní moderní život si asi těžko dokážeme představit bez výpočetní techniky, která nám každý den usnadňuje práci a komunikaci mezi lidmi. Nejen že se každý setkává s počítačem v zaměstnání a ve škole, ale osobní počítač se stal téměř pro všechny samozřejmostí i v domácnosti. Ale samotný počítač by pro nás neměl až takový význam, kdyby nebyl připojen k celosvětové počítačové síti-Internet. Internet je jedním z nejvýznamnějších prvků při práci s počítačem. Můžeme zde najít nepřeberné množství informací a zábavy. Potřebná data můžeme vyhledat velmi jednoduše a rychle.

Významnou součástí Internetu jsou webové stránky a elektronická pošta. Každá firma, škola, organizace nebo běžný uživatel si může vytvořit své webové rozhraní, které pak bude volně přístupné pro všechny uživatele připojené k Internetu. Pro školství je dnes Internet velmi důležitou součástí, protože je studentům poskytován zdarma. Mohou zde najít veškeré studijní podklady a materiály potřebné k výuce. Nedílnou součástí je také školní systém. Ten usnadňuje studentům vyřizovat administrativní záležitosti odkudkoliv, kde mají přístup na Internet. Jak můžeme vidět, význam Internetu v dnešní době je opravdu obrovský.

Tato práce se zabývá návrhem webového rozhraní, které bude sloužit jako oboustranný česko-anglický slovník vybraných pojmů z oblasti automatického řízení. K vytvoření tohoto rozhraní byl použit programovací jazyk PHP ve spolupráci s databází MySQL.

Cílem práce je vytvořit takový slovník, který usnadňuje překlad pojmů z oboru automatického řízení z českého jazyka do anglického a naopak. Databáze obsahuje ty nejpoužívanější vybrané pojmy z dané oblasti. Slovník nabízí také prostor pro komunikaci uživatelů s administrátorem, či autorem, aby bylo možné později přidávat pojmy, které uživatel ve slovníku nenalezl. Tato spoluúčast uživatelů na dalším rozvoji databáze je velmi důležitá. Součástí překladu je u některých slov také ukázka užití pojmu v anglické větě.

Jádro tohoto webového rozhraní bylo navrženo tak, aby vyhledávání pojmů bylo jednoduché, přehledné a efektivní.

1 TEORETICKÁ ČÁST

V této části jsou popsány programovací nástroje použité pro vytvoření požadovaného webového rozhraní. Jedná se o jazyky HTML, PHP a SQL.

1.1 HTML

1.1.1 Historie HTML

HTML (HyperText Mark-up Language) je jazyk pro tvorbu webového rozhraní, který vytvořil na počátku devadesátých let Tim Berners-Lee. HTML bylo přísně založeno na mezinárodně schválené metodě pro formování textu do strukturálních jednotek jako jsou odstavce, nadpisy, položky seznamu atd., aby mohlo být implementováno na jakémkoliv počítači. Hlavní myšlenkou bylo, že jazyk bude nezávislý na formátoru (prohlížeč, nebo jiný zobrazovací systém), který ve skutečnosti zobrazuje text na obrazovce. Vynález HTML samozřejmě vedl ke vzniku prvních webových prohlížečů, jako byl Mosaic, nebo Lynx (určené pro DOS). V roce 1994 vznikly nové specifikace HTML2 a HTML+, na které mělo později navázat HTML3. Byl také vytvořen prohlížeč Netscape. V roce 1995 byl přijat koncept HTML3, který se mimochodem vypořádal s tabulkovou organizací informací v HTML. Byl také uvolněn prohlížeč Microsoft internet explorer. Během téhož roku se vynořily všechny druhy nových HTML značek. Například atribut BGCOLOR značky BODY a FONT FACE, které ovládají stylistické aspekty dokumentu. Koncept HTML3 přechází v HTML3.2. Vzniká možnost rozdělení obrazovky na rámce. Tento návrh byl implementován do prohlížeče Netscape's Navigator. Vznikají také kaskádové styly (CSS), které koncem roku získávají svoji podobu. S nástupem HTML4 prodělal jazyk ještě pár změn až do podoby, jaký jej známe dnes [1].

1.1.2 Základy HTML

Tagem neboli značkou **<html>** začíná a končí celý dokument. Veškerý další obsah musí být uvnitř. Jedná se o značku nepovinnou, většina prohlížečů si ji domyslí. Pokud ale chceme mít soubory v souladu s normou, potom tag používáme. Tag *html* nemá žádné atributy.

Tag **<head>** označuje hlavičku dokumentu, která se nezobrazuje. Obsahuje nepovinně další tagy (*title*, *meta*, *link*, *style*, *script* aj.). Pokud v hlavičce použijeme prostý text,

v některých prohlížečích se zobrazí na začátku stránky! Tag *head* nemá rovněž žádné atributy.

Tag **<body>** označuje tělo dokumentu. Obsahuje veškerý zobrazovaný obsah stránky. Veškerý text, který zde napíšeme bude zobrazen prohlížečem, pokud nebude uveden jako poznámka [2].

Nyní uvedu pár základních atributů tagu *body*:

Tabulka I. Základní atributy tagu *body*

Atribut tagu <body>	Význam	Hodnoty	CSS náhrada pro tag body
bbgcolor	barva pozadí	<u>barva</u>	background-color
background	obrázek pozadí	<u>URL</u> obrázku	background-image
text	barva popředí (textu)	<u>barva</u>	color
link	barva nenavštívených odkazů	<u>barva</u>	a:link {color: ...}
vlink	barva navštívených odkazů	<u>barva</u>	a:visited {color: ...}
alink	barva klikaných odkazů (a pod tabulátorem)	<u>barva</u>	a:active {color: ...}

Příklad:

```
<body background="obrazek.gif" bgcolor="yellow" text="blue" link="green"
vlink="olive" alink="#FF0000">
```

Zadávání barev pomocí atributů tagu *body* je zastaralé. Doporučuje se barvit dokument pomocí CSS stylů, např.:

```
<body style="color: blue; background-image: URL('obrazek.gif')">
```

nebo ještě lépe v hlavičce dokumentu pomocí tagu *<style>*, jak je uvedeno na další stránce:

```
<style>
body {color: blue; background-color: yellow; background-image:
URL('obrazek.gif')}
a:link {color: green;}
a:visited {color: olive;}
a:active {color: #FF0000}
</style>
```

Posledním základním tagem jsou značky `<!--` a `-->`, které označují poznámku. Ta nebude zobrazena v prohlížeči. Značka `<!--` je tedy začátek poznámky a `-->` její konec.

Klasický HTML soubor tedy může vypadat takto:

```
<html>
  <head>
    obsah hlavičky
  </head>

  <body>
    obsah těla
  </body>
</html>
```

Nejpoužívanější příkazy:

Odkazy

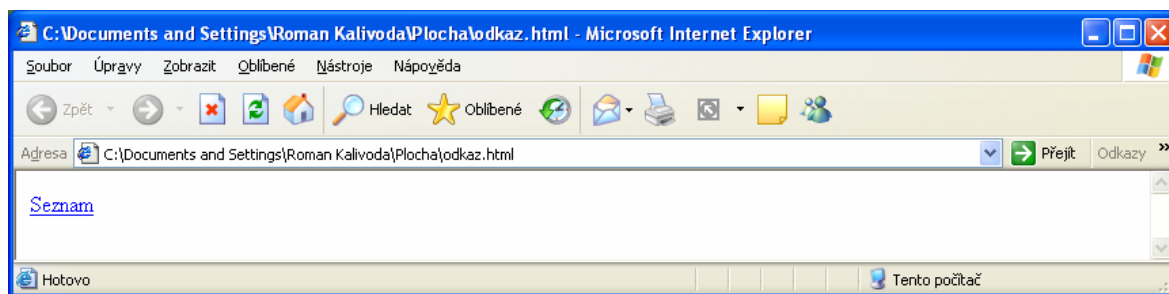
Odkaz na jiný web se provádí příkazem `<a href="požadovaná stránka">odkaz</a>`

Příklad:

```
<HTML>
  <BODY>
    <a href="http://www.seznam.cz">Seznam</a>

  </BODY>
</HTML>
```

Zobrazí se to takto:



Obr. 1. Odkaz

Odkaz Vás nemusí přesměrovat pouze na jinou webovou stránku, ale také na jiný soubor do stejného adresáře bez `http://`, jedná se o relativní adresování:

Příklad:

```
<a href="stranka.html">Odkaz na stranka</a>
```

Odkaz na soubor `stranka.html` o úroveň výš (dvě tečky):

Příklad:

```
<a href="../stranka.html">Odkaz do vyššího adresáře</a>
```

Můžeme ale také zadat přímo celou cestu k požadované stránce.

Příklad:

```
<a href="C:\pokus\stranka.html">Odkaz do zadaného adresáře</a>
```

Otevření odkazu do nového okna:

```
<a href="stranka.html" target="_blank">odkaz do nového okna</a>
```

nebo

```
<a href="http://www.seznam.cz" target="_blank">odkaz na Seznam do nového  
okna</a>
```

Záložky

Záložky používáme zejména při obsáhlém textu, kdy se chceme rychle přemístit na požadované místo v článku.

Příklad:

záložka někde v dokumentu

```
<a name="jmeno_zalozky"></a>
```

a takto se na ni po kliknutí na odkaz odrolujeme

```
<a href="#jmeno_zalozky">odrolovat</a>
```

Styly

Tag <style> patří mezi tagy <head> a </head>:

Příklad:

text černý, barva pozadí bílá, obrázek na pozadí

```
<style type="text/css">
body {color: black; background-color: white; background-image:
url("pozadi.gif");}
</style>
```

Další příklady:

nenavštívený odkaz, navštívený odkaz:

```
<style type="text/css">
a:link{color:blue}
a:visited {color: navy;}
</style>
```

Celý dokument napsán písmem *Arial*:

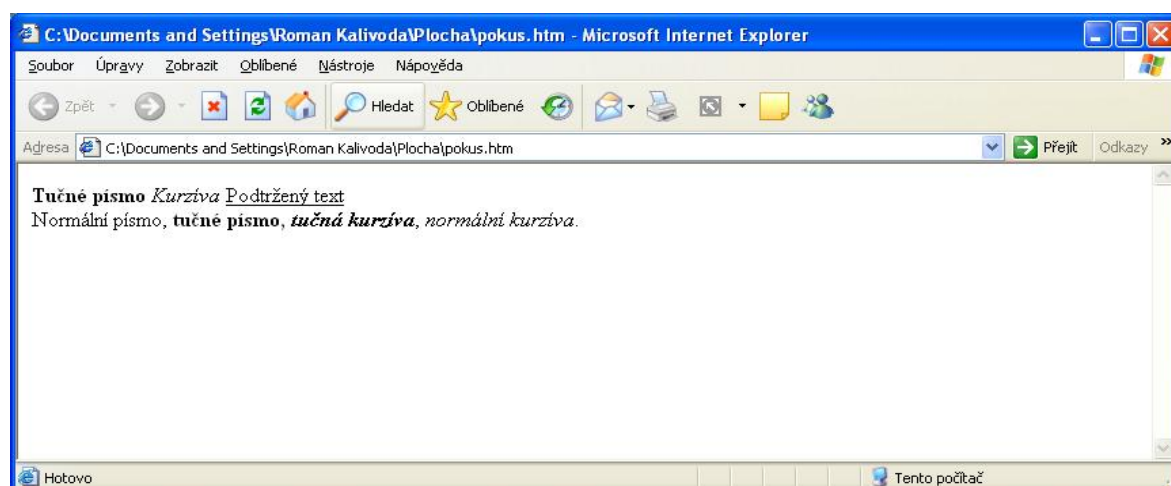
```
<style type="text/css">
body {font-family: Arial, sans-serif;}
</style>
```

Tučné a podtržené písmo, kurzíva

Příklad:

```
<HTML>
  <BODY>
    <b>Tučné písmo</b>
    <i>Kurzíva</i>
    <u> Podtržený text</u><br>
    Normální písmo, <b>tučné písmo, <i>tučná kurzíva</i></b>, <i>normální
    kurzíva</i>.
  </BODY>
</HTML>
```

Ukázka:



Obr. 2. Druhy písma

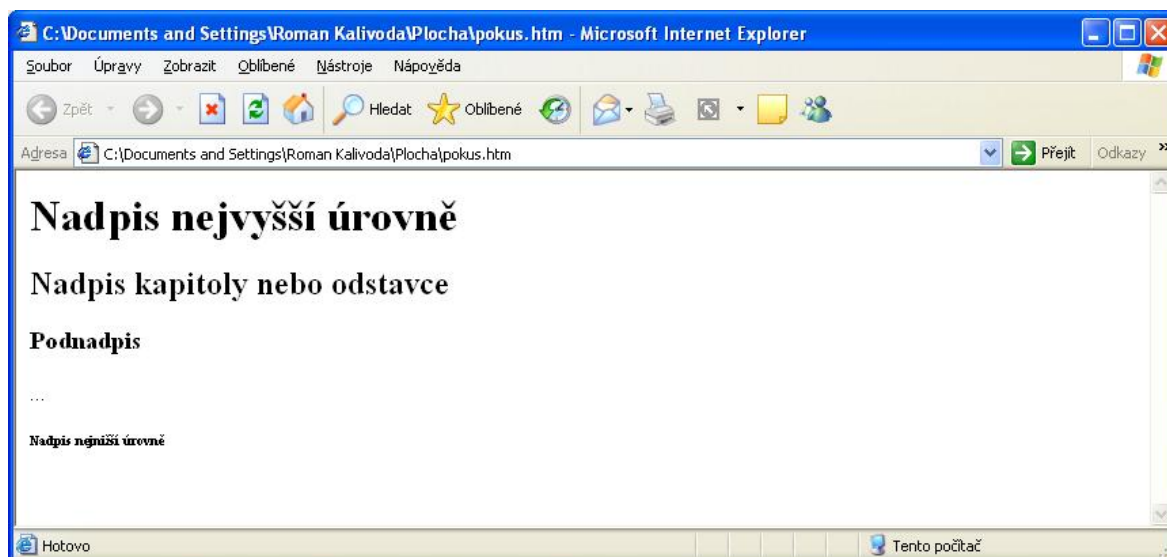
Nadpisy

Pro velikost písma nadpisů používáme příkazy <h1> až <h6>, přičemž <h1> je nadpis nejvyšší úrovně a <h6> naopak nejnižší.

Příklad:

```
<h1>Nadpis nejvyšší úrovně</h1>
<h2>Nadpis kapitoly nebo odstavce</h2>
<h3>Podnadpis</h3>
...
<h6>Nadpis nejnižší úrovně</h6>
```

Ukázka:



Obr. 3. Nadpisy

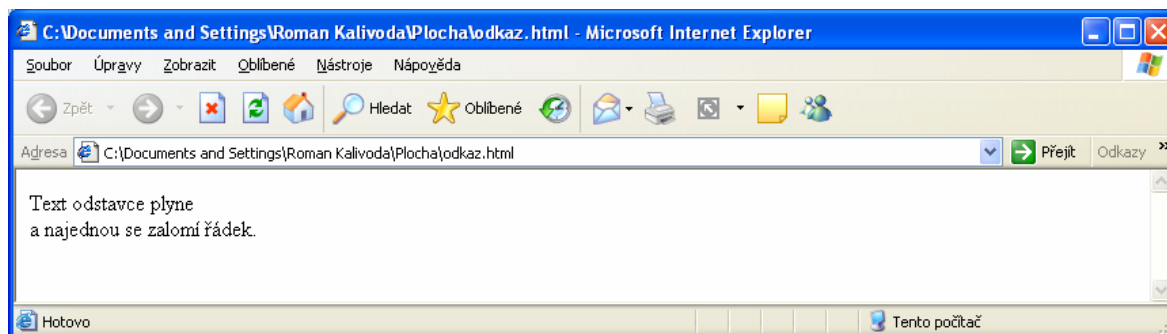
Odstavce a zalomení řádku

Tag <p> značí odstavec. Zalamuje řádky podle šířky prostoru, který má (nejčastěji okna). Je-li potřeba uvnitř odstavce zalomit řádek, použije se tag
, v normálních textech k tomu ale není důvod:

Příklad:

```
<p>Text odstavce plyne<br> a najednou se zalomí řádek.</p>
```

Ukázka:



Obr. 4. Zalomení řádku

Vložení obrázku

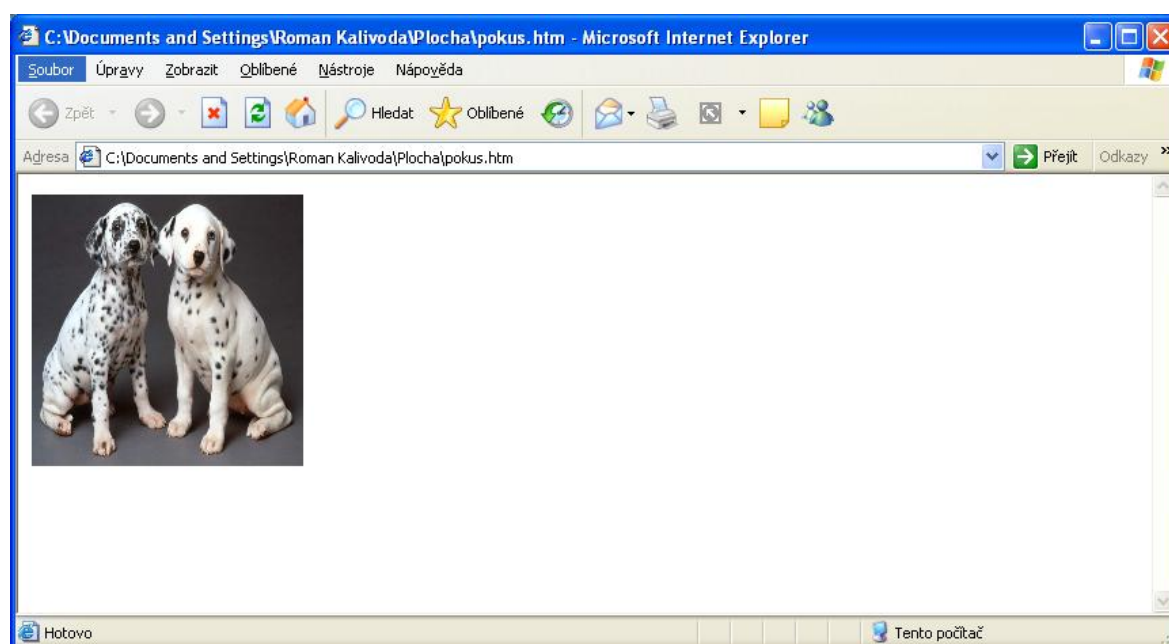
Příklady:

vložíme obrázek se zadanou šířkou a výškou ze stejného adresáře:

```

```

Ukázka:



Obr. 5. Vložení obrázku

Obrázek z webu (absolutní adresa, začíná http://):

```

```

Obrázek fungující jako odkaz:

```
<a href="cilova_stranka.html"></a>
```

Tabulky

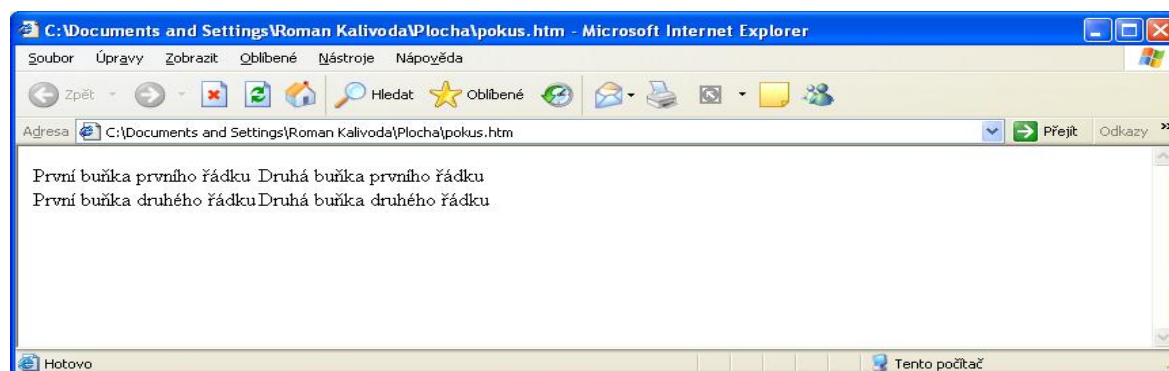
Příklady:

Nejjednodušší tabulka 2x2 bez ohraničení:

```
<table cellpadding="0">

<tr>
    <td>První buňka prvního řádku</td>
    <td>Druhá buňka prvního řádku</td>
</tr>
<tr>
    <td>První buňka druhého řádku</td>
    <td>Druhá buňka druhého řádku</td>
</tr>
</table>
```

Tabulka bude vypadat takto:



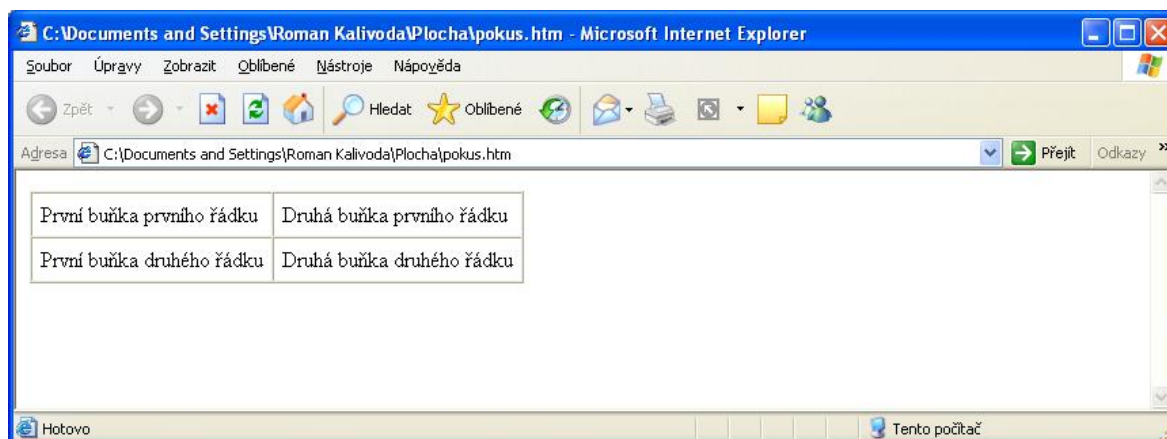
Obr. 6. Jednoduchá tabulka

Zapnutí rámečku (border) a vnitřních okrajů buněk (cellpadding):

```
<table border="1" cellpadding="6" cellspacing="0">

<tr>
    <td>První buňka prvního řádku</td>
    <td>Druhá buňka prvního řádku</td>
</tr>
<tr>
    <td>První buňka druhého řádku</td>
    <td>Druhá buňka druhého řádku</td>
</tr>
</table>
```

Tabulka s rámečkem:



Obr. 7. Tabulka s ohraničením

Poznámky

Když si chceme napsat do zdrojového kódu poznámku pouze pro sebe a nechceme, aby byla zobrazena prohlížečem, používáme vkládání poznámek, nebo komentářů. K tomu slouží následující znaky:

<!-- značí začátek poznámky

--> konec poznámky [3].

V praxi to vypadá následovně:

```
<HTML>

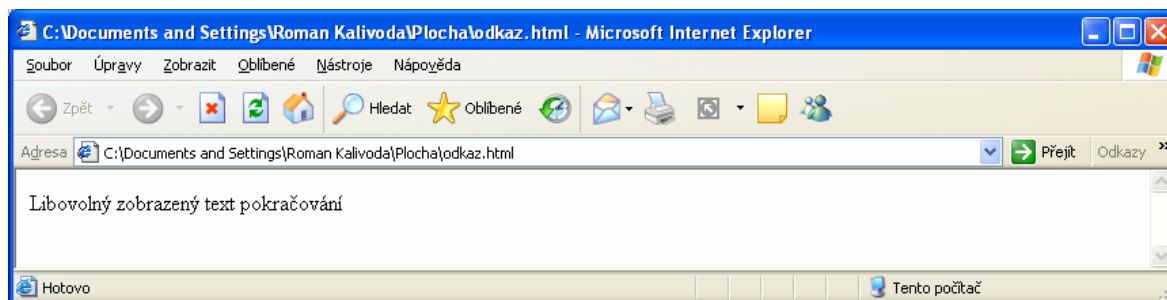
  <BODY>

Libovolný zobrazený text<!--poznámka  --> pokračování

  </BODY>

</HTML>
```

Prohlížeč zobrazí:



Obr. 8. Poznámka

1.1.3 CSS

CSS neboli kaskádové styly nám usnadňují nastavení základního vzhledu stránky. Například barvu a styl písma, pozadí atd. Toto už bylo nastíněno v předchozí kapitole. Nastavení můžeme provést přímo v každém zdrojovém kódu stránky, nebo ho ukládáme přímo do souboru s příponou *css*.

Takto připojíme *soubor.css* k souboru (vkládá se obvykle do každé stránky mezi `<head>` a `</head>`):

```
<link rel="stylesheet" href="soubor.css" type="text/css">
```

Předpokládáme, že se *soubor.css* nachází v tomtéž adresáři, jako je stránka.

1.1.4 HTML aneb statické stránky

Pokud si budeme chtít vytvořit vlastní webové stránky, máme celou řadu možností. Prvním impulsem takového běžného laického zájemce bývá tvorba webové stránky jako statické textové stránky. To znamená, že se každá stránka napíše podobně jako textový dokument. Je celkem jedno, zda použijeme nějaký vizuální editor typu Front Page, anebo se speciálně naučíme jazyk HTML a budeme psát stránky přímo v HTML. Podstatné je, že naše stránky budou statické.

Statické stránky nám zobrazí jen to, co jsme napsali a víc ani čárku. Pokud chceme třeba na každou stránku dopsat na konec naše jméno a adresu, nezbyde nám nic jiného, než ručně nakopírovat tyto údaje na každou stránku. Pokud těch stránek máme hodně, jedná se o velké množství ruční práce. Další náš požadavek třeba může být zobrazit ve stránce dnešní datum. To už je pomocí statických stránek prakticky nemožné, nechceme-li každý den přepisovat datum ručně. A právě proto chytří lidé vymysleli dynamické stránky viz. kapitola PHP.

1.1.5 JAVA skripty

Změnu stránky bez nutnosti obnovení jejího obsahu docílíme skripty na straně klienta (tedy uživatele), například pomocí Javascriptu. Výhoda těchto programovacích jazyků spočívá tedy v tom, že není třeba pro změnu stránky neustále obnovovat obsah stránky. Jejich veliké omezení však spočívá nejen v možnostech, které nabízí, ale také v jejich

(ne)bezpečnosti. Protože skripty se vykonávají přímo na počítači uživatele, mohl by potencionální útočník bez problémů vykonat nebezpečný kód. Z tohoto důvodu jsou např. u Javascriptu vypnuty funkce, které by přímo pracovaly se soubory na harddisku. Díky tomu se např. Javascript využívá spíše pro okamžitou úpravu CSS kódu či jako „doplněk“ při vytváření webové grafiky. Pro „opravdové“ funkce internetových stránek se ale používají skripty na straně serveru - PHP [4].

1.2 PHP

1.2.1 Historie PHP

PHP je objektově orientovaný programovací jazyk, svojí strukturou velmi podobný jazyku C++. Znalost objektově orientovaného programování (OOP) tedy může být při práci v PHP výhodou, není však nutnou podmínkou. PHP patří mezi jazyky, kde například není nutné předem definovat typ proměnných, navíc jakákoliv proměnná může kdykoliv změnit svůj typ. Jednoduše řečeno, co se týče psaní kódu, z PHP při psaní skriptů „sálá“ určitá volnost a neomezenost.

PHP urazilo v posledních několika málo letech dlouhou cestu. Růst v jeden z nejprominentnějších jazyků ovládajících Web nebyl snadný.

PHP je nástupcem staršího produktu, nazvaného PHP/FI. PHP/FI vytvořil Rasmus Lerdorf v roce 1995, na počátku jako jednoduchou sadu skriptů v jazyce Perl pro zpracování záznamů o přístupech k jeho webu. Tuto sadu nazval 'Personal Home Page Tools'. Protože byla třeba větší funkčnost, napsal Rasmus mnohem rozsáhlejší implementaci v C, která byla schopna komunikovat s databázemi a umožňovala uživatelům vyvíjet jednoduché dynamické aplikace pro Web. Rasmus se rozhodl uvolnit zdrojový kód PHP/FI pro všechny, takže kdokoliv ho může používat, stejně jako opravovat chyby a vylepšovat kód.

PHP/FI, což znamená Personal Home Page / Forms Interpreter, obsahovalo některé ze základních funkcí PHP, jak ho známe dnes. Mělo proměnné perlovského typu, automatickou interpretaci formulářových proměnných a syntaxi vloženou do HTML. Syntaxe samotná byla podobná jazyku Perl, přestože mnohem omezenější, jednodušší a v něčem nekonzistentní.

V roce 1997 se PHP/FI 2.0, druhá implementace psaná v C, stala kultovní záležitostí pro (odhadem) tisíce uživatelů po celém světě, a s přibližně 50.000 doménami oznamujícími nainstalované PHP/FI, což čítalo zhruba 1 % všech domén na Internetu. I když do projektu začalo svými kusy kódu přispívat více lidí, stále to byl velký projekt jednoho muže.

PHP/FI 2.0 bylo oficiálně uvolněno až v listopadu 1997, poté co strávilo většinu svého života v beta-verzích. Krátce nato bylo následováno první alfa-verzí PHP 3.0.

PHP 3

PHP 3.0 byla první verze, která se velmi blížila takovému PHP, jak ho známe dnes. Vytvořili ho Andi Gutmans a Zeev Suraski v roce 1997 jako kompletně přepsaný celek, poté co shledali PHP/FI 2.0 výrazně „poddimezované“ pro vývoj svých aplikací pro e-komerci. Ve snaze spolupracovat a zahájit budování nad existující uživatelskou základnou PHP/FI, rozhodli se Andi, Rasmus a Zeev pracovat společně a prohlásit PHP 3.0 za oficiálního nástupce PHP/FI 2.0, a vývoj PHP/FI 2.0 byl v podstatě zastaven.

Jednou z nejsilnějších zbraní PHP 3.0 byly jeho obrovské možnosti rozšíření. K poskytnutí pevné infrastruktury pro mnoho různých databází, protokolů a API koncovým uživatelům, přilákaly možnosti rozšíření PHP 3.0 také tucty vývojářů, kteří se připojili a vytvořili nové rozšiřující moduly. Toto byl nesporně klíč k obrovskému úspěchu PHP 3.0. Jiným klíčovým prvkem v PHP 3.0 byla podpora objektově orientované syntaxe a mnohem silnější a konzistentnější syntaxe jazyka.

Nový jazyk byl uvolněn pod novým názvem, který odstranil implikaci omezeného osobního použití, kterou neslo označení PHP/FI 2.0. Byl nazván pouze 'PHP', což je rekurzivní akronym - PHP: Hypertext Preprocessor.

Na konci roku 1998 vyrostlo PHP do rozsahu instalací v řádu (odhadem) desítek tisíc uživatelů a stovek tisíc Webů. V době svého vrcholu bylo PHP 3.0 instalováno na přibližně 10% všech WWW serverů na Internetu.

PHP 3.0 bylo oficiálně uvolněno v červnu 1998, poté co strávilo cca 9 měsíců ve veřejném testování.

PHP 4

V zimě roku 1998, krátce po oficiálním uvolnění PHP 3.0, začali Andi Gutmans a Zeev Suraski pracovat na přepsání jádra PHP. Cílem návrhu bylo zvýšit výkon pro složité aplikace a zlepšit modularitu kódové báze PHP. Takové aplikace byly schopny pracovat s PHP 3.0 (díky novým možnostem a podpoře široké škály databází a API od jiných tvůrců), ale PHP 3.0 nebylo navrženo pro efektivní práci tak náročných aplikací.

Nový „engine“ (jádro), nazvaný 'Zend Engine' (sestaven z jejich křestních jmen, Zeev a Andi), úspěšně splnil cíle návrhu a byl uveden v polovině roku 1999. PHP 4.0, založené na tomto enginu a doplněné širokou škálou nových prvků, bylo oficiálně uvolněno v květnu 2000, necelé dva roky po svém předchůdci, PHP 3.0. K podstatně zvýšenému výkonu této verze, přidává PHP 4.0 další klíčové prvky, jako je podpora pro mnoho WWW serverů, HTTP sessions, buffering výstupu, bezpečnější způsoby zpracování vstupů uživatele a mnoho nových jazykových konstrukcí.

PHP 4 je momentálně poslední uvolněnou verzí PHP. Již byla započata práce na modifikaci a vylepšení jádra Zend Engine k integraci prvků, které byly navrženy pro PHP 5.0.

Dnes používají PHP (odhadem) stovky tisíc vývojářů a nainstalované PHP hlásí několik milionů serverů - tj. přes 20 % domén na Internetu [5].

PHP 5

V červnu 2003 byla oficiálně uvolněna beta-verze PHP 5. Největší změna je v objektovém modelu, PHP se přibližuje ostatním jazykům podporujícím OOP. Snazší je také obsluha chyb [4].

1.2.2 Základy PHP

PHP je technologie běžící na serveru. Typický PHP skript obsahuje jednak části normálního HTML kódu, a jednak části programového kódu. Když webový server obdrží požadavek na zpracování takového skriptu, vezme:

- části HTML kódu tak, jak jsou
- části PHP programového kódu provede
- výsledek zkombinuje a odešle prohlížeči

Tato filozofie fungování je nesmírně mocná. Server totiž může provést jednu nebo dokonce několik operací a výsledek poslat do prohlížeče jako obyčejnou HTML stránku. Celé to předvedu na malé ukázce. Následující kód je obyčejná HTML stránka. Když si ji prohlížeč vyžádá, uvidíme v něm jen řádek mezi značkami.

```
<HTML>

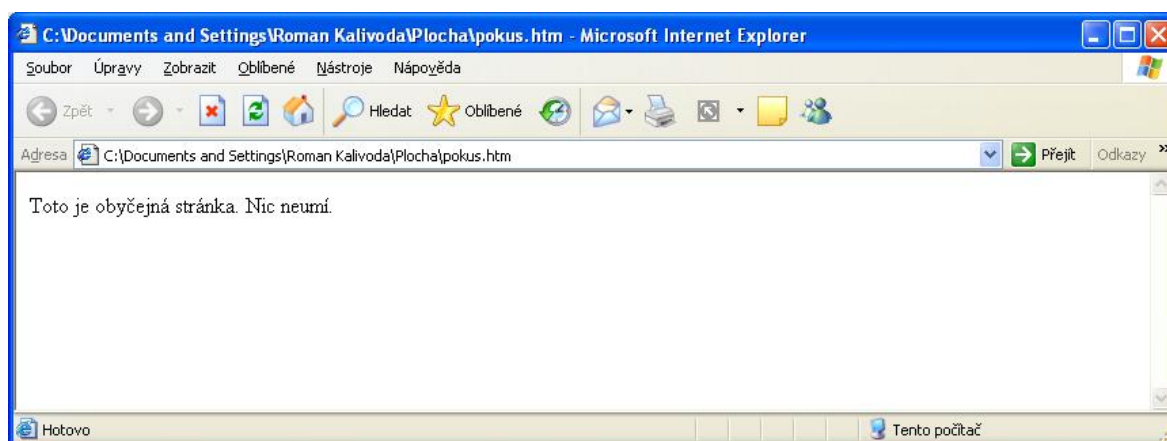
  <BODY>

    Toto je obyčejná stránka. Nic neumí.

  </BODY>

</HTML>
```

Zobrazí se takto:



Obr. 9. Obyčejná stránka

Následující kód je PHP skript. Skládá se jednak z HTML kódu, a jednak z PHP kódu. To, co vidíme mezi značkami `<? a ?>` (někdy se používá i `<?php a ?>`) je právě PHP kód. Příkaz *echo* vypisuje text na standardní výstup a příkaz *Date* vrací aktuální čas:

```
<HTML>

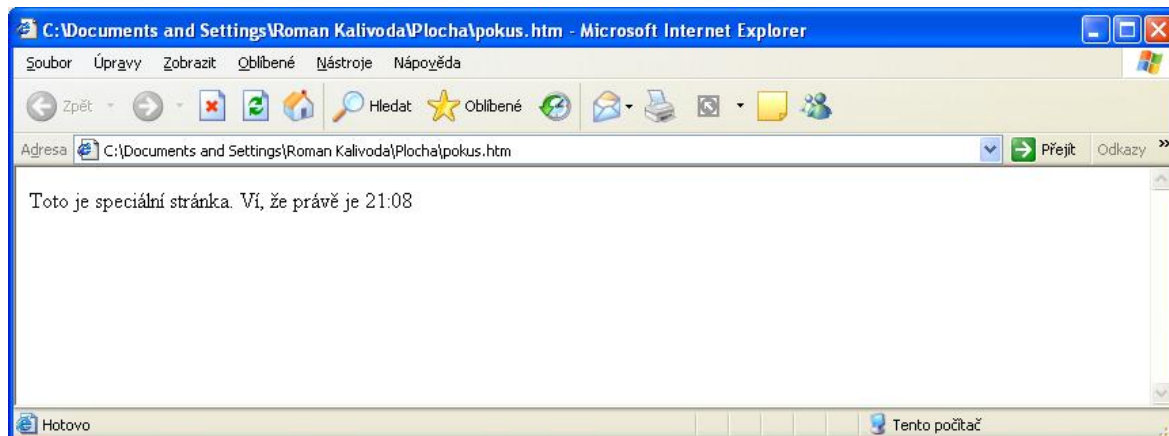
  <BODY>

    Toto je speciální stránka. Ví, že právě je <?echo Date("G:i")??

  </BODY>

</HTML>
```

Stránka nám vypíše aktuální čas na serveru v době, kdy byla generována. To proto, že kód mezi značkami `<? a ?>` se na serveru nejprve provede a teprve pak se výsledek odešle prohlížeči. Jinými slovy, do prohlížeče dorazí něco jako:



Obr. 10. Stránka s kódem PHP

Výpis textu

Uvedu druhý příklad. Ten ukáže, jak pomocí příkazu `echo` vypsát čistý text. V PHP budeme chtít vypsát třeba větu „Dobrý den.“. Pomocí příkazu `echo` to uděláme velice snadno:

```
echo " Dobrý den.";
```

Příkaz `echo` umožňuje vypsát najednou i více řetězců. Jenom je potřeba je oddělit čárkami:

```
echo "Dobrý", " ", "den.";
```

Prohlížeč nemá sebemenší tušení, co všechno se na serveru dělo než mu byl výstup odeslán, vidí jen samotný výsledek. Dodejme, že dít se na serveru mohla celá řada věcí – matematické výpočty, přístupy k databázím, formátování, operace s řetězci a podobné složitější operace.

Proměnné

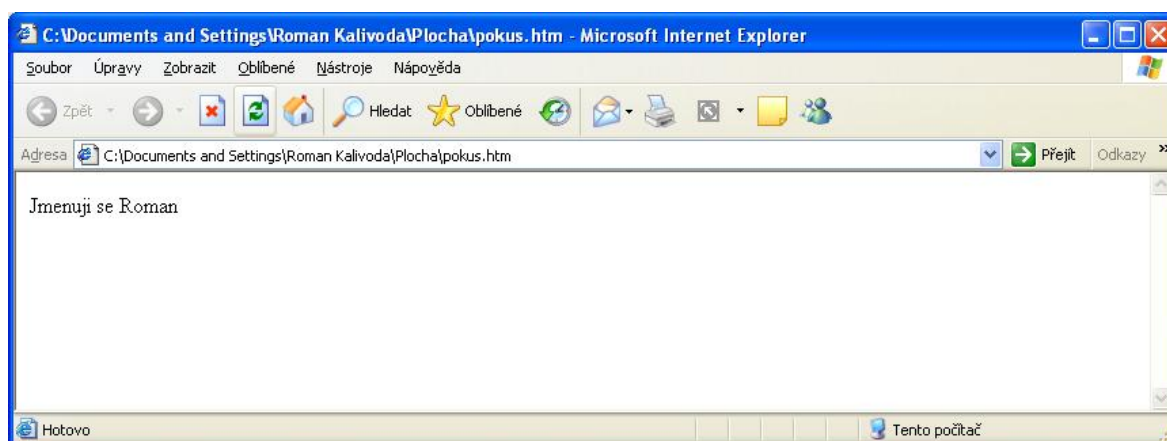
Jako třetí příklad uvedu proměnné. Je to pojmenované místo, do kterého můžete uložit hodnotu, ať už číslo, řetězec, objekt či něco jiného. Každá proměnná je v PHP pojmenována a při použití se dává v PHP před název znak dolaru \$. Jako příklad jsem uvedl třetí ukázkový skript, který ukazuje velmi jednoduché použití proměnné:

```
<html>
<head>
<title>Proměnné</title>
</head>
<body>
<h5>

<?php
    $jmeno = Roman;
    echo "Jmenuji se ", $jmeno, "<br>";
?>

</h5>
</body>
</html>
```

Prohlížeč zobrazí:



Obr. 11. Text v php

PHP rozlišuje v názvech proměnných velikosti písmen. To znamená, že třeba proměnné s názvy \$počet, \$Počet a \$POČET jsou tři naprosto různé proměnné. Této vlastnosti se odborně říká „case sensitive“, tedy citlivost na velikost písmen.

Podmínky

Téměř každý běžný programovací jazyk má konstrukci *if*. Tato konstrukce je v podstatě něco, co umožní provést kus PHP kódu jen za určité podmínky. Následující PHP skript vypíše buď „Právě je dopoledne“, nebo „Právě je odpoledne“ podle aktuálního času.

Celá konstrukce *if* má zhruba takový tvar:

```
if (podmínka)
    příkaz, který se vykoná, pokud je podmínka splněna;
else
    příkaz, který se vykoná, pokud není podmínka splněna;
```

V praxi to vypadá následovně:

```
<html>
<head>
<title>podmínka</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-
1250">
</head>
<body>
<h1><?php
    if (date("A") == "AM")
        echo "Právě je dopoledne";
    else
        echo "Právě je odpoledne";
?></h1>
</body>
</html>
```

Porovnávací operátory

V mém prvním příkladě byl v podmínce použit test na rovnost. Kromě toho je možné zadat v podmínce třeba požadavek, při kterém chceme, aby byla nějaká hodnota menší než druhá apod. Předpokládejme, že máme dvě proměnné s názvem \$a a \$b. Pro porovnání těchto proměnných můžeme použít operátory tak, jak je uvedeno v tabulce II.

Tabulka II. Porovnávací operátory

Příklad	Vysvětlení
$\$a == \b	Požadujeme, aby hodnota proměnné $\$a$ byla rovná hodnotě proměnné $\$b$
$\$a != \b	Požadujeme, aby hodnota proměnné $\$a$ nebyla rovná hodnotě proměnné $\$b$
$\$a <> \b	Požadujeme, aby hodnota proměnné $\$a$ nebyla rovná hodnotě proměnné $\$b$
$\$a < \b	Požadujeme, aby hodnota proměnné $\$a$ byla menší, než hodnota proměnné $\$b$
$\$a <= \b	Požadujeme, aby hodnota proměnné $\$a$ byla menší, nebo rovná hodnotě proměnné $\$b$
$\$a > \b	Požadujeme, aby hodnota proměnné $\$a$ byla větší, než hodnota proměnné $\$b$
$\$a >= \b	Požadujeme, aby hodnota proměnné $\$a$ byla větší, nebo rovná hodnotě proměnné $\$b$

Abychom mohli PHP skript skutečně spustit, musíme ho umístit na server s podporou PHP. V běžném internetovém prohlížeči by jsme si ho jen těžko prohlédli [5].

1.3 MYSQL

1.3.1 K čemu slouží MYSQL

PHP by určitě nebylo tak rozšířené, kdyby neumělo pracovat s databázemi, zvláště pak s MySQL. Je to velmi rychlý databázový systém, který umožňuje ukládat, třídit a hledat požadované data. Zpracováváme-li značný objem dat uložených na serveru a jako výsledek nás zajímá jeden konkrétní záznam (např. zjišťujeme jméno a příjmení osoby podle nám známého telefonního čísla), je zbytečné uložit si všechna data k sobě na počítač a celé je prohledat. Lepší je nechat daný záznam vyhledat serveru a on nám pošle výsledek. Výhody jsou jasné - na pracovní stanici již žádná data neprozkoumáváme (případně řádově menší množství než jich je na serveru) a značně ulehčujeme přenosovým linkám a správě celého systému. Na první pohled by se mohlo zdát, že řešení za použití této architektury přináší pouze samé klady, ale není tomu tak. Systémy v architektuře *client-server* jsou nákladnější na tvorbu a nezanedbatelné nejsou ani nároky kladené na hardware. Lze však říci, že se jedná o přístup, který je „zlatou střední cestou“ v návaznosti na rostoucí objem zpracovávaných dat.

1.3.2 Připojení MYSQL

MySQL je systémová databáze a tudíž běží jako serverový proces. Proto musí každý, kdo se chce k MySQL připojit, znát tyto přihlašovací informace:

Název hostitele

MySQL pochopitelně běží na nějakém stroji. Název hostitele se může zadávat jménem (třeba mysql.nekde.cz) nebo IP adresou. Server, na němž běží databáze nemusí být nutně tentýž, na němž běží webový (http) server.

Port

MySQL naslouchá standardně na portu 3306, pokud není při konfiguraci určeno jinak. Můžeme port vynechat, 3306 je doplněno automaticky.

Název databáze

Na *databázovém serveru* MySQL existuje většinou několik vzájemně nesouvisejících databází. Každá databáze obsahuje žádnou nebo více tabulek a každá může mít vlastní nastavení práv. V praxi to většinou bývá tak, že součástí hostingových služeb je možnost používat jednu databázi. V ní můžeme mít teoreticky nomezený počet tabulek a ostatní uživatelé serveru nemají k naší databázi přístup. My zase nemáme přístup k jejich databázím.

Uživatelské jméno

Uživatelské jméno k databázovému serveru nijak nesouvisí s přihlašovacím jménem k serveru. Ve skutečnosti většinou nebudeme mít na serveru, na němž databáze poběží, zřízen systémový účet. Uživatelské jméno dostaneme od správce databáze.

Heslo

Heslo dostaneme od správce databáze. Většinou bude splňovat základní bezpečnostní pravidla, tzn. bude dostatečně dlouhé a bude se skládat z písmen, číslic, případně dalších znaků.

V PHP se k MySQL databázi můžete připojit pomocí funkce `mysql_connect`. Jejími parametry jsou název hostitele, uživatelské jméno a heslo. Po připojení k MySQL si ještě musíme vybrat svoji databázi, což se děje příkazem `mysql_select_db`. Celé to může v nejjednodušším případě vypadat tak, jak je uvedeno na následující stránce:

```
<?
mysql_connect("localhost", "uzivatel", "heslo");
mysql_select_db("mojedb");
?>
```

Pokud to dopadne úspěšně, budou se od tohoto místa ve skriptu všechny příkazy určené pro MySQL provádět na tomto spojení. Pokud v témže skriptu zavoláme *mysql_connect* znovu se stejnými parametry, pokusí se PHP použít první spojení namísto vytvoření nového (vytvoření nového spojení jde ale vynutit pomocí nepovinného čtvrtého parametru funkce *mysql_connect*). Spojení vytvořené pomocí *mysql_connect* je otevřené až do doby, než je zavolána funkce *mysql_close*, nebo do ukončení běhu skriptu [5].

1.3.3 Spolupráce PHP a MySQL s webovým serverem

Klasickým stránkám odpovídá základní struktura operace serveru. Systém je tvořen dvěma prvky: prohlížečem (web browser) a webovým serverem (web server). Je zde obousměrné spojení. Prohlížeč pošle žádost (Request) serveru a ten mu pošle odpověď (Response).

Struktura, která do procesu zapojí databázi, je však složitější. Prohlížeč pošle přes http žádost o konkrétní stránku, webový server přijme tuto žádost, vytáhne si daný soubor a pošle ho PHP ke zpracování. PHP začne provádět skript. Uvnitř přijde příkaz k připojení k databázi a provedení dotazu. PHP si vytvoří spojení s MySQL a odešle požadovaný dotaz. MySQL přijme dotaz, zpracuje a pošle zpět PHP. PHP dokončí provádění skriptu a naformátované HTML vrátí webovému serveru. Ten pošle HTML zpátky prohlížeči, který vypíše požadovaná data [6].

1.3.4 Základní příkazy SQL

Vkládání dat do databáze

Když chceme pracovat s databází, musíme v ní mít nejdříve uložená nějaká data. To provedeme příkazem *INSERT*. Zjednodušená syntaxe má následující tvar:

```
INSERT INTO jméno_tabulky
        [ (položky) ]
VALUES
        (hodnoty)
```

Příkazu lze použít v několika variantách. První možností je vynechat seznam položek, které chceme do tabulky vkládat. V takovém případě musíme ovšem zajistit, aby za klauzulí *VALUES* následovaly hodnoty všech položek ve stejném pořadí, jak jsou v tabulce definovány.

Příklad:

```
INSERT INTO osoby
VALUES ('Petra', 'Nováčková', '585522/8789', '11.5.1988')
```

Příkaz nám založí do dobře známé tabulky *osoby* záznam pro *Petru Nováčkovou*. Tabulka *osoby* pak bude obsahovat následující řádky (zde byl nový záznam přidán na konec, fyzické umístění však závisí na konkrétním systému):

Tabulka III. Přidání záznamu do tabulky

jméno	příjmení	rodné_číslo	datum
Karel	Smolný	580815/8777	3.12.1992
Marek	Jurda	650404/8956	2.11.1992
Martin	Pavlásek	720303/5511	28.3.1990
Pavel	Jirásek	551212/7877	26.3.1989
Romana	Bourková	745226/8820	3.5.1995
Petra	Nováčková	585522/8789	11.5.1988

Druhou možností je použití názvu sloupců tabulky.

Příklad:

```
INSERT INTO osoby
    (jméno, příjmení, rodné_číslo, datum)
VALUES
    ('Petra', 'Nováčková', '585522/8789', '11.5.1988')
```

Podmínkou příkazu opět je, aby hodnoty za klauzulí *VALUES* odpovídaly vybraným položkám. Podotýkám, že tuto variantu s vyjmenovanými položkami je vhodnější používat vždy. Nenastanou tak problémy s již existujícím *SQL* kódem, pokud dojde ke změně v definici tabulky [5].

Získání dat z databáze

Získávání dat z databáze se provádí příkazem *SELECT*. Zjednodušená syntaxe tohoto příkazu má následující podobu:

```
SELECT [DISTINCT] položky FROM jméno_tabulky [WHERE podmínka_výběru]
[GROUP BY položky, [HAVING podmínka_agregace]] [ORDER BY položky]
```

Pro bližší vysvětlení významu jednotlivých možností tohoto příkazu uvažujme následující jednoduchou tabulku *osoby*:

Tabulka IV. Ukázková tabulka pro příkaz *SELECT*

<i>Jméno</i>	<i>příjmení</i>	<i>rodné_číslo</i>	<i>datum</i>
Karel	Smolný	580815/8777	3.12.1992
Marek	Jurda	650404/8956	2.11.1992
Martin	Pavlásek	720303/5511	28.3.1990
Pavel	Jirásek	551212/7877	26.3.1989
Pavla	Nováková	785308/4569	28.6.1997
Pavla	Nováková	555111/3042	28.5.1990
Romana	Bourková	745226/8820	3.5.1995

Pod položkami (atributy) si můžeme zjednodušeně představit sloupce tabulky, případně funkce. Tedy v případě této tabulky jde o položky *jméno*, *příjmení*, *rodné_číslo* a *datum* (nástupu do zaměstnání). Je nutné poznamenat, že na pořadí jednotlivých položek v tabulce nezáleží. Klauzule *DISTINCT* zařídí, že stejné řádky vybereme pouze jednou. Klíčové slovo *FROM* nám říká, nad jakou tabulkou má být dotaz proveden, "odkud" budou data čerpána. Klauzule *WHERE* nám umožňuje omezit výběr řádků podle požadované podmínky. Pomocí příkazu *ORDER BY* se určuje seřazení výsledku dotazu. Názornější budou následující jednoduché příklady.

Příklad:

```
SELECT jméno,příjmení,rodné_číslo,datum FROM osoby
```

Uvedený příkaz dává výsledek ve tvaru:

Tabulka V. Výsledek příkazu *SELECT*

<i>jméno</i>	<i>příjmení</i>	<i>rodné_číslo</i>	<i>datum</i>
Karel	Smolný	580815/8777	3.12.1992
Marek	Jurda	650404/8956	2.11.1992
Martin	Pavlásek	720303/5511	28.3.1990
Pavel	Jirásek	551212/7877	26.3.1989
Pavla	Nováková	785308/4569	28.6.1997
Pavla	Nováková	555111/3042	28.5.1990
Romana	Bourková	745226/8820	3.5.1995

Další příklad:

```
SELECT jméno,příjmení FROM osoby
```

dává výsledek:

Výsledek:

Tabulka VI. Výsledek příkazu *SELECT* při výběru dvou sloupců

<i>jméno</i>	<i>příjmení</i>
Karel	Smolný
Marek	Jurda
Martin	Pavlásek
Pavel	Jirásek
Pavla	Nováková
Pavla	Nováková
Romana	Bourková

Příklad s použitím klauzule *DISTINCT*:

```
SELECT DISTINCT jméno,příjmení FROM osoby
```

V tomto případě bude výsledek takový:

Tabulka VII. Výsledek příkazu *SELECT* s použitím klauzule *DISTINCT*

<i>jméno</i>	<i>příjmení</i>
Karel	Smolný
Marek	Jurda
Martin	Pavlásek
Pavel	Jirásek
Pavla	Nováková
Romana	Bourková

Příklad s podmínkou:

```
SELECT jméno,příjmení,rodné_číslo,datum FROM osoby WHERE datum>31.11.1992
```

Výsledek:

Tabulka VIII. Výsledek příkazu *SELECT* s použitím podmínky

<i>jméno</i>	<i>příjmení</i>	<i>rodné_číslo</i>	<i>datum</i>
Karel	Smolný	580815/8777	3.12.1992
Romana	Bourková	745226/8820	3.5.1995

Příklad s použitím setřídění podle příjmení:

```
SELECT jméno,příjmení FROM osoby ORDER BY příjmení
```

Výsledek bude následující:

Tabulka IX. Výsledek příkazu *SELECT* s použitím setřídění

<i>jméno</i>	<i>příjmení</i>
Romana	Bourková
Pavel	Jirásek
Marek	Jurda
Pavla	Nováková
Pavla	Nováková
Martin	Pavlásek
Karel	Smolný

Úprava a mazání dat

Když chceme některý záznam v tabulce změnit, řešíme to pomocí příkazu *UPDATE*.

Syntaxe tohoto příkazu má následující podobu:

```
UPDATE jméno_tabulky
    SET položka=hodnota [,položka=hodnota]
    WHERE podmínka
```

Obdobně jako u příkazu *INSERT* je nutné pamatovat na typovou kompatibilitu. Podmínka za klauzulí *WHERE* se řídí stejnými pravidly jako u příkazu *SELECT*. Chceme-li například

změnit datum nástupu do zaměstnání u *Marka Jurdy*, může mít syntaxe *UPDATE* tento tvar:

```
UPDATE osoby
    SET datum='23.6.1990'
    WHERE rodné_číslo IN ('650404/8956')
```

Podmínka mohla mít i jiný tvar, například omezení na *jméno* a *příjmení*. Příkaz *UPDATE* umožňuje jedním krokem změnit více položek. Na to stačí použít oddělovač (zpravidla čárku). Následující příklad ukazuje, jak vyřešit přejmenování osoby s vybraným rodným číslem [5].

```
UPDATE osoby
    SET jméno='Martin', příjmení='Nováček'
    WHERE rodné_číslo='650404/8956'
```

Mazání dat

Příkaz *DELETE* slouží k výmazu záznamů z tabulky. Odstraněn je vždy celý záznam či skupina záznamů (záleží na podmínce). Syntaxe je velmi jednoduchá a má následující tvar:

```
DELETE [FROM] jméno_tabulky
    WHERE podmínka
```

Klauzule *FROM* předcházející jméno tabulky není nutná (v některých implementacích tomu může být jinak), podmínka za *WHERE* podléhá stejným pravidlům jako v předchozích příkazech. Pokud se rozhodneme z ukázkové tabulky smazat všechny osoby, které do firmy nastoupili před 1. lednem roku 1990, příkaz *DELETE* bude vypadat takto:

```
DELETE FROM osoby
    WHERE datum<'1.1.1990'
```

Některé systémy mají mazání realizováno tak, že záznam se sice smaže (někdy ani to nemusí být pravda - označí se pouze za zrušený), ale obsazené místo se neuvolní k novému použití. Důvodem je snaha co nejvíce snížit režii systému na čas potřebný k provedení příkazu [5].

2 PRAKTICKÁ ČÁST

2.1 Slovníky na Internetu

Ještě, než jsem začal programovat zadané webové rozhraní, zkusil jsem nejdřív zjistit, zda se již na Internetu podobné slovníky nacházejí. Česko-anglických oboustranných slovníků jsem našel nespočetně mnoho, ale se zaměřením přímo na překlad pojmů z automatického řízení jsem objevil pouze jeden. Je umístěn na stránkách Univerzity Báňské v Ostravě a to na adrese <http://e-automatizace.vsb.cz/slovník/wordsdiclist.asp>.

2.2 Obecná charakteristika vytvořeného rozhraní

Podle zadaných požadavků jsem provedl návrh webového rozhraní pomocí jazyka PHP ve spolupráci s databází MySQL. Z teoretické části je zřejmé, že pro zadaný projekt nebyla jiná možnost výběru nástroje pro jeho realizaci.

Bylo potřeba navrhnout dynamické stránky na serveru, které spolupracují s databází pojmů, a pro tento úkol by jen programovací jazyk HTML nestačil. Důležité je, aby celý systém byl jednoduchý, rychlý a přehledný. Proto jsem hned na úvodní stránku (viz. obr.12) umístil panel pro rychlé vyhledávání, kde uživatel přímo zadá hledaný výraz a nemusí prolistovávat celý slovník. Celý obsah slovníku si uživatel prohlédne až kliknutím na odkaz [celý slovník](#), který je rovněž hned na úvodní obrazovce.

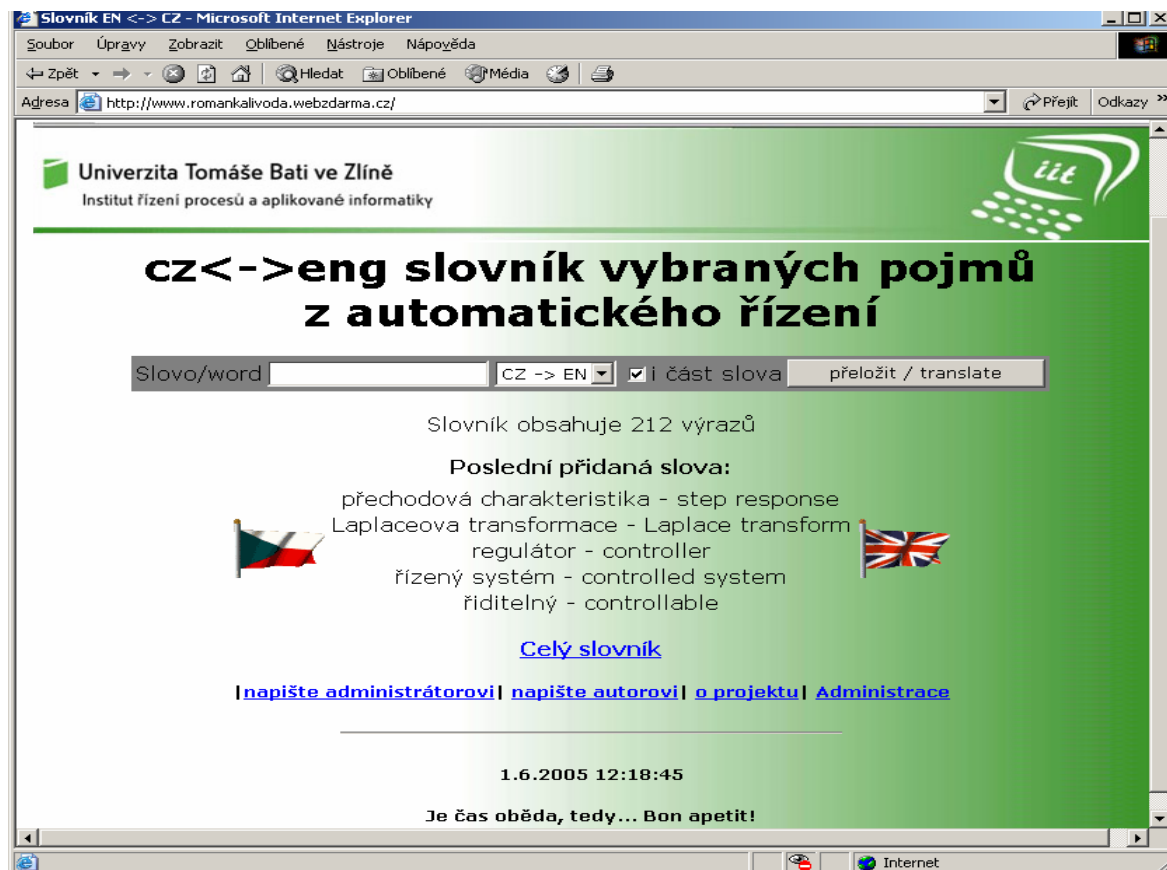
Pro zpestření slovníku je na úvodní stránku vložena tabulka obsahující poslední přidaná slova a počet výrazů, které slovník obsahuje. Dále je vložen také aktuální čas a datum pomocí skriptu, který se automaticky mění bez nutnosti obnovení. Po zadání hledaného výrazu se výsledek objeví nikoliv v novém okně, ale místo původní tabulky posledních přidaných slov (viz. obr.13). Velmi jednoduché přidávání pojmů je zajištěno použitím MySQL databáze, která je uložena na serveru.

Při programování byl použit server www.webzdarma.cz, který umožňuje prostor pro statické i dynamické stránky s podporou databází zdarma. Tento server je však pro velké množství uživatelů přetížen a zpomaluje vyhledávání ve slovníku. Později bude slovník umístěn na stránky Univerzity Tomáše Bati, kde bude pracovat rychleji efektivněji. Současná verze stránek je umístěna na adrese www.romankalivoda.webzdarma.cz.

2.3 Práce se slovníkem

2.3.1 Rychlé vyhledávání

Po zadání adresy, na které se slovník nachází se objeví úvodní stránka.

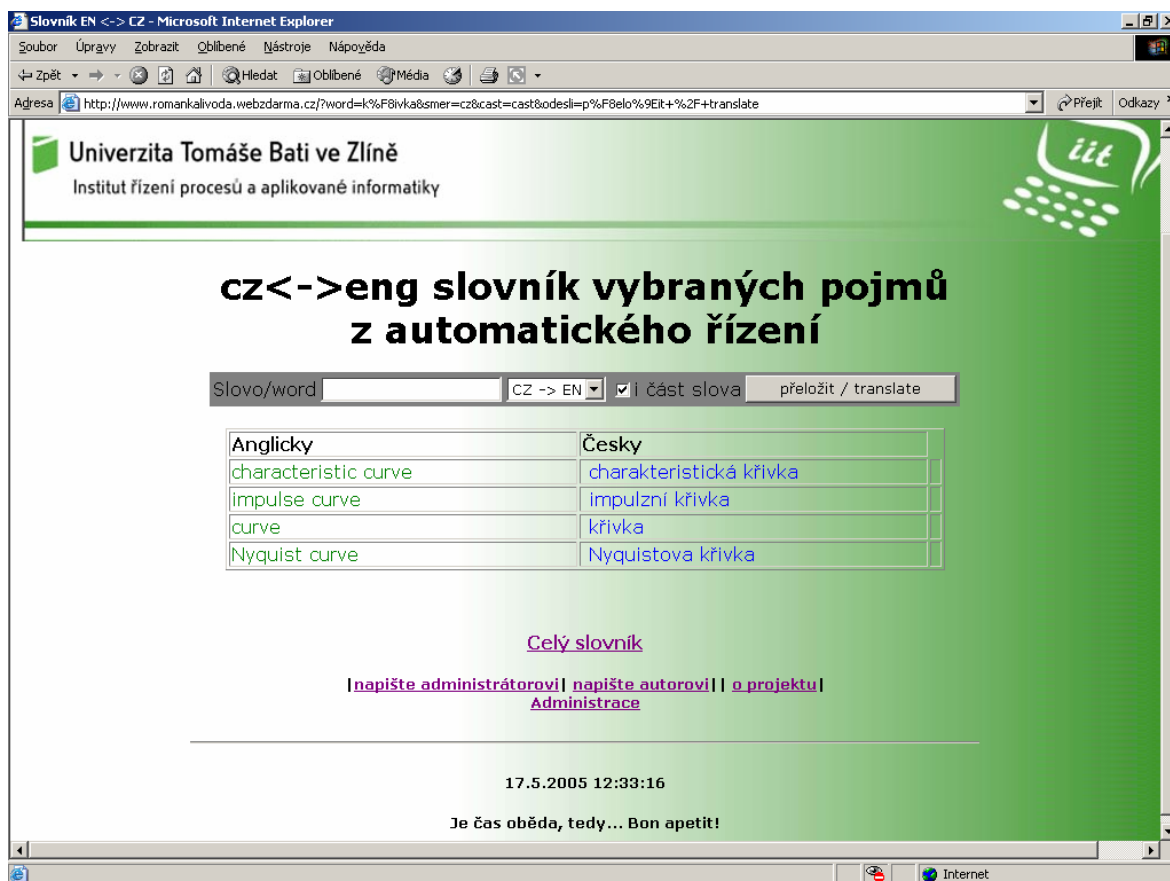


Obr. 12. Úvodní stránka

Jak už jsem naznačil v předchozí kapitole, k rychlému vyhledávání slouží tabulka umístěná přímo pod nadpisem úvodní stránky. Zde zadáváme výraz, který chceme přeložit a také volíme, zda chceme překládat z českého do anglického jazyka, nebo naopak. Zaškrtnutím volby *i část slova* docílíme toho, že nemusíme zadat hledané slovo nebo slovní spojení v kompletním tvaru, ale stačí pouze jeho část. Slovník poté vyhledá všechny výrazy obsahující tu část slova, kterou uživatel zadal do buňky *slovo/word*.

Kliknutím na *přeložit/translate* systém vyhledá odpovídající ekvivalent v zadaném jazyce.

Nyní uvedu příklad překladu slova „křivka“ z českého jazyka do anglického **s použitím** volby *i část slova*. Výsledek vypadá následovně:

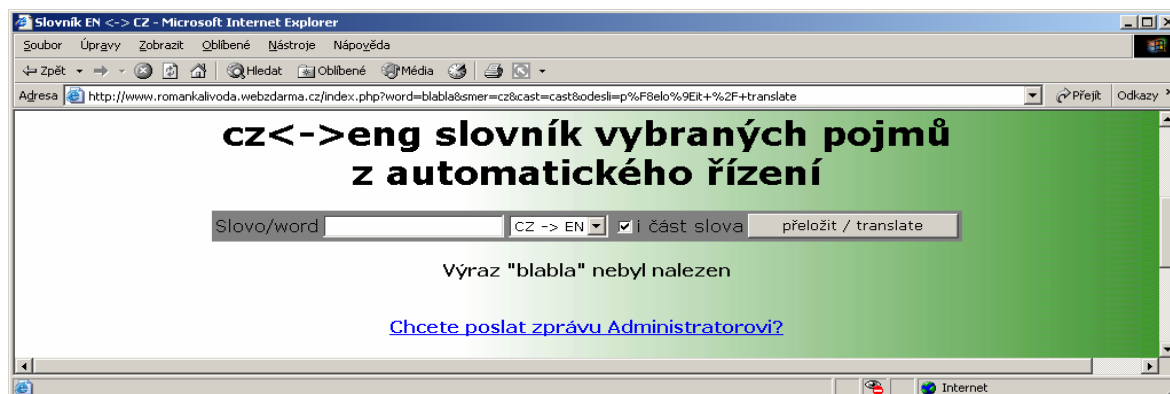
Obr. 13. Výsledek s použitím volby *i část slova*

Pokud překládáme slovo křivka z českého jazyka do anglického **bez použití** volby *i část slova*, systém vybere anglický ekvivalent pouze toho slova, které bylo zadáno. Výsledek vypadá takto:

Obr. 14. Výsledek bez použití volby *i část slova*

Jak vidíme na obrázku, výsledkem hledání je pouze jeden výraz.

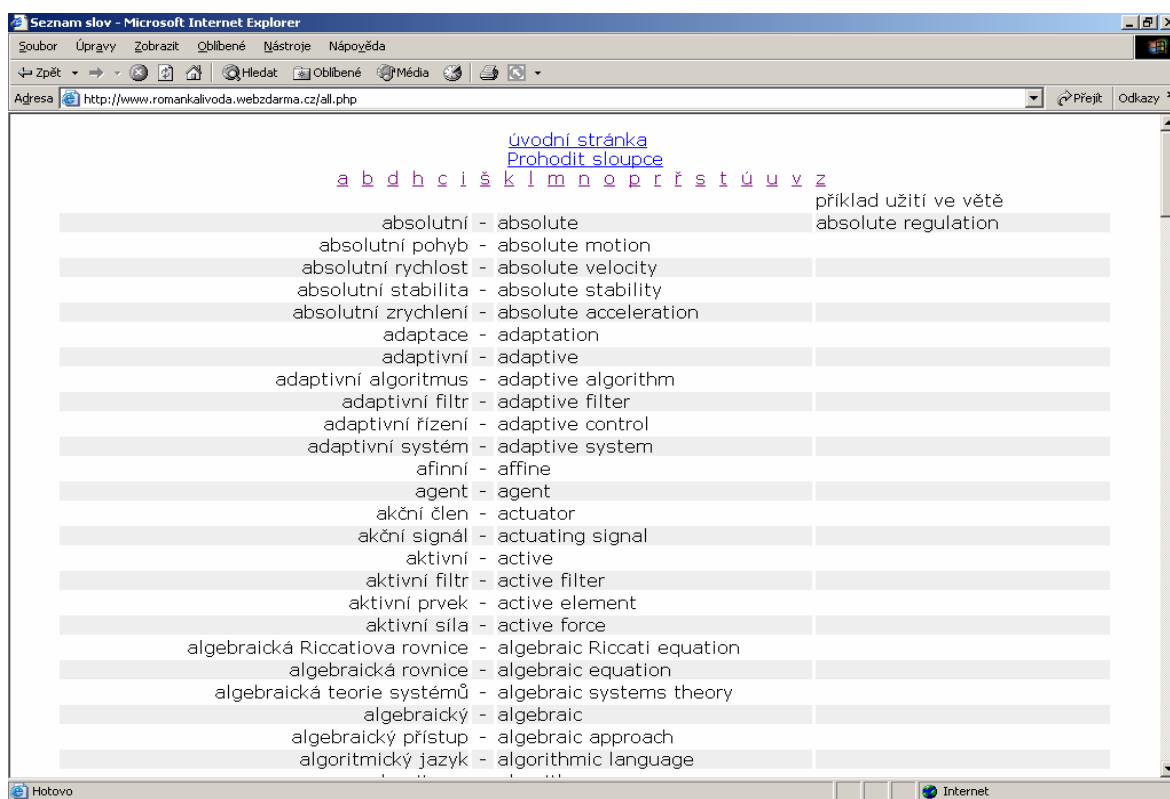
V případě, že slovo není nalezeno v databázi, zobrazí se zpráva, která upozorní uživatele a objeví se dotaz, zda chceme informovat správce zasláním e-mailu o nenalezeném výrazu. Tato „zpětná vazba“ je velmi důležitou součástí pro slovník a jeho další rozvoj.



Obr. 15. Výsledek po nenalezení slova v databázi

2.3.2 Zobrazení celého slovníku

Rychlé vyhledávání je pro nás velmi užitečné, ale mnozí uživatelé si rádi prohlédnou celý obsah slovníku. K tomu slouží odkaz [celý slovník](#) který nám umožní nahlédnout do celé databáze slovíček, jak můžeme vidět na následujícím obrázku.

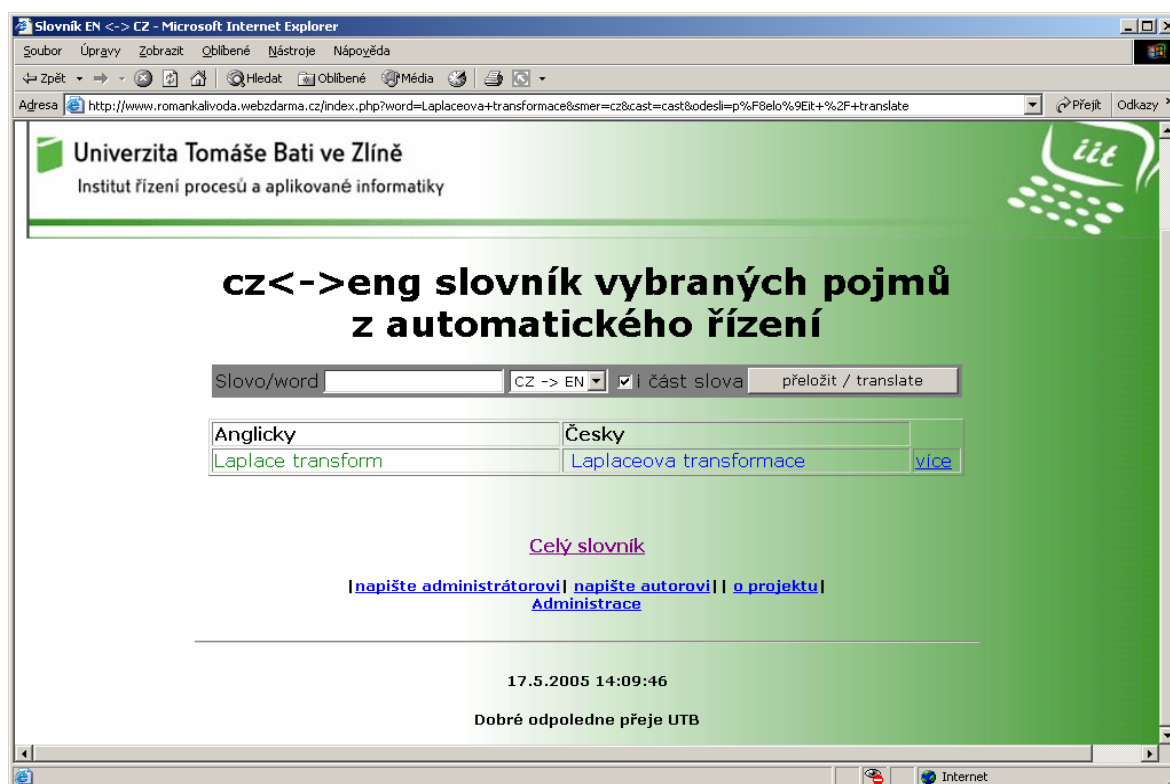


Obr. 16. Celý slovník

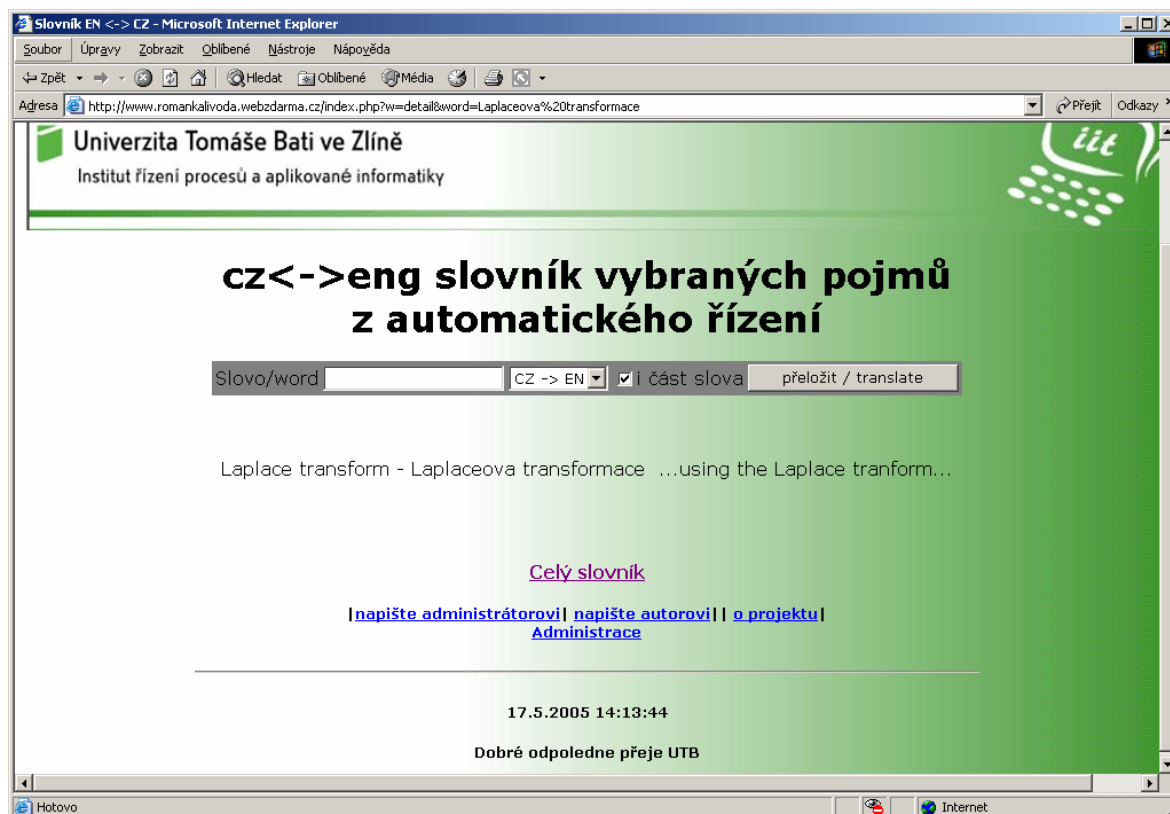
Nachází se zde také odkaz [prohodit sloupce](#). Po kliknutí na něj se prohodí sloupce slovníku tak, že vlevo budou výrazy v anglickém jazyce a vpravo v českém jazyce. Řada odkazů písmen slouží k posunutí obrazovky na slova jimi začínající. Vpravo můžeme vidět

sloupec *příklad užití ve větě*. Ten slouží k rozšíření překladu o ukázkou použití anglického slova v běžném slovním spojení. Samozřejmě není u každého slova, ale pouze u vybraných. Pokud v rychlém vyhledávání zadáme slovo, u kterého je v databázi uveden *příklad užití ve větě*, objeví se za přeloženým slovem ještě jedna buňka s odkazem [více](#). Po kliknutí na odkaz se zobrazí za přeloženým slovem ještě příklad užití slova ve větě.

Na následujících obrázcích je ukáзка překladu výrazu *Laplaceova transformace* z českého jazyka do anglického. A poté výsledek po kliknutí na odkaz [více](#).



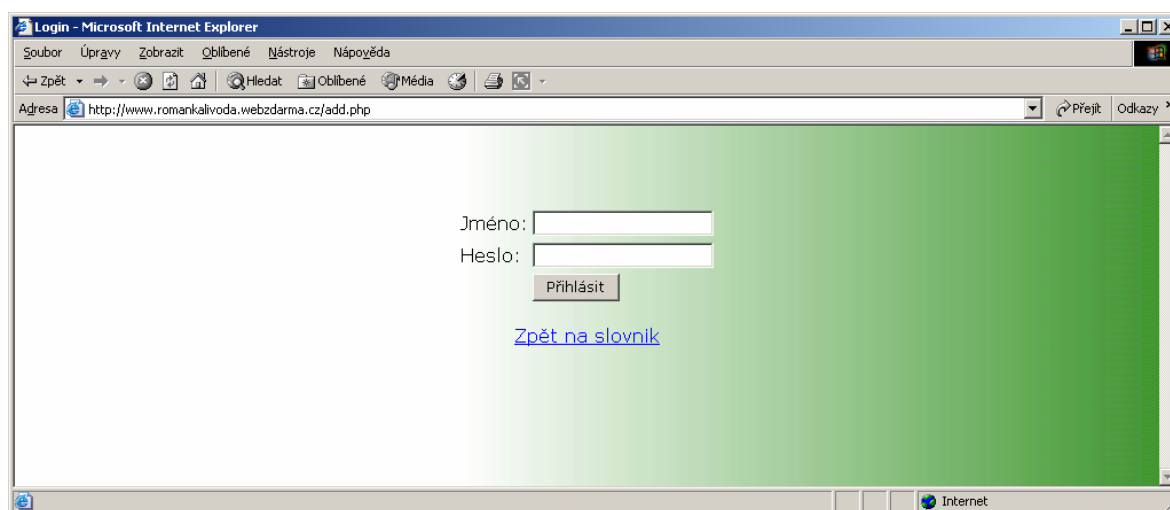
Obr. 17. Překlad slova s příkladem užití ve větě



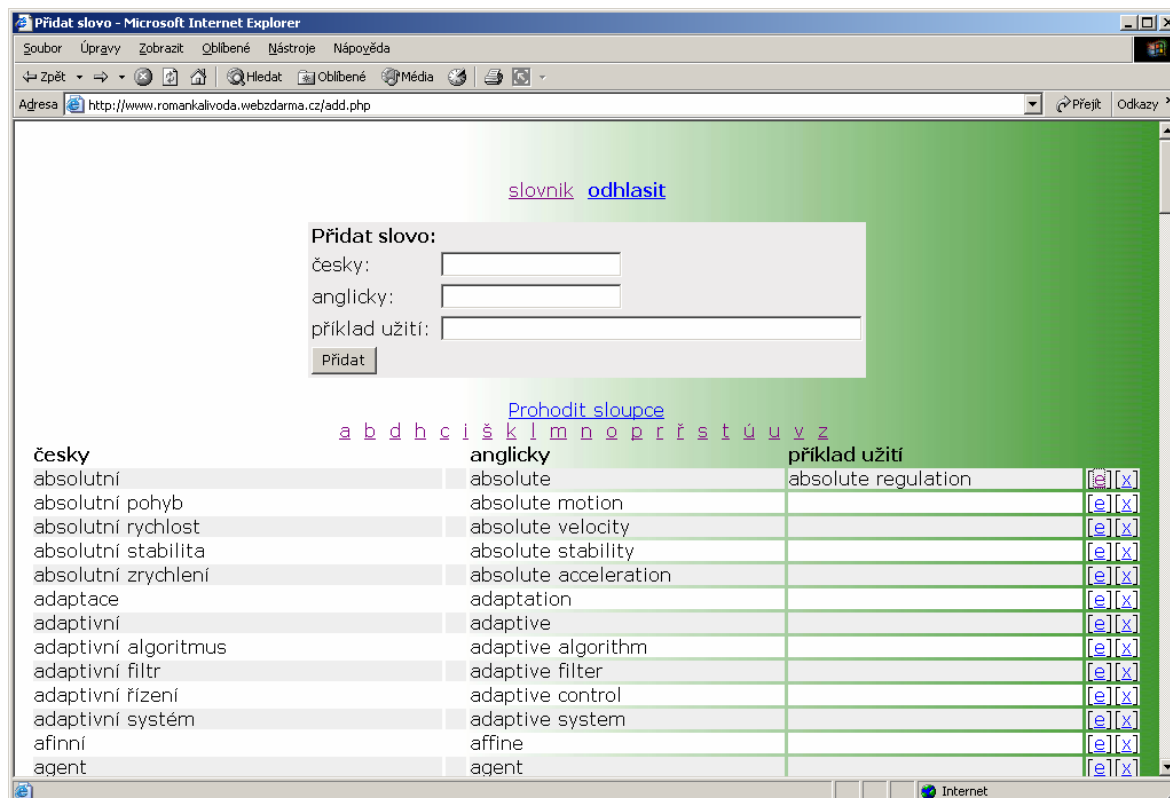
Obr. 18. Překlad slova s příkladem užití ve větě po kliknutí na odkaz více

2.3.3 Administrace slov

Administraci slov můžeme provádět kliknutím na odkaz Administrace. Ten slouží pouze pro správce slovníku nebo autora. Proto je vstup pro nežádoucí uživatele ošetřen tak, že se objeví obrazovka s požadavkem zadání jména a hesla (obr. č.19). Po správném vyplnění máme možnost přidávat, mazat a upravovat slova v databázi (obr. č.20).

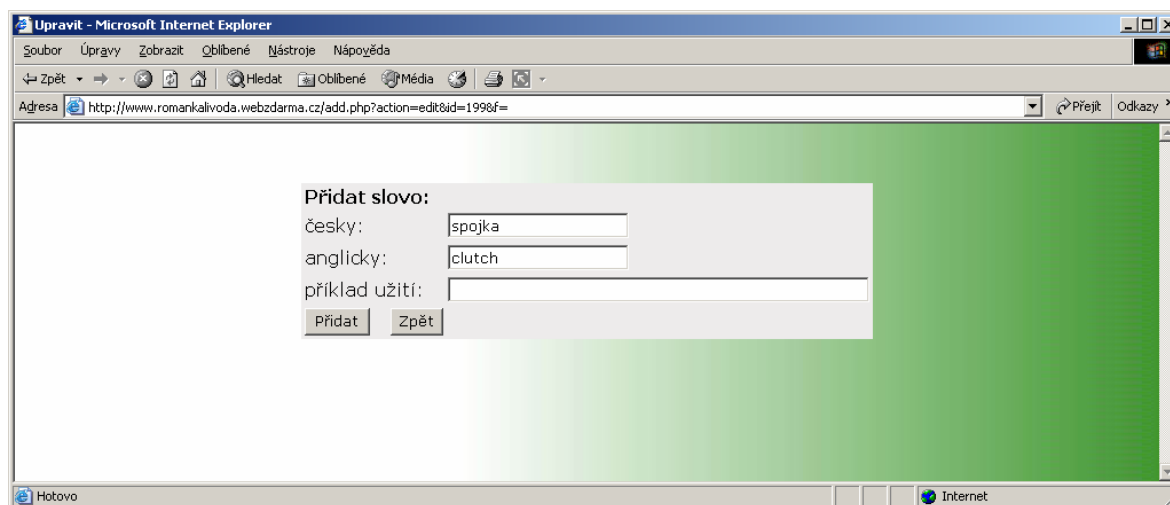


Obr. 19. Přihlašovací obrazovka pro administraci



Obr. 20. Obrazovka pro přidávání, mazání a editaci slov

Přidání výrazu do databáze provedeme vyplněním tabulky **přidat slovo** a klikneme na příkaz přidat. Mazání provádíme kliknutím na odkaz [x], který se nachází na konci každého řádku databáze. Editaci již existujících výrazu provádíme kliknutím na odkaz [e], po kterém se nám objeví okno s daným slovem a můžeme jej libovolně upravit. Příklad editace slova *spojka* je na následujícím obrázku.



Obr. 21. Editace slov

2.3.4 Ostatní funkce

Kromě vyhledávání výrazu ve slovníku může také uživatel informovat o svých připomínkách správce nebo autora. K tomu slouží odkazy [napište administrátorovi](#) a [napište autorovi](#). Odkaz [o projektu](#) seznamuje uživatele s informacemi, které se týkají této práce.

2.4 Zdrojový kód

2.4.1 Rozdělení do souborů

Pro přehlednost a snadné pozdější úpravy byl zdrojový kód rozdělen do následujících souborů:

`index.php`

`func.inc`

`add.php`

`addedit.inc`

`all.php`

`mailer.php`

`db.inc.php`

`conf.inc.php`

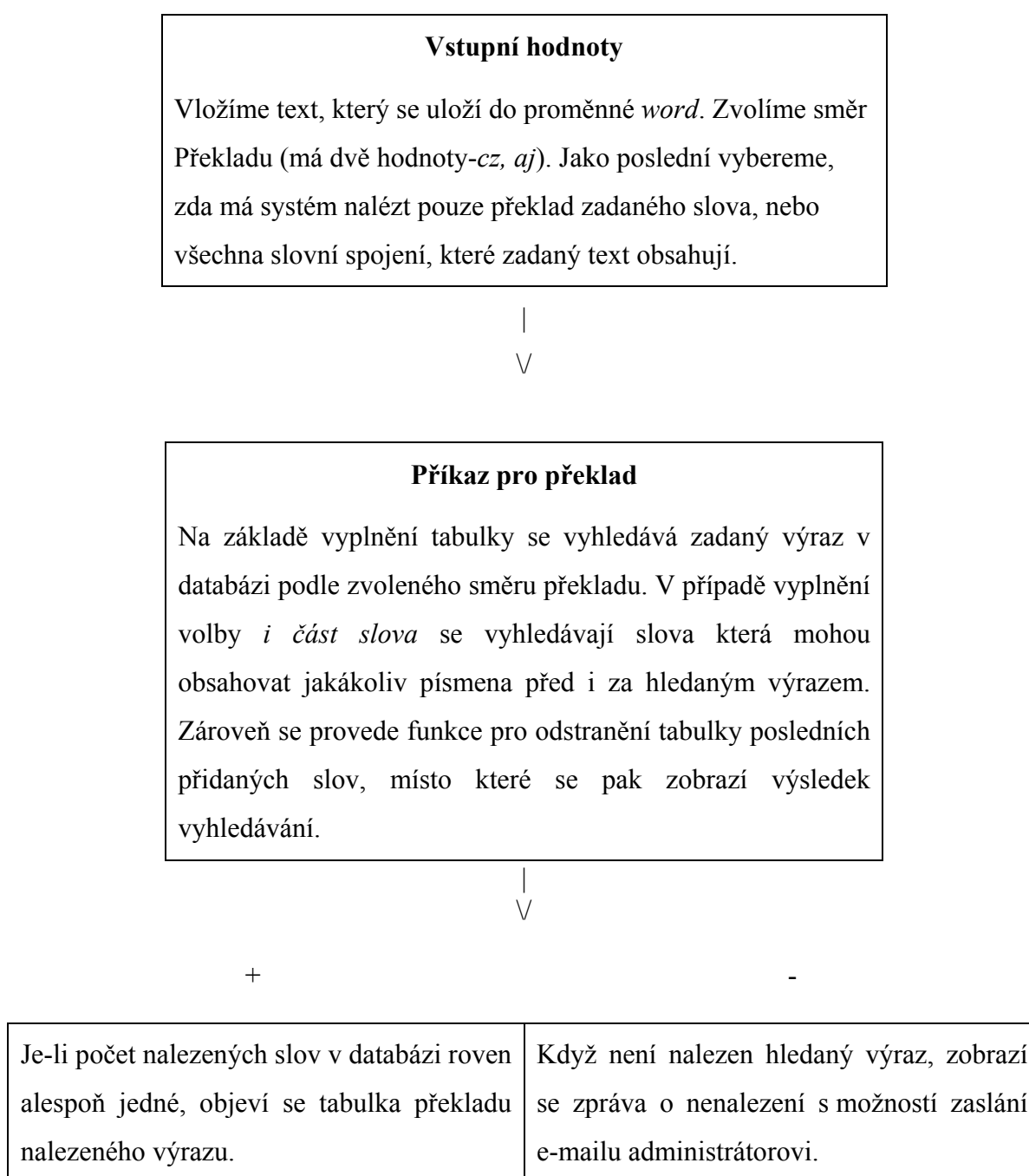
`styl.css`

Index.php je část, ve které je naprogramována úvodní stránka spolu s tabulkou pro rychlé vyhledávání. Soubor **Func.inc** navazuje na *index.php*. Obsahuje funkce pro rychlé vyhledávání a pro zobrazení posledních přidaných slov. Soubor **add.php** slouží k přidávání nových výrazů, jejich editaci a mazání. Formulář pro tyto úpravy je v části **addedit.inc**. Pro zobrazení celého slovníku je naprogramován soubor **all.php**. Část **mailer.php** slouží k upozornění administrátora o nenalezeném výrazu zasláním e-mailu. Soubor **db.inc.php** slouží k nastavení připojení na databázový server. Soubor **conf.inc.php** nám umožní snadno a rychle změnit nejen přihlašovací jméno a heslo do

systemu pro přidávání a úpravu nových výrazů, ale také e-mailovou adresu autora či administrátora. Soubor *styl.css* nám umožňuje nastavit font písma a pozadí stránek.

2.4.2 Princip rychlého vyhledávání

Na následujícím schématu jsem graficky znázornil, jak pracuje funkce pro rychlé vyhledávání výrazu ve slovníku:



V praxi to vypadá následovně:

Zdrojový kód tabulky pro rychlé vyhledávání:

```
<table border="0" bgcolor="Gray">
  <tr>
    <td>
      <p>Slovo/word</p>
    </td>
    <td>
      <input type="text" name="word">
    </td>
    <td>
      <select name="smer" size="1">
        <option value="cz">CZ -&gt; EN</option>
        <option value="aj">EN -&gt; CZ</option>
      </select>
    </td>
    <td>
      <input type="checkbox" name="cast" value="cast" checked>i část
slova
    </td>
    <td>
      <p><input type="submit" name="odesli" value="přeložit /
translate"></p>
    </td>
  </tr>
</table>
</form>
<?
include("./func.inc.php");
ProcessSearch($HTTP_GET_VARS[word],$HTTP_GET_VARS[smer],
$HTTP_GET_VARS[cast]);
```

```
if(!isset($word))LastWords();  
?>
```

Poslední řádek v kódu znamená, že pokud byl vyplněn a odeslán formulář rychlého vyhledávání, tak se nezobrazí poslední přidaná slova. Tabulka pro poslední přidaná slova je naprogramována jako funkce *LastWords()*.

Zdrojový kód funkce pro hledání výrazu ve slovníku:

```
<?  
function ProcessSearch($word,$smer,$cast){  
    include "../conf.inc.php";  
    include "../db.inc.php";  
    $server=DB_HOST;  
    $user=DB_USER;  
    $pass=DB_PASS;  
    $db=DB_NAME;  
    mysql_Pconnect ($server,$user,$pass);  
    if ($cast=="cast")  
        $result = mysql ($db, "select * from cz_aj where $smer like  
'%$word%' ORDER BY $smer;");  
    else $result = mysql ($db, "select * from cz_aj where $smer like  
'$word' ORDER BY $smer;");  
    if($result)$pocet = mysql_NumRows ($result);  
    $i=0;  
    if($word){  
        if($pocet){  
            echo"<table border=\"1\"><tr><td><p><b>Anglicky</b></p>";  
            echo "</td><td><b>Česky</b></td><td><b>Příklad  
užití</b></td></tr>";  
            while ($i<$pocet){  
                echo "<tr><td width='300'><p><font color='green'>";
```

```

        echo mysql_Result ($result,$i, "aj");

        echo "</font></p></td><td width='300'><p>&nbsp;<font
color='blue'>";

        echo mysql_Result ($result,$i, "cz");
        echo "</font></p></td><td>";

        echo mysql_result($result,$i,"example")."&nbsp;";
        echo "</td></tr>";

        $i++;
    }

}

else echo "<h4>Výraz \"$word\" nebyl nalezen</h4>

<br><b>
<a href=mailer.php?word=".urlencode($word).">Chcete poslat zprávu
Administratorovi?
</a></b><br><br>";

}

```

V této funkci pro hledání jsou nejprve nastaveny informace o databázovém serveru, aby v něm bylo možné vyhledávat a jsou stanoveny základní podmínky:

První podmínka:

```

if ($cast=="cast")

    $result = mysql ($db, "select * from cz_aj where $smer like '%$word%'
ORDER BY $smer;");

    else $result = mysql ($db, "select * from cz_aj where $smer like '$word' ORDER
BY $smer;");

```

Tato podmínka určuje, zda je vyplněná buňka *i část slova*. Pokud je splněna, vyhledává všechny výrazy obsahující zadaný výraz (%\$word%), i s libovolnými znaky před i za ním.

Pokud není splněna, vyhledá se pouze výraz odpovídající zadanému textu ('\$word').

Další podmínky:

```
if($result)$pocet = mysql_NumRows ($result);
    $i=0;
    if($word){
        if($pocet){
            echo"<table border=\"1\"><tr><td><p><b>Anglicky</b></p>";
            .
            .

            else echo "<h4>Výraz \"$word\" nebyl nalezen</h4><br><b>
<a href=mailer.php?word=\".urlencode($word).\">Chcete poslat zprávu Administratorovi?
```

Tyto podmínky určují, zda byl výraz nalezen. Jestliže ano, zobrazí se tabulka překladu, naopak když není nalezen, zobrazí se zpráva o nenalezení výrazu a dotaz, zda chce uživatel poslat e-mail administrátorovi.

2.4.3 Tabulka poslední přidaná slova

Zdrojový kód funkce pro zobrazení tabulky posledních přidaných slov:

```
function LastWords() {
    echo "<h3>Poslední přidaná slova</h3><table border=1>";
    echo "<tr>\n";
        echo "<td rowspan=\"6\"><img src=\"image/06.gif\"
width=\"68\" height=\"50\" border=\"0\"></td>\n";
        echo "<td></td>\n";
        echo "<td rowspan=\"6\"><img src=\"image/03.gif\"
width=\"68\" height=\"50\" border=\"0\"></td>\n";
        echo "</tr>\n";
    $res=mysql_query("SELECT * FROM cz_aj ORDER BY datetime DESC LIMIT
5");
    while($data=mysql_fetch_array($res)){
        echo "<tr><td border=1 align=center>$data[cz] -
$data[aj]</td></tr>\n";
    }
    echo "</table>\n";
}
```

Tato funkce vybere z databáze *cz_aj* pět slov (*DESC LIMIT 5*"), které jsou seřazeny podle času vložení a vypíše je do tabulky. Do krajních buněk tabulky jsou vloženy obrázky anglické a české vlajky. Tato tabulka nemá žádné ohraničení.

2.4.4 Funkce pro zobrazení celého slovníku

V souboru *all.php* jsou tři základní funkce. První z nich vybere pomocí příkazu *SELECT* data z databáze a vypíše je do tabulky.

Tabulka má tento tvar:

Tabulka X. Výpis slov z databáze

[cz]	[aj]	[example]
Výraz v češtině	Výraz v angličtině	Příklad užití ve větě

Ukázka zdrojového kódu:

```
$res=mysql_query("SELECT *,LOWER(SUBSTRING($order,1,1)) AS pism FROM
cz_aj ORDER by $order");

$osc=1;

while($data=mysql_fetch_array($res)){

    if($osc==1) $col="#EEEEEE";

    else $col="#FFFFFF";

    $osc=!$osc;

    $ti("<a name=\"".substr($data[$order],0,1).\""></a>";

    if($f=="en")

        echo "<tr bgcolor=$col><td
align=right>$data[aj]</td><td>&nbsp;-&nbsp;</td><td>$ti
$data[cz]</td><td>$data[example]</td></tr>\n";

    else

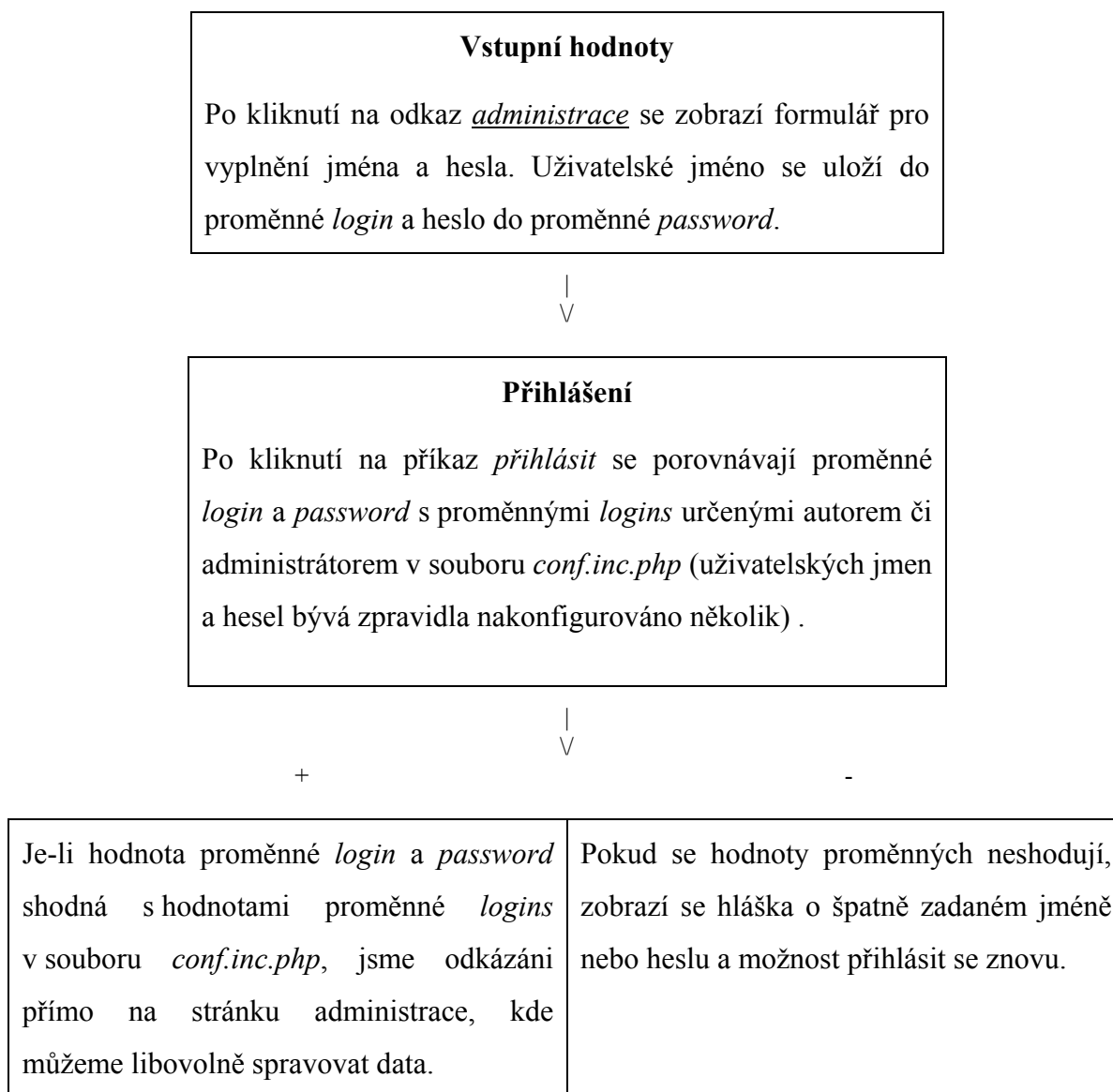
        echo "<tr bgcolor=$col><td
align=right>$data[cz]</td><td>&nbsp;-&nbsp;</td><td>$ti
$data[aj]</td><td>$data[example]</td></tr>\n"; }
```

Příkazem *SELECT* se vyberou data z databáze. Proměnná *\$order* nám určuje směr překladu. Data do tabulky se vypisují pomocí cyklu *while* a zároveň se přiřazují k výrazům

a *ref* se pak na obrazovku vypíše odkazy počátečních písmen, které jsou v databázi uložena jako sloupec [*pism*].

2.4.5 Princip administrace slov

Zdrojový kód souboru *add.php* je velice podobný se souborem *all.php*. Zde se také zobrazí celý slovník, ale k tomu máme ještě možnost upravovat data v databázi (tj. přidávat, mazat a editovat). Jak jsem již uvedl v kapitole *práce se slovníkem*, vstup na stránku sloužící pro správu databáze je povolen pouze pro administrátora nebo autora. To je ošetřeno vstupním jménem a heslem, které je nakonfigurováno v souboru *conf.inc.php* jako proměnná *logins*. Na následujícím schématu jsem znázornil, jak pracuje funkce pro zabezpečení vstupu pro neoprávněné uživatele.



V praxi vypadá zdrojový kód přihlášení následovně:

```
if($action=="login"){

    if( ($login)&&($password)&&($logins[$login]==$password) ){

        $user=$login;

        $auth=session_id();

        header("Location: add.php");

        exit;

    }

    header("Location: add.php?badlogin=1");

    exit;

}

if($badlogin){

    echo "spatne jmeno nebo heslo <a href=add.php>prihlasit</a>";

    exit;

}

if($_SESSION_VARS["user"]){ }

else{

    echo "<html><head><title>Login</title>\n";

    echo "<meta http-equiv=\"content-type\" content=\"text/html; charset=windows-1250\">\n";

    echo "</head><body>\n";

    echo "<br><form action=add.php method=post><input type=hidden name=action value=login>\n";

    echo "<table border=0><tr><td>Jméno:</td><td><input type=text name=login></td></tr>\n";

    echo "<tr><td>Heslo:</td><td><input type=password name=password></td></tr>\n";

    echo "<tr><td></td><td><input type=submit value=Přihlásit></td></tr></table>\n";

    echo "<br><a href=index.php>Zpět na slovník</a></body></html>";

    exit;

}
```

Vytvořil jsem pomocnou proměnnou *action* do které se ukládají akce, které chceme provést. Tato proměnná hraje velmi významnou roli nejen pro přihlašování, ale především pro správu dat. Pomocí formuláře pro přidání slova a odkazů [x] a [e] (viz. obr. 20), které se nachází na konci každého řádku, se ukládají do proměnné *action* hodnoty *add*, *del*, *edit* a *update*. Jestliže je proměnná *action* nastavená na hodnotu *add*, provede se pomocí příkazu *INSERT* přidání výrazu do databáze. Druhou možností je, že je proměnná *action* rovna hodnotě *del*. Pak se pomocí příkazu *DELETE* smaže daný řádek. Je-li proměnná *action* rovna hodnotě *edit*, zobrazí se tabulka pro úpravu výrazu ze souboru *addedit.inc*. Po té se uloží do proměnné *action* hodnotu *update*. Pak máme možnost upravit data. Pro lepší představu, jak tyto operace pracují uvedu části zdrojového kódu:

```
if($action=="add"){

    $res=mysql_query("INSERT INTO cz_aj SET aj=\"$aj\", cz=\"$cz\",
example=\"$example\", datetime=".time());

    if(!$res)die("chyba");

    header("Location: add.php?f=$f");

    exit;

}

if($action=="update"){

    $res=mysql_query("UPDATE cz_aj SET aj=\"$aj\", cz=\"$cz\",
example=\"$example\" WHERE id=$id");

    if(!$res)die("chyba");

    header("Location: add.php?f=$f");

    exit; }

if($action=="del"){

    $res=mysql_query("DELETE FROM cz_aj WHERE id=$id");

    if(!$res)die("chyba");

    header("Location: add.php?f=$f");

    exit; }

if($action=="edit"){

    $res=mysql_query("SELECT * FROM cz_aj WHERE id=$id");

    if(!$res)die("chyba");

    $data=mysql_fetch_array($res);
```

```

        echo "<html><head><title>Upravit</title></head><link
rel=StyleSheet type=\"text/css\" href=\"../styl.css\"></head><body>\n";

        include "addedit.inc.php";

        echo "</body></html>";

        exit; }

```

Tabulka pro přidání a úpravu slov naprogramovaná v souboru *addedit.inc*

```

<table border="0" bgcolor="#EDEBEB">

    <tr>

        <td><b>Přidat slovo:</b></td><td></td>

    </tr>

    <tr>

        <td>česky:</td><td><input type=text name=cz
value="<?=$data[cz]?>"></td>

    </tr>

    <tr>

        <td>anglicky:</td><td><input type=text name=aj
value="<?=$data[aj]?>"></td>

    </tr>

    <tr>

        <td>příklad užití:</td><td><input size=50 type=text
name=example value="<?=$data[example]?>"></td>

    </tr>

    <tr>

        <td><input type=submit value=Přidat>

        <? global $HTTP_GET_VARS?>

        <input type="hidden" name="f" value="<?=$HTTP_GET_VARS[f]?>">

        <input type="hidden" name="id" value="<?=$data[id]?>">

        <?

            if(!isset($data))echo "<input type=hidden name=action
value=add>";

            else {

```

```
                                echo "<input type=hidden name=action
value=update>";

                                echo "<input type=\"button\" value=\"Zpět\"
onclick=\"history.go(-1)\">";

                                }

                                ?>

                                </td><td></td>

                                </tr>

                                </table>
```

2.5 Grafický vzhled

2.5.1 Barvy a pozadí

Při návrhu vzhledu slovníku jsem bral ohled na to, že stránky budou v budoucnu umístěny na školní web. Požadavkem bylo, aby vzhled i barvy byly v podobném stylu jako stránky institutu. Proto bylo zvoleno zelené pozadí s úvodním logem Univerzity Tomáše Bati. Pozadí a písmo lze libovolně měnit v souboru *styl.css*.

2.5.2 Návrh šablony

Aby byl vzhled při různém rozlišení stejný, využil jsem tabulky. Tato tabulka byla u stránek s logem univerzity rozdělena na dvě části. Návrh ukazuje následující tabulka:

Tabulka XI. rozdělení šablony stránky tabulkou

LOGO
Obsah stránky

U stránek bez loga nebylo potřeba tabulku rozdělovat.

2.6 Vývoj projektu

2.6.1 Vývojové prostředky

Vývoj stránek probíhal většinu času na mém osobním počítači.

Tabulka XII. Technické parametry a software počítače pro vývoj systému

Počítač	Athlon 64 3000, socket (939), 512 MB RAM (400Mhz), pevný disk 200GB(7200 ot./min), grafická karta RADEON X700 PRO 128 MB, monitor Acer 17" LCD 12ms, DVI.
System	Windows XP professional
Editor	PHPedit 1.0, Front page (návrh šablony)

2.6.2 Testování na Internetu

Funkce slovníku bylo třeba stále testovat v reálném prostředí. K tomuto účelu mi sloužil uživatelský účet na serveru <http://www.webzdarma.cz>. Ten poskytuje zdarma prostor pro webové stránky, použití jazyka PHP a databáze MySQL. Součástí je také administrační rozhraní databáze *phpMyAdmin*.

ZÁVĚR

Cílem bakalářské práce bylo vytvořit takové webové rozhraní, které by usnadnilo překlad výrazů o oboru automatického řízení nejen pro studenty naší univerzity, ale i pro všechny uživatele na Internetu. Snahou bylo vytvořit takový slovník, aby práce s ním byla rychlá, jednoduchá a příjemná.

V teoretické části jsou popsány všechny prostředky, které navržený systém využívá. Jedná se o jazyk HTML a kaskádové styly, skriptovací jazyk PHP a databázový systém MySQL.

V praktické části je podrobně popsána práce se slovníkem a také je zde rozebrán a stručně vysvětlen zdrojový kód celého systému. Další část je věnována návrhu vzhledu a prostředkům pro realizaci.

Stránky jsou navrženy tak, aby mohly být jednoduše upravovány a aktualizovány. Tento systém je pro studenty a učitele připraven tak, aby se mohli podílet na rozšiřování databáze výrazů. Současně autor počítá s administrátorem, který se bude starat o správu dat.

Díky dnešnímu rozšíření Internetu může být tento slovník velmi dobrou pomůckou pro velký počet uživatelů.

Jádro systému je navíc navrženo tak, že v budoucnu může být využito pro různá uplatnění, např. pro jiné vícejazyčné slovníky apod.

SEZNAM POUŽITÝCH ZDROJŮ

- [1] Jiří Kosek: Podrobný průvodce HTML, Grada 1998. [cit. 2005-4-15]
- [2] Petr Broža: Tvorba WWW stránek, Computer Press 1999. [cit. 2005-4-16]
- [3] Základy HTML. Učebnice HTML jazyka [online]. [cit. 2005-4-20]. Dostupné z URL:
< <http://www.jakpsatweb.cz/zaklady-html.html> >.
- [4] Tvorba webu. Učebnice pro tvorbu webu [online]. [cit. 2005-4-22]. Dostupné z URL:
< <http://www.webtvorba.cz> >.
- [5] Vše o programování. Průvodce programovacími jazyky [online]. [cit. 2005-4-26].
Dostupné z URL: <<http://www.zive.cz/h/Programovani/>>.
- [6] Thomson L., Gilmore W., Welling L. PHP a MySQL – rozvoj webových aplikací.
1. vyd. Praha: SoftPress s.r.o., 2002, ISBN 80-86497-20-8. [cit. 2005-5-2].
- [7] Jiří Kosek: PHP – tvorba interaktivních internetových aplikací. Granada Publishing,
1999, ISBN 80-7169-373-1.
- [8] <http://www.kosek.cz/php>
- [9] <http://www.builder.cz>
- [10] Anglicko-český slovník automatizační techniky a řízení [online]. Dostupný z URL:
<<http://e-automatizace.vsb.cz/slovník>>
- [11] <http://www1.elsevier.com/homepage/saf/ifac/site/IPV%20keywords.htm>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

WWW	World-Wide-Web – oblast Internetu přístupná webovými prohlížeči
HTML	HyperText Markup Language – standard pro tvorbu webových stránek
PHP	Hypertext PreProcessor – skriptovací jazyk pro tvorbu webových stránek
MySQL	system relačních databází
CSS	Cascading Style Sheets – kaskádové styly
URL	Uniform Ressource Locator – jedinečné označení objektu Internetu
SQL	Structured Query Language – jazyk sloužící pro komunikaci s databází
HTTP	HyperText Transfer Protocol – přenosový protokol webové služby
C++	objektově orientovaný programovací jazyk C

SEZNAM OBRÁZKŮ

<i>Obr. 1. Odkaz</i>	10
<i>Obr. 2. Druhy písma</i>	12
<i>Obr. 3. Nadpisy</i>	13
<i>Obr. 4. Zalomení řádku</i>	13
<i>Obr. 5. Vložení obrázku</i>	14
<i>Obr. 6. Jednoduchá tabulka</i>	15
<i>Obr. 7. Tabulka s ohraničením</i>	16
<i>Obr. 8. Poznámka</i>	16
<i>Obr. 9. Obyčejná stránka</i>	21
<i>Obr. 10. Stránka s kódem PHP</i>	22
<i>Obr. 11. Text v php</i>	23
<i>Obr. 12. Úvodní stránka</i>	34
<i>Obr. 13. Výsledek s použitím volby i část slova</i>	35
<i>Obr. 14. Výsledek bez použití volby i část slova</i>	35
<i>Obr. 15. Výsledek po nenalezení slova v databázi</i>	36
<i>Obr. 16. Celý slovník</i>	36
<i>Obr. 17. Překlad slova s příkladem užití ve větě</i>	37
<i>Obr. 18. Překlad slova s příkladem užití ve větě po kliknutí na odkaz <u>více</u></i>	38
<i>Obr. 19. Přihlašovací obrazovka pro administraci</i>	38
<i>Obr. 20. Obrazovka pro přidávání, mazání a editaci slov</i>	39
<i>Obr. 21. Editace slov</i>	39

SEZNAM TABULEK

Tabulka I. Základní atributy tagu <i>body</i>	8
Tabulka II. Porovnávací operátory	25
Tabulka III. Přidání záznamu do tabulky	28
Tabulka IV. Ukázková tabulka pro příkaz <i>SELECT</i>	29
Tabulka V. Výsledek příkazu <i>SELECT</i>	30
Tabulka VI. Výsledek příkazu <i>SELECT</i> při výběru dvou sloupců.....	30
Tabulka VII. Výsledek příkazu <i>SELECT</i> s použitím klauzule <i>DISTINC</i>	30
Tabulka VIII. Výsledek příkazu <i>SELECT</i> s použitím podmínky	31
Tabulka IX. Výsledek příkazu <i>SELECT</i> s použitím setřídění	31
Tabulka X. Výpis slov z databáze	46
Tabulka XI. rozdělení šablony stránky tabulkou	52
Tabulka XII. Technické parametry a software počítače pro vývoj systému	53