VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

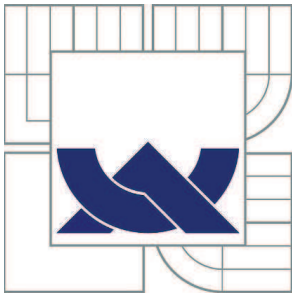
INTERNETOVÝ SYSTÉM PRO ODESÍLÁNÍ NOVINEK EMAILEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

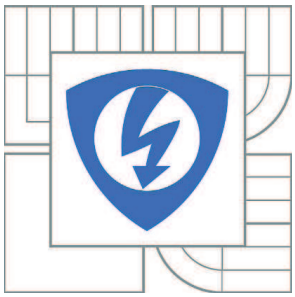
MICHAEL HORÁK

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

INTERNETOVÝ SYSTÉM PRO ODESÍLÁNÍ NOVINEK EMAILEM

INTERNET SYSTEM FOR SENDING NEWS BY E-MAIL

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

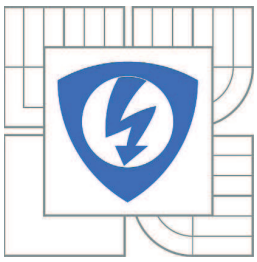
AUTOR PRÁCE — MICHAEL HORÁK

AUTHOR

VEDOUCÍ PRÁCE — Ing. ONDŘEJ PAVELKA

SUPERVISOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student: Michael Horák ***ID:*** 119436
Ročník: 3 ***Akademický rok:*** 2010/2011

NÁZEV TÉMATU:

Internetový systém pro odesílání novinek emailem

POKYNY PRO VYPRACOVÁNÍ:

Cílem bakalářské práce je návrh konceptu a realizace internetového systému pro správu emailových kontaktů a rozesílání novinek zájemcům podle zadaného klíče. K vytvoření systému využijte technologie PHP popřípadě MySQL. Systém by měl obsahovat editor textů - WYSIWYG, fotogalerií, import a export kontaktů z používaných formátů dat. Kontakty by mělo být možné rozřazovat do skupin, označovat konkrétní kontakty. Dále bude systém archivovat všechny odeslané newslettery, bude možné kopírovat použité části. Systém bude umožňovat přihlášení a odhlášení odběru.

DOPORUČENÁ LITERATURA:

[1] Lacko, L. : PHP 5 a MySQL 5 - Hotová řešení, Computer Press, 2007. ISBN 978-80-251-1695-1

Termín zadání: 7.2.2011 ***Termín odevzdání:*** 2.6.2011

Vedoucí práce: Ing. Ondřej Pavelka

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

HORÁK, M. *Internetový systém pro odesílání novinek emailem.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 64 s. Vedoucí bakalářské práce Ing. Ondřej Pavelka.

Abstrakt

Tato bakalářská práce se věnuje návrhu a realizaciinternetového systému pro odesílání novinek e-mailem. Zejména se zaměřuje na práci s kontakty a jejich seskupování do skupin, editaci předpřipravených textů a samotné odesílání elektronických zpáv.

V teoretické části jsou rozebrány vlastnosti jazyka PHP. Dále jsou zde popsány metody vytvoření webových formulářů, předávání dat v nich zadaných a následně jejich zpracování. Blíže jsou zde rovněž popsány funkce jazyka PHP, zejména pro práci s textem, programovací struktury a podmínky. Dále jsou popsány funkce pro odesílání e-mailů, funkce pro vytvoření hashů a funkce spravující relace.

V praktické části jsou nejdříve popsány problémy obecně. Následně jsou tyto funkce rozebrány podrobněji, je popsáno, jak bude skript probíhat a jaké funkce PHP se při tom použijí.

Klíčová slova

PHP, e-mail, databáze, HTML, funkce, kontakt, skupina

Abstract

The thesis deals with project lay-out and realization of the internet system dedicated to sending newsletters via e-mail. It especially aims at working with contacts and putting them in groups, editation of pre-prepared texts and sending the e-mails themselves.

In the theoretical part, there's described main traits of PHP language. There's also described methods of creating the web forms, handing over their inputs and processing of that data. Next, there's closer look on the PHP functions, especially for text processing, programming structures and conditions. Other functions, that can be found in this thesis described, are meant for sending the e-mails, creating hashs and managing sessions.

In practical part, the problems are described generally. In the next part of a practical realization, this methods are looked closer upon and described more thoroughly. Also there's defined how does the scripts work and which functions are used to make it work.

Key words

PHP, e-mail, database, HTML, function, contact, group

Prohlášení o původnosti práce

Prohlašuji, že svou semestrální práci na téma internetový systém pro odesílání novinek e-mailem jsem vypracoval samostatně pod vedením vedoucího diplomové práce s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny uvedeny v seznamu literatury na konci práce. Jako autor uvedené semestrální práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

. . . . . .
V Brně dne

. . . . . .
podpis autora

Obsah

SEZNAM OBRÁZKŮ

1. Úvod

Internet a zejména jeho součást web, se staly už téměř neoddělitelnou součástí moderní společnosti. Stejně tak se stal programovací a skriptovací jazyk PHP součástí webů. Zde se tento jazyk stará o samotnou logiku a funkci některých webových aplikací. V této práci se z nepřeberného množství využití php, mimo jiné, objeví hlavně dynamické generování formulářů, zpracování údajů v nich zadaných, přístup k adresářům a souborům na straně serveru a samozřejmě odesílání e-mailových zpráv.

V zadání práce je vytvořit internetový systém pro odesílání novinek a správu kontaktů a jejich rozřazování do skupin. Systém tedy bude muset umět kontakt přidat do databáze, přiřadit ho do uživatelem vytvořených skupin a samozřejmě ho při nepotřebě i z databáze odstranit. Dále by měla být možnost se přihlásit k odběru novinek i mimo tento systém. Pro samotné jádro systému, odesílání e-mailu má už jazyk PHP předpřipraveny funkce.

V další, pořadí druhé kapitole se budu krátce věnovat jazyku php, jeho historii, důvodům, proč ho používat, ale i jeho nevýhodám. V třetí kapitole jsou popsány hlavně funkce jazyka PHP, které mohou být a ve většině případů i jsou v této práci využité. Mimo to tu ale je i krátká vsuvka z jazyka HTML, když je popsána jedna z jeho součástí, formuláře. Čtvrtá kapitola obsahuje návrh řešení jádra systému. Zde se čtenář setká s popisem databází, a funkcí pro práci se skupinami a kontakty. V páté kapitole se již zabývám praktickým řešením kapitoly předchozí. V kapitole šesté je uveden podrobný návod pro zprovoznění a obsluhu systému. Pak již následuje závěr, kde je zhodnoceno, jak byl výsledek funkční a co by ještě šlo vylepšit. V dodatcích je uveden výpis celého zdrojového kódu.

2. JAZYK PHP

Jak je uvedeno v [4,6], PHP je programovací a skriptovací jazyk, který vnáší dynamičnost do jinak statických stránek, napsaných ve značkovacím jazyce HTML. PHP kód ve vyžádané stránce se zpracuje na straně serveru a ke klientovi se zašle pouze výsledek skriptu. Tento jazyk však lze použít v příkazové řádce, či v něm lze naprogramovat aplikace s grafickým uživatelským rozhraním (GUI), pracující na straně klienta.

PHP je v dnešní době podporováno většinou serverů, operačních systému i jejich platforem a existuje pro něj mnoho rozšíření. Je šířeno zcela zdarma pod speciální PHP licencí, dle korporace Free Software Foundation se jedná o „copyleft svobodný software" [7], dle organizace Open Source Initiation je PHP pod open source licencí. Copyleft znamená to, že jakýkoli software odvozený z rodičovského, bude mít stejné licenční podmínky jako rodič, tedy to, že z open source PHP nejde vytvořit placený software.

Historický vývoj

Vývoj PHP [6] začal v roce 1994, kdy jeho tvůrce Rasmus Lerdorf vytvořil několik skriptů pro programovací jazyk, které nazval „Personal Home Page/Form Interpreter", zkráceně PHP/FI. Dnes se však pod pojmem PHP setkáme spíše se spojením slov Hypertext Preprocessor. PHP/FI bylo ve verzi 1.0 vydáno 8. června 1995, verze 2 byla vydána o dva roky později. V témže roce dva izraelští vývojáři, Zeev Suraski a Andi Gutmans, započali práci na třetí verzi PHP, která byla oficielně vydána v červnu roku 1998. Od té se ve zkratce přestává uvádět část „/FI".

Po vydání třetí verze začali její tvůrci s přepisem samotného jádra, což vyústilo k vydání Zend engine. Jedná se o open source skriptovací jazyk, který se od svého vydání v roce 1999 stal nedílnou součástí PHP. Čtvrtá verze PHP byla vydána 22 června roku 2000. Vývoj PHP4 byl však už ukončen.

Nejnovější pátá verze PHP využívá ke svému chodu Zend Engine II. generace. Od roku 2008 je to jediná stabilní verze, která je nadále vyvíjená. Momentálně jsou na serverech podporovány dvě verze, 5.2.14 a 5.3.3, obě jsou k dispozici od 22. červenci 2010.

Nová verze PHP je ve vývoji paralelně s pátou již několik let. Vzhledem k velkému množství změn oproti předchozím verzím, měla být vydána jako PHP6. Jednou z nich měla být plná podpora kódování Unicode. To by mělo do budoucna dovolit pojmenovávat řetězce, třídy, metody a funkce i znaky, které nejsou součásti ASCII tabulky znaků. Implementace této úpravy se však ukázala obtížnou. Vývoj uvázl na mrtvém bodě a od března 2010 se začala hledat nová řešení tohoto problému.

Vybrané vlastnosti

- Zápis programu v PHP se uzavírá mezi párové značky <?php a ?>.
- Typ proměnné se nemusí definovat, určí se po přiřazení hodnoty.
- V souvislosti s předchozím bodem, má PHP více operátorů porovnání.
    - Operátor „==“ před porovnáním hodnot přetypuje proměnné na stejný datový typ.
    - Operátor „===“ vrátí shodu pouze tehdy, je-li obsah porovnávaných výrazů stejný a zároveň jsou stejného datového typu.
- Pole může obsahovat ve svých prvcích různé datové typy.
- Počet dimenzí pole/matic není omezen.
- Indexem pole nemusí být pouze číslo, lze použít i řetězec nebo logickou hodnotu.
- V řetězcích uzavřených do uvozovek se proměnné nahradí jejich hodnotou. Uzavřeme-li řetězec do apostrofů, zůstane proměnná jako její název.

Výhody a nevýhody PHP

2.1.1 Výhody

- Multiplatformnost a přenositelnost, většina funkcí funguje pod oběma hlavními serverovými operačními systémy (Linux a Microsoft Windows).
- Velmi rozšířené mezi hostingovými službami.
- Specializace na webové stránky.
- Přes pět a půl tisíce funkcí v základní knihovně, další lze získat.
- Podpora mnoha databázových systémů (například MySQL).
- Většinou kvalitně zpracovaná dokumentace.
- Velké množství zdarma použitelných kódů.

Nevýhody

Některé z níže napsaných nevýhod mají být řešeny v PHP6. Do té doby ovšem do tohoto výčtu patří.

- V základní instalaci chybí ladící nástroj.
- Pojmenování funkcí není unifikované, například je možné se setkat s funkcí pro práci s řetězci `strpos()`, ale i `str_replace()`.
- S předchozím bodem souvisí často nejednotné pořadí parametrů funkcí.
- Jazyk je popsán pouze implementací, nikde není definován.
- Slabá podpora Unicode, je potřeba použít speciální knihovnu
- Malé procento funkcí je vázáno na operační systém.
- Obsahuje funkce, které mohou být potenciálním nebezpečím.

3. POPIS ZPŮSOBŮ ŘEŠENÍ

Zadávání dat

První z funkcí, kterou bude muset systém mít, je zadávání uživatelských dat. Tuto interaktivitu zajišťují webové formuláře, které se definují pomocí jazyka HTML. Formulář se zobrazí na straně klienta, který do něj zadá příslušná data. Ty se následně odešlou skriptu, zadanému v deklaraci formuláře. V rámci této práce to budou PHP skripty.

Deklarace formuláře

Formulář je uzavřen mezi tagy (značky) `<form>` a `</form>`. Má několik atributů, konkrétně to jsou `name`, `action`, `method`, `enctype` a `target` [1]. Syntaxe celé deklarace je uvedena níže.

```
<form name=”nazev_formulare” action=”cil” method=”metoda”
enctype=”zpusob_kodovani” target=”vysledna_stranka”>

    tělo fomuláře

</form>
```

`Name` specifikuje název formuláře. Není to povinný atribut.

Atribut `action` specifikuje to, co se bude se zadanými daty dělat. Jeho obsahem je adresa zpracovávacího skriptu, ať už ve formě url, či relativní adresy k adrese formuláře. Není-li uveden, pak se data odešlou zpět na stránku formuláře.

Atribut `method` je spojen s předáváním dat z formuláře do zpracovávacího skriptu a rozepíši se o něm v části 3.1.3 zpracování zadaných dat.

Další dva atributy, `enctype` a `target` nebudu v této práci využívat. `Enctype` slouží k určení způsobu zakódování výstupních dat. Pro mé účely bude stačit nechat přednastavenou hodnotu. Atribut `target` je zodpovědný za to, ve kterém okně se otevře odkazovaná stránka. Dle nastavené hodnoty se tak může stát v novém okně, ve stejném okně, v nadřazeném okně, používáme-li rámy, a v novém okně plné velikosti. Pořadí hodnot pro tento atribut vzhledem k předchozí větě je `_blank`, `_self`, `_parrent` a `_top`. Oba dva parametry nejsou povinné.

Prvky formulářů

Prvek formuláře deklarujeme nepárovým tagem `<input>`, popř. `<select>` a `<textarea>`. Prvek `input` je definován jménem, hodnotou, zarovnáním, pro čtení a zapnutím. Poslední dva atributy nemusí ve všech prohlížečích fungovat korektně.

Jméno prvku je definováno parametrem `name`. Tento parametr se odesílá spolu se zadanými daty a hraje důležitou roli pro zpracování zadaných dat. Význam atributu hodnota souvisí s typem prvku input, viz níže. Zatímco u textových polí nemusí být zadán a určuje přednastavenou hodnotu, u prvků jako checkbox či radio udává přímo hodnotu pro cílový skript. Tento atribut se definuje za klíčovým slovem `value`. Atribut zarovnání, `allign` definuje polohu prvku formuláře stejně jako u obrázku.

Dále je prvek input definován typem. Pro přehlednost jsou tyto typy uvedeny v tabulce 3.1.

Tab. 3.1: Typy input polí

Typ	Popis	Rozšiřující atributy
text	Pole, do kterého se zadává text.	size=[číslo] šířka pole ve znacích, maxlength=[číslo] maximální šířka zadaného řetězce, autocomplete=[on/off] povolení automatického doplňování známých hodnot
password	Textové pole, které zadané znaky nahrazuje například hvězdičkami.	size maxlength
checkbox	Zatrhávací pole.	checked - pokud je zadaný, checkbox se zobrazí jako zaškrtnutý
radio	Přepínací tlačítko. Z několika prvků tohoto typu se stejným jménem, ale různými hodnotami může být zatržen vždy pouze jeden.	Checked
file	Umožňuje vybrat soubor, který se odešle na server.	accept = MIME_type specifikuje typ souboru, který bude odeslán na server
hidden	Pole, které uživatel neuvidí. Přenáší	

	přednastavenou hodnotu.	
submit	Tlačítko, po jehož stisknutí se formulář odešle .	
image	Taktéž potvrzující tlačítko. Navíc se odesílá i souřadnice kliknutí.	src=URL obrázku
button	Tlačítko, kterému lze přiřadit nějaký skript, nejčastěji je spjato s javascriptem.	
reset	Tlačítko, které po stisknutí nastaví všechna pole na původní hodnotu.	

Níže je uvedená syntaxe pro definici vstupního prvku formuláře.

```
<input type=”typ” name=”jmeno” value=”hodnota” dalsi_parametry >
```

například:

```
<input type=”text” name=”Prijmeni” value=”” size=”20” >
```

Tato definice na příslušné stránce zobrazí textové pole, které je pojmenované Prijmeni, což uživatel nevidí, není v něm nic předepsané a fyzicky je široké 20 znaků. Pokud by bylo nastaveno `value="Zadejte příjmení"`, pak by po načtení stránky toto pole nebylo prázdné, ale obsahovalo by „Zadejte příjmení“.

Zpracování zadaných dat

Po stisknutí formulářového prvku typu submit se vyplněná data z formuláře odešlou cíli, který je specifikován v deklaraci formuláře v prvku action. Způsob, jakým se data odešlou závisí na tom, jestli jsou v konfiguračním souboru `php.ini` povoleny globální proměnné a hodnotě atributu method nastavené v počáteční značce webového formuláře.

Obecně se nedoporučuje povolovat globální proměnné. Jedná se o potencionální bezpečnostní riziko. Od verze 4.2.0 jsou ihned po instalaci zakázané a do budoucna, pravděpodobně od verze PHP 6 se plánuje zrušit tuto možnost odesílání dat úplně. Pro úplnost je níže uveden zápis.

```
$promenna = $jmeno;
```

Kde „jmeno“ je hodnota nastavená v atributu name HTML prvku input. V tuto chvíli se do hodnoty promenna uloží obsah prvku jmeno.

Atribut method může nabývat dvou hodnot, konkrétně post a get. Rozdíl mezi nimi je v tom, že použijeme-li hodnotu get, odesílají se data z formuláře jako součást adresy URL, zatímco při hodnotě post se data posílají skrytě. Byť při tomto atributu není možné simulovat práci formuláře v prohlížeči pouhým měněním hodnot v adresním řádku. Obecně se doporučuje nastavit právě tato hodnota, zvláště pokud je odesílaných dat více.

Po odeslání dat zbývá jejích přijmutí v cílovém skriptu. Pro to existuje v PHP několik možností. Byly-li použity metody odeslání post a get lze použít zápisu:

```
$promenna = $_GET ["jmeno"];
```

případně

```
$promenna = $_POST ["jmeno"];
```

Další možnost je použít univerzální metodu $_REQUEST. Na straně formuláře zůstává nastaveno v atributu method buďto post nebo get, na straně skriptu lze však použít právě této univerzální metody.

Programovací struktury

Snad každý programovací jazyk podporuje větvení programu a opakování výpočtů v cyklech. Protože toho budu též využívat, jsou níže popsány základní programovací struktury jazyka PHP a jejich syntaxe [4, 5].

Pokud tělo podmínky, případně cyklu obsahuje víc než jeden řádek kódu, musíme ten uzavřít do složených závorek.

Zatímco skript napsaný v PHP vložíme do html kódu snadno, jen ho uzavřeme do párových značek, kód v HTML už tak snadno do PHP kódu vložit nelze. Blok PHP kódu musíme přerušit a kýžený HTML kód se vloží do vzniklé mezery. Problém to začíná být, pokud je potřeba vložit HTML kód do PHP cyklu nebo podmínky. Takovýto cyklus se přeruší normálně, je ale potřeba ho řádně ukončit. Pro tento účel se používá klíčového slova end + název cyklu/podmínky. Příklad pro cyklus for je uveden níže.

```
<?php
for (inicializace cyklu){
```

```
PHP část cyklu }?>
html část
<?php endfor; ?>
```

Cykly

Základními cykly v jazyce PHP jsou `do while`, `while` a `for`. Základní rozdíl mezi nimi je v tom, kdy se provede tělo cyklu a kdy se vyhodnotí podmínka, případně podmínky. Cyklus se provádí tak dlouho, dokud je podmínka pravdivá.

U cyklu `do while` se provádí nejdříve příkazy a až potom se vyhodnotí podmínka. Z toho plyne, že tento cyklus se provede vždy minimálně jednou.

Syntaxe cyklu `do while`:

```
do{
tělo cyku
}
while (podmínka);
```

U cyklů `while` a `for` se nejdříve vyhodnotí podmínka a pokud je pravdivá, provede se tělo cyklu. Zatímco cyklus `while` se užívá hlavně, pokud neznáme přesný počet opakování, u cyklu `for` se většinou počet opakování zadává předem.

Zápis cyklu `while`:

```
while (podmínka){
tělo cyku
}
```

Zápis cyklu for je o trochu složitější, skládá se ze tří částí. Část inicializace proměnné určuje to, od jakého čísla se začne cyklus provádět. Část podmínka udává jak dlouho, případně kolikrát se cyklus bude provádět a v části operace se zpravidla operuje opět s proměnou z první části. Zajistí se zde to, aby byla v budoucnosti splněna podmínka.

```
for(inicializace proměnné; podmínka; operace){
tělo cyklu}
```

Většinou cyklus for vypadá nějak takto:

```
for(i = 0; i<10, i++){
tělo cyklu}
```

Cyklus se provede desetkrát, pro i od 0 do 9.

Podmínky

Podmínky se všeobecně v programových strukturách používají k větvení a určují, jak se bude program dále vyvíjet. I podmínky v PHP jsou velmi podobné ostatním jazykům. Používají se především `if` a jeho rozšíření pomocí `else` a `elseif` a `switch`.

Princip funkce příkazu `if` je podobný cyklům. Stejně jako ony se provádí pouze tehdy, je-li zadaná podmínka pravdivá. Na rozdíl od nich ale proběhne pouze jednou. Dodatečná podmínka `elseif` rozšíří celý skript o další možnosti zpracování. Přidává další podmínky, které mohou vstupní data splňovat. Těchto rozšiřujících podmínek může být více. Rozšíření `else` způsobí to, že pokud není žádná z podmínek pravdivá, provedou se příkazy uvedené až za tímto klíčovým slovem. Zápis této podmínky je uveden níže.

```
if (podmínka){
příkaz 1;}
elseif(podmínka 2){
příkaz 2;}
else{
příkaz 3;}
```

Příkaz `switch` je svou prací podobný příkazu `if` s několika rozšířeními `elseif`. Tento příkaz je vhodné použít, bude-li pro rozhodovací skript jen několik málo vstupních hodnot.

```
switch (proměnná) {
case "hodnota1" : proces1; break;
case "hodnota2" : proces2; break;
default : proces3;
}
```

Řízení běhu cyklů a podmínek

Cykly a podmínky můžeme řídit dvěma způsoby. Prvním z nich je příkaz `break`. Ten slouží k ukončení běhu skriptu v době, kdy ještě následují příkazy. Použije-li se příkaz `continue`, tak se průběh skriptu vrátí na začátek a začne se odtud zase zpracovávat.

```
While (podmínka1){
Příkazy1;
If (podmínka2) break;
If (podmínka3) continue;
Příkazy2;}
```

Dokud bude platit podmínka1, tak se bude provádět celý cyklus while. Ve chvíli, kdy začne platit podmínka2 se cyklus po provedení příkazů1 násilně ukončí. V momentě, kdy platí zároveň podmínka1 a podmínka3 se provedou příkazy1 a následně se začne cyklus zpracovávat zase od začátku. Vyhodnotí se podmínka1 a je-li pravdivá, pokračuje se příkazem1.

Práce s textovými řetězci

Definice řetězců

Řetězec je libovolná posloupnost znaků, od jednoho znaku, až po několik vět. Aby byla skupina znaků rozeznána jako text, vkládá se mezi apostrofy (' ') nebo do uvozovek (" "). Zpracování řetězce závisí na tom, mezi jaké znaky ho uzavřeme, více je o tom už napsané v bodě 2.2. Některé znaky nelze přímo zobrazit. Před takové se musí umístit zpětné lomítko (\). Tyto znaky jsou uvedeny v tabulce 3.2.

Tab. 3.2: Speciální znaky

Znak	Popis
\n	Přechod na nový řádek
\r	Konec řádku
\t	Tabulátor
\\	Zpětné lomítko
\$	Symbol dolaru
\“	Uvozovka
\[0-7]{1,3}	Číslo v osmičkové soustavě

\x[0-9A-Fa-f]{1,2}	Číslo v šestnáctkové soustavě

Funkce pro výpis řetězců

Vzhledem k tomu, že valnou většinu informací z webových stránek vstřebá člověk z textu, má PHP pro práci s ním mnoho funkcí. První věc, co je potřeba je text vypsat. K tomu jsou určeny funkce `print()` a `echo()`. Rozdíl mezi nimi je ten, že funkce print vypíše pouze jeden textový řetězec, zatímco funkci echo lze prostřednictvím parametrů předat textových řetězců víc.

```
Print (string argument)
```

```
Echo (string argument1, string argument2 …[, string argumentN])
```

Pokud se přiřazuje text do proměnné, lze opět použít apostrofů i uvozovek. Pokud je potřeba spojit obsah proměnné s jiným řetězcem, použijí se složené závorky nebo operátor tečka (.). Ta se použije i při spojování řetězců do jednoho.

```
$promenna = "text";
Echo ("${promenna}ová zpráva"."<br />");
Echo("Zobrazíme"." "."$promenna"." "."uložený v proměnné
\$promenna"."<br />");
```

Výstup výše uvedeného kódu na webové stránce bude následovný.

```
textová zpráva
Zobrazíme text uložený v proměnné $promenna
```

Funkce pro práci s textovými řetězci

Jak jsem již dříve napsal, PHP má v sobě v základní instalaci zabudováno přes pět a půl tisíce funkcí. Mnoho z nich je stvořeno pro práci s textovými řetězci. Jedna z nejpoužívanějších je funkce, která vrátí délku načteného řetězce. Jedná se o funkci `strlen` a má následovný zápis.

```
Int strlen (string retezec)
$delka = strlen("ahoj");
```

V tuto chvíli se do proměnné delka uloží číslo čtyři datového typu integer. Další z používaných funkcí je vyhledávání textového podřetězce v jiném řetězci. K tomuto účelu slouží funkce strpos. Má tři parametry, jen dva z nich jsou ale povinné. Jsou to řetězec, ve kterém se bude vyhledávat a samotný vyhledávaný text. Pokud by se použil rozšiřující parametr start, tak se nastaví relativní pozice začátku hledání.

```
Int strpos (string řetězec, string vyhledánaný_text [, int start])
$text  = "Toto je text";
$pozice = strops ($text, "je")
```

Do promproměnné pozice se zapíše pozice počátečního znaku řetězce „je“. V zadaném příkladu to bude 5.

Někdy je potřeba z dlouhého textového řetězce vybrat podřetězec. K tomu slouží funkce substr. Stejně jako předchozí má tři rozšiřující parametry, povinné jen dva. První parametr udává řetězec, ze kterého se bude sub-řetězec extrahovat. Druhý parametr udává počáteční pozici vyjímaného řetězce a třetí parametr udává délku tohoto řetězce. Pokud je druhý parametr záporný, tak udává pozici od konce řetězce. Není-li třetí parametr použit, tak se od zadané pozice zkopíruje řetězec až do konce. Tato funkce je zejména vhodná k prohledávání textových souborů s přesně danou strukturou.

```
String substr (string řetězec, int start [, int délka])
$text  = "Jméno     Přijmení            XX";
        //0123456789012345678901234567890123456789
        //0         1         2         3
$vek = substr($text, 30, 2);
```

Zakomentovaná čísla pod řetězcem text značí pozice jednotlivých znaků v řetězci. Do proměnné vek se uloží řetězec, který začíná v řetězci text na třicáté pozici a je dva znaky dlouhý. Obě předchozí funkce do sebe slučuje substr_replace. Její zápis je následující:

```
String substr_replace (string původní, string náhrada, int start [,
int délka])
```

Tato funkce nahradí v řetězci původní na pozici start tolik zanků, jak je dlouhý řetězec náhrada, případně tolik znaků, kolik je určeno parametrem délka. Pokud ovšem je

potřeba nahradit v původním řetězci slovo jiným, jinak dlouhým, tak je tato funkce samostatně nepoužitelná. Bylo by potřeba několik dalších řádků programu, aby všechno fungovalo správně. Proto se používá funkce `str_replace`.

```
String str_replace (string původní, string nové, string text)
```

Tato funkce nahradí v řetězci `text` sub-řetězec `původní` za řetězec `nové`. Pokud je potřeba odstranit z řetězce zbytečné mezery a tabulátory, použije se funkce trim.

```
string trim (string řetězec)
$osekane = trim ("hodně     zbytečných          mezer     ");
```

Do proměnné `osekane` se uloží nový řetězec „hodně zbytečných mezer“. Je-li potřeba rozdělit řetězec na úseky o určité délce rozdělené oddělovačem použije se funkce `chunk_split`. Parametr `text` je rozdělovaný řetězec, `délka` je počet znaků, na které se řetězec rozdělí. Jako oddělovač je implicitně nastavena mezera, lze to ale nastavením parametru `oddělovač` změnit.

```
chunk_split (string text, int délka, string oddělovač)
echo(chunk_split ("abcdef", "3", #));
```

Funkce echo vypíše na obrazovku abc#def#.

Pokud je potřeba pracovat s poli, převédět jejich obsah na řetězce a naopak, tak se použijí funkce `explode, implode` a případně `join`. Funkce `implode` a `join` jsou co do funkce prakticky totožné. Liší se pouze pořadím vstupních argumentů. Ty jsou pole, které se bude rozdělovat a oddělovač, který bude oddělovat jednotlivé prvky. Vstupní argumenty funkce `explode` jsou oddělovač a řetězec, který bude rozdělen do prvků pole.

```
string implode(array pole, string oddělovač);
string join(string oddělovač, array pole);
array explode(string oddělovač, string řetězec);
$pole = explode("#","ab#cd#ef");
echo("$pole[1]");
```

Proměnná `pole` obsahuje tři prvky: ab, cd a ef. funkce echo vypíše prvek pole s indexem jedna, tedy prvek cd.

Je-li potřeba vzít dlouhý řetězec a zalomit ho do řádků určité délky, použije se funkce `wordwrap`. Má čtyři vstupní parametry. Jediný povinný je vstupní řetězec, který se bude dělit. Druhý je délka řádku ve znacích, není-li zadán, tak se použije hodnota 75. Třetí parametr je dělení řádku. Jedná se o znak či řetězec, který se zobrazí na konci každého zalamovaného řádku. Implicitně je nastaven jako „/n“. Poslední parametr nabývá hodnot logické jedničky a nuly. Je-li nastavená jednička, tak každý řádek bude mít právě nastavenou délku. Rozdělí se i když by to mělo být uprostřed slova.

```
string wordwrap (string řetězec, [, int šířka, string znak, bool
zalamování]);
```

Pokud je potřeba zašifrovat určitý řetězec, použije se funkce `crypt`. Ta má dva vstupní parametry. Povinný je řetězec, který je potřeba zašifrovat, nepovinný je další řetězec, podle něhož se vytvoří základ pro zašifrování. Ten většinou není delší než dva znaky. Výstupem funkce je řetězec, který má od vstupního rozdílnou délku. K zašifrování se implicitně použije šifra DES, lze však nastavit i jiné šifrovací algoritmy. Důležité je, že není žádná funkce, která by byla schopná ze zašifrovaného řetězce zjistit řetězec původní.

```
string crypt (string řetězec [, string základ]);
```

Někdy je i potřeba převést znak na ASCII hodnotu a naopak. K tomuto účelu slouží funkce `ord` a `chr`.

```
$a1 = (ord(a));
$a2 = (chr(97));
```

V proměnné a1 je uloženo 97, což je ASCII hodnota písmene a, funkce `chr` je reciproká funkci `ord`. To znamená, že převádí čísla z intervalu 32 až 126, což jsou tisknutelné ASCII znaky, na text.

Validace údajů a regulérní výrazy

Jsou-li zadávána data uživatelem, dost často je potřeba zkontrolovat jejich správné zadání či platnost. Ověření může probíhat na straně serveru, typicky PHP, či na straně klienta, kde se o to stará například javascript. Někdy lze zadání nesprávných údajů předejít. Pokud je možné implementovat nějaký mechanismus prevence zadání nesprávných údajů, je většinou vhodné tak učinit.

Regulérní výrazy jsou způsob, jakým ověřovat data v jazyce PHP. Určují masku, proti které se testuje zadaný řetězec. Testovaný řetězec tedy musí obsahovat znaky jako regulérní výraz. Regulérní výraz se může skládat z přímo zadaných znaků, zástupných znaků, takzvaných metaznaků a intervalů povolených (zakázaných) znaků [2,3,4,5].

Tab. 3.3: Zástupné znaky regulérních výrazů

Znak	Popis
^	Udává začátek řetězce.
$	Udává konec řetězce.
A	Pokud musí text obsahovat jeden určitý znak, tak ho do regulérního výrazu vypíšeme.
.	Slouží k nahrazení jednoho libovolného znaku.
?	Udává, že znak předcházející tomuto může ale i nemusí být v textu.
*	Znamená opakování znaků. Znak uvedený před tímto se v textu může libovolněkrát opakovat.
+	Význam má podobný jako předchozí, vyžaduje však minimálně jedno opakování.
[ ]	Znaky uvedeny v hranatých závorkách uvádí interval, do kterého musí znak patřit.Znaky lze vyjmenovat buď přímo nebo napsat první a poslední oddělený pomlčkou. Je-li první znak v závorkách ^,tak se obsah závorek neguje.
( )	Je-li potřeba zajistit opakování několika znaků, vypíší se do kulatých závorek.
{x,y}	Parametry x udává, kolikrát se minimálně musí v textu objevit výraz před složenými závorkami. Parametr y udává maximální počet opakování.
\	Je-li potřeba v textu vyhledat i některý z tohoto výčtu řídících znaků, uvádí se před něj zpětné lomítko.
|	Metaznak, rozdělující řetězec v regulérním výrazu na několik subřetězců. Testovaný výraz pak vyhoví, jestli obsahuje alespoň jeden z těchto subřetězců.
[[:>:]]	Metaznak ukazující na začátek řetězce.
[[:<:]]	Metaznak ukazující na konec řetězce.
[[:třída:]]	Testovaný výraz obsahuje jeden znak z třídy znaků.

Metaznak [[:třída:]], jak je napsáno v tabulce 3.3 obsahuje třídy znaků. Pro celkový výčet je jich mnoho, v tabulce 3.4 je uvedeno několik v této práci použitelných.

Tab. 3.4: Vybrané třídy metaznaků

Znak	Popis	Znak	Popis
Alphanum	Alfanumerické znaky (písmena anglické abecedy a číslice).	**print**	Jakékoliv tisknutelné znaky.
Alpha	Písmena anglické abecedy	**punct**	Interpunkční a další znaky (závorky, hvězdička, zavináč atp.).
Digit	číslice	**space**	Mezera, ale i tabulátor, nová řádka či stránka.

Už jen zbývá popsat funkce, pro práci s regulérními výrazy. K tomuto účelu existují dvě skupiny funkcí, obě mající podobné užití. První z nich je skupina PCRE, druhá POSIX. Ta však od PHP verze 5.3.0 přestává být podporována. Proto se v této práci budu věnovat hlavně skupině PCRE.

Knihovna funkcí PCRE implementuje regulérní výrazy se stejnou syntaxí a sémantikou, jako má jazyk Perl s drobnými změnami. Samotný výraz, proti kterému se řetězec testuje, musí být uzavřen v oddělovacích znacích. Těmi nesmí být alfanumerické a prázdné (bílé) znaky s výjimkou zpětného lomítka. Jako oddělovací znaky se často používají závorky. Pokud je potřeba použít v regulérním výrazu právě tohoto oddělovacího znaku, píše se před ním zpětné lomítko. Za koncovým oddělovacím znakem lze použít modifikátory, které mají vliv na porovnávaní. Některé z nich jsou uvedeny v tabulce 3.5.

Tab. 3.5: Modifikátory vzoru

Modifikátor	Popis
I	Způsobí to, že se při porovnávání nebere ohled na velikost písma
M	Používá se, obsahuje-li načtený řetězec víc než jeden řádek. PCRE totiž defaultně považuje načtený řetězec za jednořádkový text. Pokud řetězec neobsahuje znaky pro nový řádek (\n) nebo vzor neobsahuje metaznaků ^ a $, pak tento parametr nemá žádný efekt.
S	Zástupný metaznak tečka zastupuje všechny znaky, i znaky pro nové řádky.
X	Bílé znaky jsou ignorovány, pokud nejsou v třídách znaků nebo v komentářích.

Pro práci s regulérními výrazy existuje v knihovně PCRE celkem devět funkcí. Pro účely této práce popíši však jen ty relevantní.

První z funkcí je `preg_match`. Tato funkce vrátí číslo, které je rovno počtu shod vstupního řetězce se vzorem. Toto číslo bude nula, pokud ke shodě nedošlo nebo jedna, pokud došlo k jedné či více shodám.

```
int preg_match (string vzor, string vstup [,array shody, int návěst,
int offset]);
```

Vzor je regulérní výraz, proti kterému se bude řetězec vstup testovat. Dalšími parametry je pole shod, návěst a offset. Pokud došlo ke shodě, pole shod má v prvku s indexem nula výraz shodný se vzorem. Prvek offset určuje, začátek hledání shody. Implicitně je nastavený na nulu, tedy vstupní řetězec se bude prohledávat od prvního znaku. Parametr návěst není pro mé účely užitečný, tak ho nebudu rozebírat.

Je-li potřeba dát všechny prvky pole, které se shodují se vzorem do nového pole, použije se funkce `preg_grep`.

```
array preg_grep (string vzor, array vstup [, int návěst]);
```

Řetězec vzor není potřeba popisovat. Význam má stejný, jako u předchozí funkce. Pole vstup je vstupní pole, jehož prvky se budou testovat proti vzoru. Pokud se jako návěst použije PREG_GREP_INVERT, tak bude výsledná matice obsahovat prvky, které se neshodují s regulérním výrazem.

Funkce preg_replace prohledá vstupní řetězec, pole či jinou proměnou a dojde-li ke shodě, pak se shodující se výraz nahradí zadaným výrazem.

```
mixed preg_replace (mixed vzor, mixed náhrada, mixed vstup [, int
limit, int počet]);
```

Mixed znamená to, že parametrem může být více typů dat. U této funkce to mohou být řetězec (string) či pole (array). Parametry vzor a vstup jsou už rozebrány výše, u této funkce mají stejný význam. Prvek náhrada nahradí při shodě prvek vyhovující regulérnímu výrazu vzor. Parametr limit udává, maximální počet náhrad pro každý vstupní prvek. Implicitně je tento parametr nastaven na -1, což znamená neomezený počet náhrad. Parametr počet, pokud je zadaný, tak po provedení funkce obsahuje počet provedených náhrad.

Funkce, která rozdělí prvky řetězce do pole podle regulérního výrazu je `preg_split`.

```
array preg_spilt (string vzor, string vstup [, int limit, int
návěst]);
```

Tato funkce je podobná funkci `split`. Parametry vzor a vstup mají opět stejný význam, jako u předešlých funkcí. Parametr limit udává, kolik se provede rozdělení. Implicitně je nastavený na -1, tedy vždy se provedou všechna rozdělení. Pokud by se mělo rozdělovat vícekrát, než je zadaný limit, pak se poslední část vstupního řetězce nerozdělí a uloží se do posledního prvku matice v celku. Parametr návěst může obsahovat až tři hodnot, které se dají kombinovat operátorem | (roura, pipe). První z nich je PREG_SPLIT_NO_EMPTY, při jehož zadání funkce `preg_split ()` nevrací prázdná místa. Při zadání druhého parametru, PREG_SPLIT_DELIM_CAPTURE funkce vrací i oddělující znaky. PREG_SPLIT_OFFSET_CAPTURE způsobí to, že funkce vrací pole o dvou rozměrech. V prvním prvku na daném řádku vrácené matice je uložen oddělený řetězec a v druhém prvku řádku je uložena pozice jeho prvního znaku v řetězci.

Funkce pro práci se soubory a adresáři

Pro práci se soubory a adresáři existuje v PHP mnoho funkcí, umožňující například otevření souboru či adresáře, zápis a čtení údajů. Někdy může být problém s přístupovými právy k souboru, zvláště na linuxových serverových distribucích [4,5]. Pak je potřeba k tomuto souboru správně nastavit práva, ať už přímo na serveru (například pomocí příkazu chmod) nebo už při nahrávání na server.

Funkce pro přístup k souborům

Má-li se s daným souborem, případně adresářem pracovat, je důležité, aby existoval. K ověření existence souboru slouží funkce `file_exists`.

```
bool file_exists (string název_souboru);
```

Ta, pokud soubor existuje, vrací logickou hodnotu true, při neexistenci false. Má jediný vstupní parametr a to je název souboru.

Pracujeme-li se složitějším souborovým systémem, je potřeba rozlišit, jestli je daný objekt adresář či soubor. K tomu jsou určeny dvě funkce. Pro soubory to je `is_file` a pro složky to je `is_dir`.

```
bool is_file (string název_souboru);
bool is_dir (string název_složky);
```

Funkce `is_file` vrací logickou hodnotu true, pokud je testovaný objekt soubor. Pokud se jedná o neexistující objekt či složku, pak vrací logickou nulu. Význam funkce `is_dir` je obdobný.

Další funkce rozšiřuje použitelnost předchozích dvou. Funkce `opendir` vrací deskriptor adresáře použitelný v dalších funkcích nebo logickou hodnotu false při neúspěchu. Logické hodnota false ovšem může být reprezentována i ekvivalentními nelogickými hodnotami. Například prázdným řetězcem či číslem nula. Na to je potřeba dát pozor zvláště při porovnávání hodnot pomocí operátoru ===, viz kapitola 2.

Deskriptor je popisovač, který v sobě nese informaci umožňující získat přístup k identifikovatelnému prostředku. Pro každý soubor je potřeba mít vlastní deskriptor [4].

```
int opendir (string cesta);
string readdir (int deskriptor_adresáře);
void closedir (int deskriptor_adresáře);
```

Funkce `readdir` vrací název dalšího souboru z adresáře, k vypsání všech je proto potřeba použít cyklu. Po ukončení práce se soubory či složkou se deskriptor uvolní použtím funkce `closedir`.

Funkce pro práci s obsahem souboru

Abychom mohli pracovat s obsahem souboru, musíme ho nejdřív otevřít funkcí `fopen`.

```
int fopen (string název_souboru, string režim [, bool
použít_include_path]);
```

Pomocí parametru název_souboru se specifikuje soubor, který se touto funkcí otevře. Touto funkcí lze otevřít soubory na místním serveru ale i prostřednictvím protokolu HTTP či FTP. Argumentem režim se určí, způsob práce s obsahem souboru. Pro přehlednost jsou uvedeny v tabulce 3.6. Parametr použít_include_path může nabývat pouze logických hodnot, implicitně je nastaven na hodnotu false. Pokud se nastaví na true, tak se otevíraný soubor bude hledat v umístění specifikovaném v include_path. To se nachází v konfiguračním souboru php.ini.

Tab. 3.6: Režimy práce se souborem

Režim	čtení/zápis	Pozice	Popis
r	ano/ne	počátek	Soubor se otevře pro čtení, obsah se zachová.
r+	ano/ano	počátek	To samé jako „r“.
w	ne/ano	počátek	Otevře soubor pro zápis, obsah vymaže. Pokud soubor neexistuje, tak ho vytvoří.
w+	ano/ano	počátek	To samé jako „w“.
a	ne/ano	konec	Otevře soubor pro přidání (a – append – připojit), obsah zachová. Pokud soubor neexistuje, tak ho vytvoří.
a+	ano/ano	konec	To samé jako „a“.
b	-/-	-	Použije-li se jako parametr, pak se pracuje se souborem v binárním režimu. Implicitně se totiž soubory otevírají jako textové.

Do souboru se zapisuje pomocí funkce `fwrite`. Existuje ještě funkce `fputs`, která je pouze aliasem `fwrite`.

```
int fwrite (int deskriptor, string vstup [, int délka]);
int fputs (int deskriptor, string vstup [, int délka]);
```

Parametr deskriptor získáme otevřením souboru pomocí funkce `fopen`. Do souboru se zapíše řetězec specifikovaný parametrem vstup. Pokud je zadaný nepovinný parametr délka, tak se zapisování zastaví po zapsání tolika znaků, kolik je tímto parametrem definováno. Pokud je vstupní řetězec kratší, pak se zapisování ukončí po zapsání řetězce.

Pokud je potřeba ze souboru data přečíst, pak se použije funkce `fread`, případně `fgets`.

```
string fread (int deskriptor, int délka);
string fgets (int deskriptor, int délka);
```

Jejich chování je lehce odlišné. Funkce `freads` čte ze souboru do té doby, dokud nenarazí na konec souboru nebo řetězec není delší, než zadaný parametr délka. `Fgets` načte pouze jeden řádek. Limit daný parametrem délka je stejný jako u `fread`, jen zmenšený o jedna.

Pro účely čtení ze souboru je velmi použitelná funkce `feof`.

```
bool feof (int deskriptor);
```

Jejím výstupem je logická hodnota. Ta je rovna true, pokud při čtení dorazíme na konec souboru (end of file, EOF), nebo nastala chyba. Jinak vrací hodnotu false.

K uzavření souboru slouží funkce analogická ke `closedir`, `fclose`.

```
bool fclose (int deskriptor);
```

Funkce vrací hodnotu true při úspěchu a false při neúspěchu.

Při přístupu k souborům se obvykle volají funkce v pořadí `fopen`, `fwrite/fread a fclose`. Proto byly vytvořeny funkce, které tyto předchozí sdružují do jedné.

```
int file_put_contents (string název_souboru, mixed vstup [, int
návěsti]);
```

Tato funkce vloží do souboru určeného parametrem název vstupní data. Ty mohou být buďto řetězec (string), či pole (array). Nepovinný parametr návěst může obsahovat FILE_USE_INCLUDE_PATH, jehož význam je vysvětlen výše. Dále FILE_APPEND, což má stejný význam jako bychom soubor otevřeli v režimu „a“, případně „a+“. Dále je možné soubor zamknout pro zápis zadáním LOCK_EX. Tyto parametry lze opět spojovat symbolem roura.

```
string file_get_contents (string název_souboru [, bool
použít_include_path, int offset, int maxlen]);
```

Tato funkce načte se souboru specifikovaným parametrem název data. Lze použít parametr FILE_USE_INCLUDE_PATH. Parametr offset určuje počáteční pozici, od které se bude číst. Pokud není specifikován, je nastaven na -1, což znamená, že číst se bude od začátku. Parametrem maxlen se specifikuje délka načteného řetězce.

Vložení souboru do kódu PHP

Opakuje-li se často nějaká část kódu, nemusí být opětovně programátorem vypisována. Lze ji uložit do samostatného souboru a do kódu vložit dodatečně. Sníží to množství práce a zvýší přehlednost kódu.

K tomuto účelu slouží příkazy `include` a `require`. Oba mají jediný vstupní parametr a tím je název souboru, který se bude vkládat, případně cesta k němu.

```
include (string soubor);
```

```
require (string soubor);
```

Oba příkazy fungují podobně. Rozdíl mezi nimi je ten, že include generuje při chybě pouze warning, kdežto require fatal error. To se následně projeví na dalším zpracovávání skriptu. Je-li potřeba zajistit, aby se soubor vložil jen jednou, použije se `include_once` či `require_once`.

Odesílání elektronické pošty

K odesílání elektronické pošty slouží v PHP funkce mail. Ta umí odeslat e-mail i v html kompozici. Pro složitější poštu obsahující například obrázky se však doporučuje použít balíček funkcí z jazyka PEAR, PEAR::Mail_Mime [5].

```
bool mail (string příjemce, string předmět, string tělo [, string
další_záhlaví, string další_parametry]);
```

Je-li e-mail přijat k odeslání v pořádku, funkce vrátí logickou hodnotu true, v opačném případě false. Logická hodnota true však znamená pouze to, že e-mail by přijat k odeslání, nezaručuje to, že e-mail dorazí k adresátu. V parametru adresát specifikujeme příjemce e-mailu. Ten může být jeden, ale i více, přičemž jednotlivé e-mailové adresy oddělujeme čárkou. Formátování této adresy, případně adres musí splňovat specifika zadané v RFC 2822. Příklad správných adres je uveden níže. Parametr předmět specifikuje předmět e-mailu a musí splňovat požadavky kladené RFC 2047.

- jmeno@host.cz
- jmeno@host.cz, dalsi.jmeno@host.cz
- Jméno < jmeno@host.cz>
- Jméno < jmeno@host.cz>, Další jméno < dalsí.jmeno@host.cz>

Parametr tělo již obsahuje odesílanou elektronickou zprávu. Řádky této zprávy by neměly být delší než sedmdesát znaků a měly by být odděleny znakem pro nový řádek, /n. Volitelný parametr další_záhlaví je řetězec obsahující rozšiřující parametry záhlaví elektronické pošty. Typicky obsahuje prvky jako odesílatel a kopie. Pokud se použije více těchto prvků, rozšiřujících záhlaví je doporučeno oddělit je znaky /r/n. Jejich význam je blíže popsán v tabulce 3.2. Další_parametry se starají o ovládání funkce z příkazové řádky. Vzhledem k tomu, že to neplánuji ve své práci využít, se o tom nebudu hlouběji zmiňovat.

Relace - sessions

Protokol HTTP je sám o sobě bez stavový [9]. Aby bylo možné si předávat informace o stavu aplikace v rámci jednoho webu, byly od verze PHP 3 implementovány sessions, česky někdy nazývané jako relace.

Výše bylo popsáno, jak se předávají parametry z formulářů pomocí url i bez ní. Data přenášená pomocí relací, často proměnné, často není třeba přenášet mezi klientem a serverem, na rozdíl od vyplněných dat formulářů. Skripty na serveru tedy dostávají data přímo z interpretu jazyka PHP.

Základních funkcí pro práci s relacemi je několik. Funkce

```
session_start();
```

inicializuje relaci. Proměnná se zadá pomocí funkce

```
$_SESSION['promenna'];.
```

Pokud není proměnná nadále potřeba, lze ji uvolnit pomocí

```
unset($_SESSION['promenna']).
```

Relace se ukončuje pomocí

```
session_destroy();,
```

Tato funkce se využije například při odhlášení uživatele.

Pokud je při provádění funkce session_start() k dispozici identifikátor relace, pak lze s relací pracovat normálně, pokud není k dispozici, pak se identifikátor vytvoří. Proměnné lze do relace přidávat i z ní odebírat kdykoli relace běží. Při použití funkce seesion_destory() se při zrušení relace uvolní i všechny proměnné, které v ní byly zadány.

Hashovací funkce

Účel hashovací funkce je vytvořit ze vstupních dat jejich „otisk“. Tento otisk je u jednoho hashovacího algoritmu vždy stejně dlouhý řetězec [8]. Další z vlastností hashovacích funkcí je že malou změnou vstupních dat docílíme velké změny otisku, z otisku je prakticky nemožné získat původní data a je velmi nepravděpodobné, že dva různé vstupy budou mít stejný otisk.

PHP umožňuje spočítat jak hash textového řetězce, tak souboru. V této práci využívám pouze první, tak se budu věnovat pouze tomu. K výpočtu hashe lze využít dva algoritmy, MD5 a SHA1.

Funkce mají zápis:

```
string md5(string $str, [, bool $raw_output = false]),
```

či

```
string sha1(string $str, [, bool $raw_output = false]).
```

Kde parametr $str je řetězec, ze kterého se spočítá hash. Nepovinná proměnná $raw_output pokud je nastavená na hodnotu true způsobí to, že místo řetězce určité délky je výstupem funkce surový binární formát s jinou, zpravidla menší, délkou.

4. NÁVRH OBECNÉHO ŘEŠENÍ

V této kapitole se budu věnovat návrhu několika elementárních funkcí, ze kterých se bude skládat výsledný systém. Konkrétně to budou funkce pro práci se skupinami, funkce pro práci s kontakty a funkce, které budou schopny roztřídit kontakty podle příslušnosti ke skupinám. Obecně se všechny budou skládat ze dvou částí. První bude webový formulář starající se o interakci uživatele se systémem. Druhá část bude samotný skript, který zpracuje zadaná data a požadavky.

Funkce pro práci se skupinami

Databáze skupin bude uložena v textovém souboru. Na každém řádku bude jedna skupina. Práce se skupinami bude poměrně jednoduchá. Základní dvě operace budou přidání a odebrání skupiny.

Přidání skupiny

Pro přidání skupiny bude potřeba vložit její jméno. To obstará webový formulář a skript, který ho zpracuje. Tento skript bude muset umět z formuláře přebrat název skupiny a vložit ji do databáze. Databázi bude tvořit textový soubor, na jehož každém řádku bude název jedné skupiny. Nová skupina se přidá na konec tohoto textového souboru. S tím bude provázaná náležitost kontaktů k těmto skupinám. Po každé, když se skupina přidá, tak se objeví volba, jestli se k ní přidají všechny nebo žádný kontakt.

Dále zde bude volba, jestli bude skupina soukromá či veřejná. To, jaká skupina je bude uloženo v databázi skupin na stejném řádku, jako je název skupiny. Za oddělovací značkou bude buďto jednička nebo nula, dle stavu skupiny soukromá/veřejná.

Odebrání skupiny

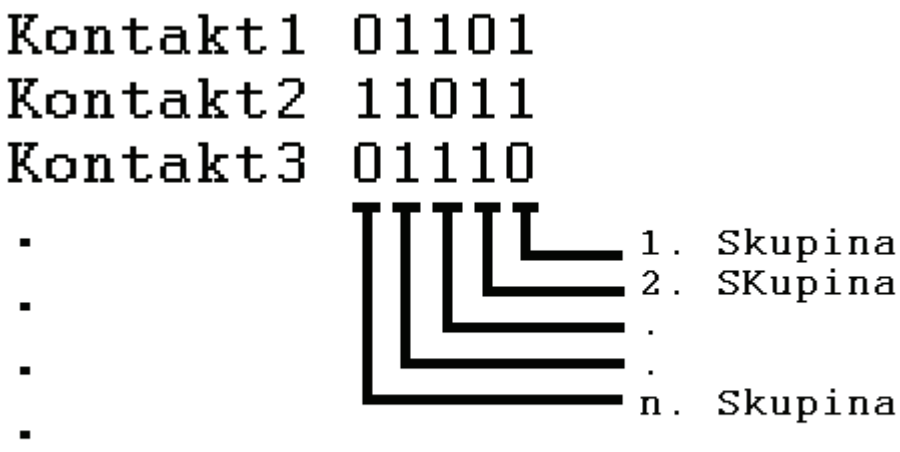
Odebrání skupiny bude probíhat tak, že se daná skupina odebere z databáze. S tím se ale musí provázat i odebrání příslušnosti kontaktů k ní. Proto je potřeba mít přehled o čísle skupiny. Plánuji číslovat skupiny, ale i kontakty podle řádků v textovém souboru. Odebrání bude probíhat opět skrz webový formulář.

Přejmenování skupiny

Přejmenování skupiny bude svou funkcí hodně podobné odebrání, s tím rozdílem, že se skupina neodebere, ale přejmenuje. S tím je pochopitelně spojeno i to, že se v tomto případě nebude dělat nic s příslušností kontaktů přejmenovávané skupině.

Funkce pro práci s kontakty

Jako u skupin, bude databáze kontaktů v textovém souboru. Kontakt samotný bude složen ze dvou částí. První z nich je jeho e-mailová adresa. Druhá bude řetězec nul a jedniček. Ten bude mít přesně tolik znaků, kolik je definovaných skupin, přičemž pořadí každého znaku je svázáno s pořadím skupiny. Bude-li kontakt přiřazený ke skupině, bude na příslušné pozici jednička, v opačném případě nula.



Obr.: 4.1 Kontakt a pole skupin

Přidání kontaktu

Než začne být systém použitelný, bude potřeba naplnit jeho databázi kontakty. To půjde dvěma způsoby. Buďto se o to postará sám uživatel přímo v systému nebo se přihlásí zájemce o odběr novinek. V prvním případě se bude vstupní formulář skládat z několika částí.

- pole pro adresu kontaktu
- pole pro přiřazení ke skupinám

Pole pro adresu není potřeba komentovat. Pole pro přiřazení kontaktů bude dynamicky generováno systémem a bude mít tolik vstupních prvků, kolik je momentálně zadaných

skupin. Vstupní prvky se budou skládat ze zatrhávacího tlačítka, takzvaného checkboxu a názvu skupiny. Zatrhnutím checkboxu se kontakt do dané skupiny přiřadí.

Zde leží hlavní rozdíl mezi přidání kontaktu uživatelem a zájemcem o newsletter. Zájemce bude mít pouze jedno vstupní pole pro zadání své e-mailové adresy. Ta se navíc bude testovat proti regulérnímu výrazu, jestli je platná. Takto zadaný kontakt bude mít v databázi příslušnost pouze ke skupině, která bude sdružovat zájemce o newslettery.

Odebrání kontaktu

Odebrání kontaktu bude opět probíhat na dvou rovinách. Odebrání bude moci spustit uživatel, či zájemce.

Uživatel bude mít naprostou volbu ve volbě odebíraného kontaktu. Systém musí rozpoznat odebíraný kontakt a vymazat ho z databáze. Tato funkce je velmi podobná funkci určené k odebírání skupin.

Pokud si zájemce o zasílání novinek nebude nadále zasílání přát, bude mít k dispozici rozhraní k vypnutí této funkce. V tomto rozhraní se zadá adresa, na kterou se novinky odesílají, a odešle se zpracujícímu skriptu. Ten projde databázi kontaktů a najde-li shodu, pak kontakt vymaže. Zde je potřeba pojistit, aby byl smazán pouze kontakt zadaný zájemcem. Pokud byl zadán uživatelem, tak se jeho příslušnost ke skupině newsletter nastaví na nulu, ale v databázi zůstane.

Dodatečné přiřazení kontaktu ke skupinám

Program musí počítat i s tím, že kontakt může přestat být potřebný v určité skupině a naopak v některé scházet. Proto musí umět kontaktu přiřadit a odebrat náležitost ke skupinám dodatečně. Uživatelské rozhraní této funkce bude velmi podobné tomu, kdy se kontakt přidává. Rozdíl bude v tom, že v tuto chvíli má už kontakt přiřazenu alespoň jednu skupinu. To se musí odrazit i v tom, že vstupní pole, které bude mít na starosti přiřazování kontakt-skupina, bude mít určité prvky předvyplněné podle náležitosti kontaktu k těmto skupinám.

Funkce pro správu předpřipravených textů

Jednou z hlavních funkcí celého programu, je možnost, předpřipravit si texty, ze kterých bude možné skládat obsah těla odesílaných e-mailů. Texty budou opatřeny názvem pro jednodušší rozlišení. Text bude možné přidat, editovat a smazat.

Přidání nového textu

Při přidání textu se vytvoří formulář se dvěma vstupními poli. Jedno bude standardní textové a bude se do něj zadávat popis textu. Druhé bude tvořené HTML prvkem <textarea> a do něj se bude zadávat samotný text pro vložení do e-mailu.

Po odeslání formuláře načte skript zadaná data, opatří je oddělovacími značkami a uloží do souboru.

Editace a mazání textů

Nejdříve se otevře soubor s již zadanými texty a všechny se načtou. Pak se rozdělí dle oddělovacích značek na jednotlivé popisky a texty. Ty se pak vloží do nově vygenerovaných formulářů jako předvyplněné hodnoty. Dále ve formuláři bude skrytý vstupní prvek, který zpracovávacímu skriptu předá pozici mazaného textu. Každý z těchto formulářů bude mít možnost text editovat či smazat.

Při obou akcích se nejdříve načte do pole celý obsah souboru, ve kterém jsou texty uloženy. Následně se v prvku pole na pozici, která byla předávána skrytým vstupním prvkem, provede buďto smazání či editace.

Při editaci se celý text z editovaného prvku pole nahradí novým. Pokud se prvek maže, pak se celé pole přeskupí tak, že se prvky následující mazaného posunou o jedno místo směrem k začátku pole.

Funkce pro odesílání e-mailu

Funkce pro odeslání budou základní dvě. Odeslání pro jednoho příjemce a pro celou skupinu, případně skupiny. Funkce pro jednoho příjemce by měla být schopna načíst kontakt z databáze a odeslat mu zadaný text. Pokud bude více příjemců, pak se bude muset při odeslání použít rozhodování a cyklus. Rozhodování zjistí, jestli kontakt patří do kýžené skupiny a cyklus to zjistí u všech kontaktů v databázi. Všechny kontakty, které splňují podmínku, se uloží do pole. Z tohoto pole se pak budou načítat při samotném odesílání elektronické zprávy.

5. PRAKTICKÉ ŘEŠENÍ NÁVRHU

V této kapitole se budu věnovat návrhu kódu, jehož funkce je popsána v kapitole předchozí. Kód systému uvádím jen ve výjimečných případech, nechci jím zde zbytečně zabírat místo. Celý je umístěn v textovém souboru na přiloženém disku a v samotných souborech ve složce projekt.

Obecné funkce

Obecné funkce jsou takové, které využiji ve více typech funkcí komplexnějších. Do této kategorie by se dalo zařadit například přistupování k souborům a generování formulářů.

Práce se soubory

K souboru se přistupuje funkcí `fopen()`. Parametr přístupu se použije dle potřeby. Pokud se bude jen ze souboru číst, použije se parametr r (r+). Pokud bude potřeba před zápisem obsah souboru vymazat, použije se parametr w. Data se k souboru připojí, obsah se zachová, při použití parametru a. Po skončení práce se souborem se použije funkce `fclose()`. Využití v praxi je patrno z dvou výpisů kódu níže.

Čtení souboru po řádcích

Pro účely práce s databázemi bude vhodné, aby se s každým řádkem souboru pracovalo zvlášť. Pro jednoduchou práci bude vhodné načíst každý řádek souboru do jednoho prvku pole. S tím pak lze libovolně pracovat pomocí cyklů. K tomu se dá využít cyklu while a funkcí `feof()` a `fgets`(). Cyklus while prochází soubor tak dlouho, dokud nenarazí na jeho konec, který se zjistí pomocí funkce `feof()`. Při každém vykonání cyklu se využije funkce `fgets`(), která postupně načítá řádky souboru. Ty se uloží každý do příslušného prvku pole. V tuto chvíli je číslo řádku indexem prvku matice. Níže je kos kódu, který provádí vše výše zmíněné.

```
$matice_k = array();

$i = 0;

$soubor_k = fopen("kontakty.txt","r") or die ("Soubor se nepodarilo
otevrit");

while(!feof($soubor_k))

  {
```

```
    $radek = fgets($soubor_k, 150);
    $matice_k[$i]=$radek;
    $i++;
  }
fclose ($soubor_k);
```

Čtení celého souboru

Ve skriptech, kde se pracuje s ukládáním delšího textu, není situace tak jednoduchá. Text je potřeba opatřit značkami, které celý obsah souboru rozdělí na části, které obsahují jednotlivé logické celky. Pokud je u každého celku ještě nějaká informace navíc, pak počet různých značek roste úměrně počtu informací.

```
$soubor_t = fopen("texty.txt","r") or die ("Soubor se nepodarilo
otevrit");
$obsah = fread($soubor_t, filesize("texty.txt"));
fclose ($soubor_t);
$roz = explode("^#POPIS#^", $obsah);
list($popis,$text) = explode("^#TEXT#^", $roz[$pozice]);
```

V prvním řádku kódu se otevře soubor texty.txt a jeho deskriptor se uloží do proměnné $soubor_t. Pokud soubor texty.txt není možné otevřít, pak se vypíše hláška „Soubor se nepodařilo otevřít“. Následně se do proměnné $obsah načte obsah souboru texty.txt. Pak se tento soubor uzavře. Na dalším řádku se do jednotlivých buněk pole $roz uloží jednotlivé texty, které jsou v tomto případě odděleny značkou ^#POPIS#^. V posledním řádku se rozdělí obsah buňky pole $roz na pozici $pozice na dvě části, které odděluje značka ^#TEXT#^. Tyto dvě části jsou popisek textu a text samotný, uloženy v proměnných $popis a $text.

Dynamické generování formulářů

Na správném vygenerování formuláře stojí funkčnost celého systému. Ke generování se využije správného cyklu. U vstupních prvků lze dokonce pomocí PHP určit jeho předávanou hodnotu. U takto generovaných prvků může nastat problém s předáváním parametrů. Řešením je premovat všechny tyto prvky stejně a na konci jména zadat hranaté závorky. To způsobí to, že zadané hodnoty se budou odesílat jako pole hodnot, které se jmenuje jako vstupní prvek. Více bude jasné z příkladu.

```
<input type=”text” name=”jmeno[]” > <!-- takto se vytvoří několik
vstupních textových prvků se jménem jmeno[]-->
```

Aby bylo při zpracování zadaných dat jasné, kolikrát se cyklus provedl, je potřeba toto číslo nějak předat zpracovávacímu skriptu. Lze to dvěma způsoby. První z nich je nechat si skript zjistit počet prvků z velikosti pole pomocí funkce `count()`. Dále je možné toto číslo odeslat spolu s daty. K tomuto účelu lze využít prvek formuláře typu hidden. Do něj se uloží například počet iterací cyklu a odešle se, aniž by o tom uživatel věděl.

Funkce pro práci se skupinami a kontakty

Níže jsou již uvedeny popisy funkcí, které jsou určené ke specializovaným operacím. Jedná se o realizaci problémů, jež jsou nastíněny v kapitole 4.

Přidání skupiny

Název skupiny se zadá přes webový formulář. Ten navíc ještě obsahuje prvek checkbox, po jehož zaškrtnutí se všechno kontakty do této skupiny přiřadí a přepínač, tvořený HTML prvky radio. Přepínač přepíná mezi stavy skupiny veřejná/soukromá (implicitně je nastaven na soukromá).

Při zpracování se otevře soubor, obsahující všechny skupiny v režimu „a“ či „a+“. Do něj se nová skupina uloží a následně se soubor zavře. Otevře se soubor, obsahující kontakty a na konec pole, obsahující příslušnost ke skupinám každého kontaktu se přidá buďto nula, zůstal-li prvek checkbox v předchozím formuláři nezaškrtnutý nebo jednička v případě opačném. To zajistí to, že každý kontakt má své skupinové pole stejné délky, jako je počet skupin.

K přidání nuly/jedničky se použije funkce `substr_replace ()`. Popis této funkce je uveden v kapitole 3. Předá se jí řetězec, ve kterém se provede změna, v tomto případě pole skupin. Pozici znaku, který se bude přidávat, zjistíme buďto z počtu skupin, nebo použijeme funkci `strlen ()` a k jejímu výsledku přičteme jedničku. Vzhledem k tomu, že se tato operace bude provádět s celou databází kontaktů, tak se použije cyklu. Po skončení celé operace se výsledek uloží do databáze tak, že přepíše předešlá data (režim „w“ či „w+“ při přístupu k souboru) a ukončí se práce se souborem.

Tyto funkce obsahují soubory pridat_skup.php, kde se nachází skript zpracovávací zadaná data, prace_se_skupinami.php, který obsahuje formulář pro zadání nové skupiny. Databáze je uložena v souboru skupiny.txt. V souboru skupiny_r.php je skript, který obsahuje funkce pro načtení skupin.

Odebrání skupiny z databáze

Odebrání skupiny je analogické jejímu přidání. Načte se obsah souboru se skupinami a z těchto dat se vygeneruje formulář, který bude mít tolik vstupních prvků typu radio, kolik je skupin. Zpracující skript dostane po odeslání formuláře číselné hodnoty řádku mazané skupiny a počtu skupin. Opět se prvky databáze se skupinami načtou do pole.

Cyklem for, který se začne provádět od hodnoty mazaného prvku, se pole přeskupí tak, že se mazaný prvek přepíše následujícím prvkem. Všechny ostatní prvky se o jedno pole posunou. Následně se provede stejná operace s polem skupin u každého kontaktu. Funkcí explode () se každý řádek rozdělí na e-mailovou adresu a pole náležitosti ke skupinám. V poli se pak odebere náležitost k právě mazané skupině stejně, jako když se daná skupina mazala. (pole skupin se ozdělí na jednotlivé prvky funkcí substr() a následně se s ní pracuje stejně jako se skupinami samotnými)

E-mailová adresa a upravené pole skupin se opět sloučí do jednoho řetězce a přepíší svou původní hodnotu. Jakmile se úspěšně dokončí jak mazání skupiny, tak úprava pole skupin u kontaktu, tak se změny uloží do databází. To by mělo teoreticky zamezit inkonzistenci obou databází.

Data se zadají ve formuláři, který obsahuje soubor prace_se_skupinami.php. Skript, který vše zpracuje, se nachází v souboru smazat_skup.php. Dále se pracuje se soubory kontakty.txt a skupiny.txt, jejichž obsah se mění.

Přidání kontaktu uživatelem

Při přidání kontaktu se vygeneruje formulář, který obsahuje vstupní textové pole určené k zadání e-mailové adresy a skupinu checkboxů. Těchto checkboxů bude tolik, kolik je skupin a jejich zatrhnutím se kontakt přiřadí k dané skupině. Všechny tyto postupy jsou již popsány výše, takže nepovažuji za nutné je znovu vypisovat.

Zpracovávací skript bude testovat zadanou adresu funkcí `preg_match ()`. Pokud vyhovuje regulérnímu výrazu, pak se výstup checkboxů převede na řetězec a spojí se pomocí oddělovače s adresou. Výsledný řetězec se následně uloží do databáze kontaktů.

Formulář pro zadání dat se nachází v souboru pridat_kontakt.php. V tomto souboru se nachází i skript sloužící ke zpracování dat. Pracuje i se soubory kontakty.txt a skupiny.txt. Do prvnního zapisuje změny, druhý používá ke generaci formuláře.

Přidání kontaktů zájemcem o newsletter

Funkcí je tento skript velmi podobný přidání kontaktu uživatelem. Hlavní rozdíl je v tom, že zájemce o newsletter nebude mít přístup ke všem skupinám, ale pouze k těm, které jsou označeny jako veřejné. Samotný formulář se skládá z textového vstupního prvku a tolika checkboxů, kolik je veřejných skupin.

Po odeslání formuláře se zkontroluje, jestli je v textovém poli zadána opravdová e-mailová adresa. Jestli tomu tak je, pak se zkontroluje, jestli byla vybrána alespoň jedna skupina a jestli tomu tak je, pak se kontakt uloží do databáze. Pokud nějaký údaj není zadán, nebo je zadán špatně, pak se znovu objeví formulář žádající o správné údaje.

Další zpracování zadaných dat už je stejné, jako při přidání kontaktu uživatelem.

Funkce pro práci s textem

Wysiwyg editor textu

Sám jsem wysiwyg editor textu neprogramoval. Použil jsem již hotové řešení, ckeditor, jenž byl v době psaní práce k dispozici na adrese http://ckeditor.com/. Z textových editorů volně ke stažení je nejpropracovanější a nabízí nejvíce možností editace textu.

K zprovoznění bylo zapotřebí zkopírovat složku se soubory editoru (ckeditor/) do složky, kde je uložen web. Dále bylo potřeba do hlavičky stránky, kde je tento editor použit vložit řádky kódu, které odkazují na tento editor:

```
<script type="text/javascript" src="ckeditor/ckeditor.js"></script>

     <script src="ckeditor/sample.js"
type="text/javascript"></script>

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

V poslední řadě bylo potřeba upravit HTML prvek textarea, který slouží k zadání delšího textu. To bylo provedeno přidáním třídy do jeho parametrů, viz zvýrazněný text o řádek níže:

```
<textarea class="ckeditor" name="text" rows="10"  cols="70">
</textarea>.
```

Přidání nového textu

Pro zadání textu slouží formulář, který se skládá z textového vstupu pro zadání popisu a wysiwyg editoru pro zadání samotného textu.

Zpracovávací skript kontroluje, jestli byly obě vstupní pole vyplněná. Pokud ne, pak zobrazí znovu formulář, kde data, která již byla zadána, jsou předvyplněna pomocí funkce echo (), která do již zadaných polí vepíše data předaná z předchozího formuláře.

Pokud byla obě pole vyplněna, spustí se část skriptu, která je zapíše do databáze. Nejdříve se načtou zadaná data do proměnných $popis a $text. Ty se v dalším kroku spojí do jednoho řetězce spolu s rozlišovacími značkami, který vypadá takto: „^#POPIS#^$popis/n^#TEXT#^$text/n“. Kde ^#POPIS#^ a ^#TEXT#^ jsou zmíněné značky.

Následně se otevře soubor s texty a byly-li nějaké zadány, načtou se do proměnné $stary_text. Pak se tento soubor uzavře a otevře se znovu pro zápis. Pokud má proměnná $stary_text nějaký obsah, pak se do souboru s texty nejdříve zapíší staré texty a následně nový zadaný, v opačném případě se zapíše pouze nový text.

Formulář sloužící k zadání dat se nachází v souboru pridat_text.php. Data se zpracují v souboru pridat_text_sc.php. Databáze textů je v souboru texty.txt.

Úprava textů

Zde jsem se musel odklonit od teoretického návrhu. Původní myšlenka byla, že se hned na první stránce vygeneruje tolik formulářů, kolik je zadaných textů. Každý z nich by byl rovnou editovatelný a po odeslání by se text rovnou změnil. Po zprovoznění wysiwyg editoru to takto nefungovalo. Proto jsem na první stránce vytvořil jen náhledy, které se dají editovat až v následujícím kroku.

Pro vytvoření první stránky je nejdříve potřeba načíst všechny již zadané texty. Jak načíst obsah celého souboru naráz je již napsáno výše. Data jsou načtena jako jeden textový řetězec. Ten je nejdříve rozdělen funkcí explode () na popisky spojené oddělovací značkou s textem, do pole, které má tolik prvků, kolik je zadaných textů. Počet těchto řetězců spočte funkcí count ().

Následně se cyklem for, který se provede tolikrát, kolik je řetězců provede několik úkonů. Nejdříve se funkcí list() rozloží řetězec na popis a samotný text. Pak se pomocí funkce echo () vypíše popis. Na text je použita funkce strip_tags(), která ho zbaví HTML tagů. Následuje funkce substr (), která z textu vybere prvních 77 znaků. Následně se tento text vypíše jako náhled. Kdyby tento text nebyl zbaven HTML tagů, je možné, že by se při výpisu

vypsal jen jeden z párových HTML tagů a tím by mohly vzniknout chyby při zobrazení stránek.

V tom samém cyklu se vytvoří dva formuláře. Oba obsahují tři prvky hidden, pozice, akce a opakovani. Prvek pozice vždy předává pořadí textu v databázi. Na prvku akce záleží další zpracování formulářů. Je-li 1, pak se vybraný text edituje, je-li 2 tak se smaže. Prvek opakovani bude vysvětlen dále.

Je-li text editován a hodnota opakovani je rovna 1, pak se vytvoří již plnohodnotný formulář pro editaci textu. Díky předané pozici z minulého formuláře je možné načíst z databáze popisek a text, které se předvyplní do příslušných polí. Princip načtení textů je stejný, jako na předchozí stránce. Dále formulář obsahuje stejná tři hidden pole, jako na předchozí stránce. Pozice, akce a opakovani. Hodnota akce a pozice zůstává nezměněna, opakovani se mění na 2.

Je-li text editován a hodnota opakovani je rovna 2, provede se uložení textu. Nejdříve se načte text z databáze do pole. Pak se popisek a text zadané v minulém formuláři sloučí do jednoho řetězce, stejně jako v podkapitole 5.2.3. Následně se tento řetězec uloží do prvku pole s indexem pozice. Nakonec je celé pole uloženo do souboru texty.txt.

Když je text mazán a hodnota opakovani je rovna 1, nejdříve je položen uživateli dotaz, jestli chce daný text opravdu odstranit. Je vygenerován další formulář, do kterého se vyplní popisek a text mazaného předvyplněného textu. Po odeslání se se hodnota opakovani změní na 2 a celý text se odebere z databáze stejným způsobem, jako například kontakt.

Stránka, kde se vybírá co a jak se bude s texty dělat je uprava_textu.php. Všechny úpravy a mazání probíhají ve skriptech v souboru uprava_textu_sc.php. Databáze textů je v souboru tetxty.txt.

Odesílání elektronické pošty

Elektronickou poštu lze odeslat celkem čtyřmi způsoby. Lze ji odeslat s předvyplněným textem skupinám či jednotlivcům nebo jak skupinám, tak jednotlivcům odeslat nový e-mail.

Skupinám s předpřipravenými texty

První součástí formuláře je pole, ve kterém jsou vypsány všechny skupiny. U každé je vstupní pole typu checkbox, po jehož zaškrtnutí se při odeslání formuláře odešle i pozice

vybrané skupiny/vybraných skupin. Následuje pět roletových polí, ze kterých lze vybrat dle popisu předpřipravené texty. Stejným způsobem lze vybrat i z archivu některý z již odeslaných e-mailů.

Další pokračování ve tvorbě e-mailu je podmíněno vybráním alespoň jedné skupiny pro příjem zprávy.

Následuje již formulář pro samotné odeslání. Nejdříve je zobrazeno textové pole, kam byly vypsány e-mailové adresy všech příjemců e-mailu. Kód skriptu který tak činí je vypsán níže.

```
for ($i = 0; $i < $y - 1; $i++)     //Pro kazdy kontakt v databazi

  {   $tmp_mat = explode (" ", $matice_k[$i]);            //
rozdělení kontaktu na 2 části

   $tmp_con = $tmp_mat[0];    // přiřazení kontaktu do proměnné
tmp_con (contact)

   $tmp_gf = $tmp_mat[1];     // přiřazení pole skupin do proměnné
tmp_gf (group field)

   for ($j = 0; $j < $x  - 1; $j++)     //Rozdeleni pole skupin
kazdeho kontaktu do matice

    {     $jm[$j] = substr($tmp_gf, $j, 1);      }

   for ($k = 0; $k < $sk; $k++)    //Tolikrat, kolik je skupin se
porovna

    {

     $kl = $skupiny[$k];

     if ($jm[$kl]==1)  //Pokud je v poli skupin kontatu jednicka na
pozici skupiny (kontakt do skupiny nalezi)

      {

       $to_str = "$to_str"."$tmp_con, ";   //kontakt se prida do
seznamu prijemcu

       break;      //ukonceni cyklu - kontakt s prida jen jednou

      }

     else

      {}

    }

  }
```

Další vstupní pole je určeno předmětu e-mailu. Poslední vstupní prvek je textové pole, do kterého se vloží všechny vybrané texty.

Po zmáčknutí tlačítka odeslat se zavolá funkce mail, které se předají všechna zadaná data.

Formulář k výběru skupin a přepřipravených textů se nachází v souboru mail_form.php. Jeho druhá část se nachází v souboru mail_skupinam_sc.php a data k odeslání se nacházejí v souboru mail.php. Dále se pracuje s databází skupin v souboru skupiny.txt, databází kontaktů v souboru kontakty.txt a databází textů v souboru texty.txt.

Jednotlivcům s předpřipraveným textem

Vstupní pole je téměř totožné s polem v předchozí kapitole. Oproti němu však v tomto chybí výběr skupin. Při odesílání da z formuláře se tedy předávají pouze pozice vybraných textů.

V následujícím formuláři je pole, do kterého se dají samostatně zapsat příjemci e-mailů. Dále je možné vybrat deset příjemců z roletových menu hned pod tímto vstupním polem. Zbytek formuláře je opět totožný s předchozím.

Formulář přepřipravených textů se nachází v souboru mail_form.php. Jeho druhá část se nachází v souboru mail_jednotlivcum_sc.php a data k odeslání se nacházejí v souboru mail.php. Dále se pracuje s databází kontaktů v souboru kontakty.txt a databází textů v souboru texty.txt.

Nový e-mail skupinám

Formulář v této sekci se skládá z výběru skupin, který již byl popsán. Dále tu je textové pole, které slouží pro zápis předmětu. Do posledního textového pole se již vkládá text e-mailu.

Formulář k zadání veškerého textu se nachází v souboru novy_mail_sk.php. Dále se pracuje se souborem skupiny.txt.

Nový e-mail jednotlivcům

Rozdíl oproti formuláři pro nový e-mail skupinám je v tom, že na začátku je textové pole pro zápis příjemců a deset roletových menu pro případné vybrání příjemců z databáze.

Formulář k zadání veškerého textu se nachází v souboru novy_mail_jd.php. Dále se pracuje se souborem kontakty.txt.

Odeslání HTML e-mailu

Kód pro odeslnáí HTML e-mailu se nachází v souboru mail.php. Do něj se načtou data ze všech předešlých formulářů. U e-mailu jsou definovány dodatečné hlavičky from a reply-to tedy odesílatel a cíl odpovědi. Dále je v hlavičce definováno, že se nejedná o obyčený textový e-mail bez HTML formátování klauzulí Content-Type: multipart/alternative.

Pak je pomocí funkce ob_start () načten do paměti (bufferu) text e-mailu. V posledním kroku jsou již zmíněné funkci mail () předána všechna zadaná data a e-mail je odeslán.

Přihlášení do aplikace heslem

Heslo není nikde uloženo. V souboru nastavení je na prvním řádku uložena pouze jeho hash hodnota, ze které není v rozumném čase možné získat zpět původní hodnotu. Heslo lze změnit v souboru nastaveni.php, kde je pro změnu potřeba zadat staré heslo, nové heslo a to nové ještě jednou potvrdit. Zadaná data jsou odeslána do souboru nastaveni_sc.php, kde se porovnají hashe starých hesel. Pokud se rovnají, pak se porovnají nová hesla. Pokud jsou opět shodná, pak se spočítá jejich hash a uloží se místo hashe starého hesla.

K samotnému vstupu do aplikace je potřeba zadat heslo. Z něj se spočte hash a pokud je roven hashi uloženému v souboru nastaveni.txt, pak se nastartuje relace (session) a uloží se do ní proměnná $login = „in“.

Při vstupu na jakoukoli další stránku se testuje existence právě této proměnné. Pokud existuje a je zadána, pak se přístup povolí, jestli neexistuje, pak se zobrazí stránka žádající heslo.

6. INSTRUKCE PRO ZPROVOZNĚNÍ A UŽÍVÁNÍ

V této části je uveden manuál, kde se, jak již pravý nadpis, čtenář dozví jak zprovoznit tento systém.

Instalace

Způsob instalace závisí na způsobu užívání. Užívá-li se systém samostatně, pak stačí zkopírovat veškerý obsah složky projekt/ na webový server s podporou PHP5. Pokud má fungovat v tandemu s nějakým jiným webem pak je třeba na webový server zkopírovat soubor projekt/, nikoli jen jeho obsah.

Pokud je používán soubor samostatně, je třeba přejmenovat soubor login.php na index.php a zaměnit řádek include („login.php“); za include („index.php“); na čtvrtém řádku v souborech:

- smazat_skup.php
- pridat_skup.php
- pridat_kontak.php
- uprava_kotaktu.php
- mazani_kontaktu_skript.php
- pridat_text.php
- pridat_text_sc.php
- upravit_texty.php
- uprava_textu_sc.php
- prace_se_skupinami.php
- mail_form.php
- mail_skupinam_sc.php
- mail_jednotlivcum_sc.php
- novy_mail_sk.php
- novy_mail_jd.php
- mail.php

- nastaveni.php

Používá-li se systém společně s nějakým jiným systémem, pak je třeba v tomto druhém systému někde odkázat na soubor login.php, příklad odkazu je uveden níže.

```
<a href="projekt/login.php">Nějaký text</a>
```

První přihlášení a nastavení

Heslo pro první přihlášení je krtek. Takové zůstává, dokud se nezmění. Tak lze učinit po kliknutí na odkaz nastaveni.php v menu. Jak je vidět na obrázku 6.1 pro změnu hesla je nutné znát staré a následně zadat dvakrát nové heslo.

Dále je nutné nastavit e-mailovou adresu, ze které se e-mail odesílá a na kterou lze případně odpovídat. Tyto adresy se nastavují v téže sekci jako heslo.

Obr.: 6.1 Formuláře pro změnu hesla a adres

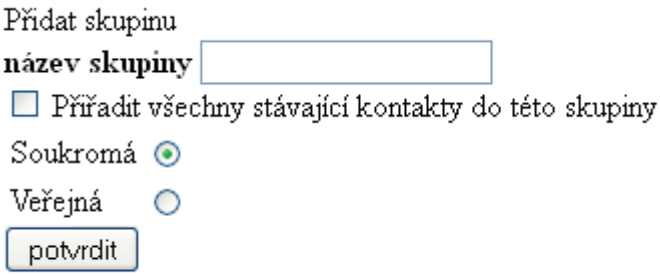
Práce se skupinami.

Skupiny se přidávají na stránce „práce se skupinami“ dostupné z menu.

Přidání nové skupiny

Název nové skupiny se vepíše do (téměř) stejnojmenného pole. Zaškrtnutím pole „Přiřadit všechny stávající kontakty do této skupiny“ se všechny kontakty v databázi stanou součástí této skupiny. Poslední částí je přepínač, jestli je skupina soukromá či veřejná. Zaškrtnutí pole ovlivňuje to, jestli se skupina zobrazí zájemci o newsletter či ne. Při zatrhnutí veřejné skupiny se skupina zájemci zobrazí v opačném případě ne.

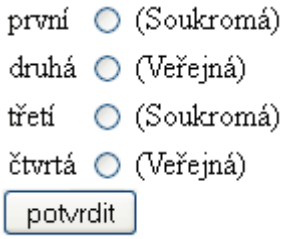
Název skupiny může obsahovat mezery, ale nesmí obsahovat řetězec „^&#&^“. Ten v databázi slouží k oddělení názvu skupiny a symbolu, který určuje, zda je skupina soukromá či veřejná.



Obr.: 6.2 Formulář pro zadání nové skupiny

Odebrání skupiny

Odebrání skupiny je možné na stejné straně. Formulář se skládá z názvu skupiny, zatrhávacího pole typu radio a výpisu, jestli je skupina soukromá či veřejná. Po odeslání požadavku na smazání skupiny se program již neptá na potvrzení.

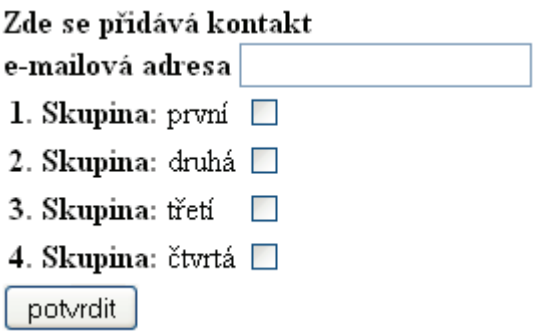


Obr.: 6.3 Odebrání skupiny

Práce s kontakty

Přidání nového kontaktu

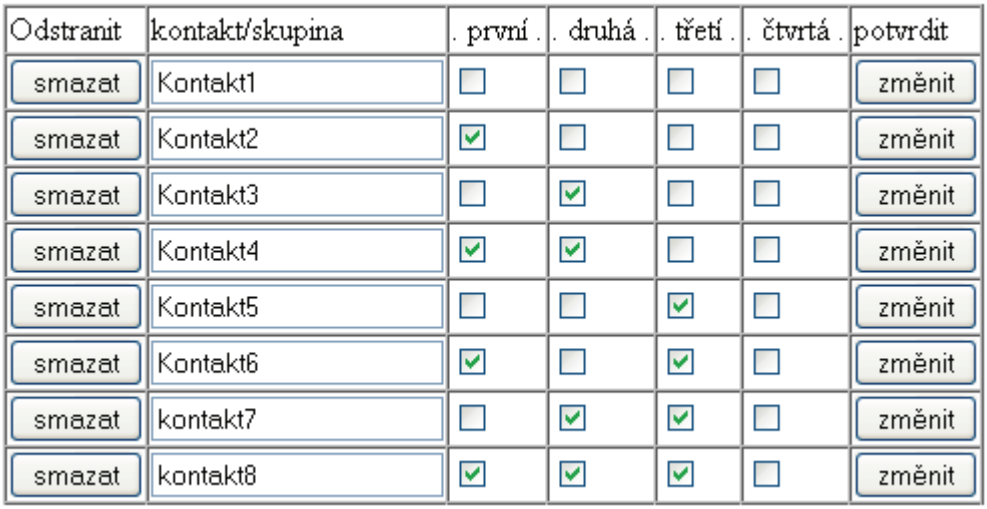
Kontakt se přidává na stránce přidat kontakt. Do pole e-mailová adresa se jak již napovídá název, uloží e-mailová adresa kontaktu. Toto pole není nijak ošetřeno proti zadání špatné adresy, protože se uživateli důvěřuje, že bude zdávat kontakty v pořádku. Dále následuje pole zatrhávacích prvků, jejichž zatržením se kontakt stane součástí příslušné skupiny.



Obr.: 6.4 Přidání nového kontaktu

Editace a odebírání kontaktů

V okně je několik vstupních prvků. Tlačítkem „smazat“ se smaže kontakt na příslušném řádku. Uživatel bude následně vyzván k potvrzení smazání kontaktu. Dalším prvkem je textové vstupní pole, v němž je vypsána e-mailová adresa kontaktu. Lze ji zde editovat. Následuje pole skupin. Zaškrtnutý vstupní prvek znamená to, že kontakt do skupiny patří. Pokud byly provedeny změny a výsledek má být uložen, pak je třeba stisknout tlačítko „změnit“. Změna je provedena okamžitě po stisknutí tlačítka.



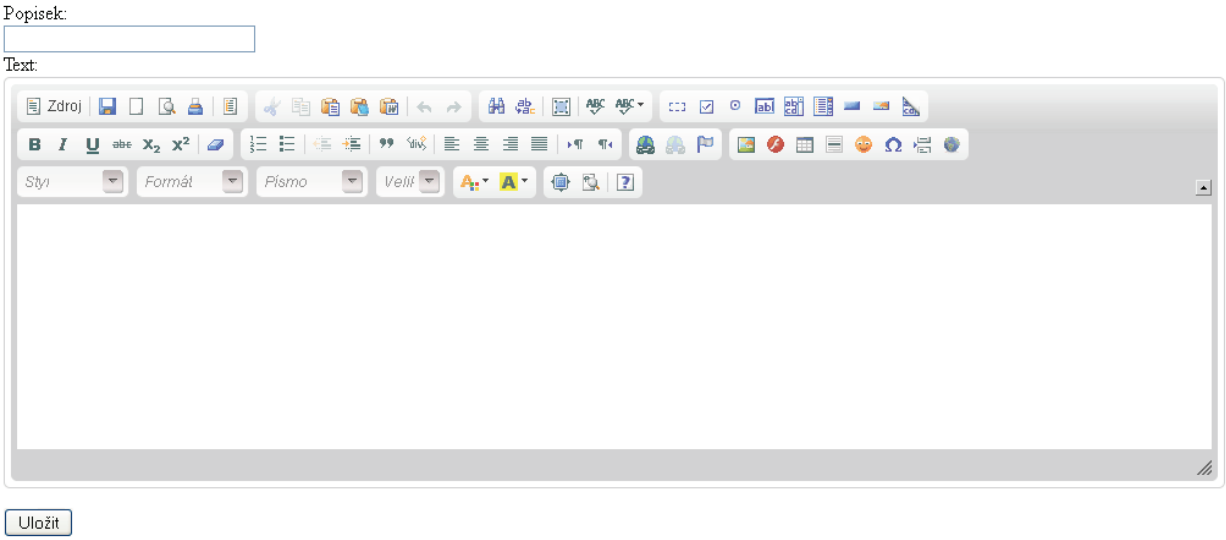
Obr.: 6.5 Stránka pro změnu kontaktů

Předpřipravené texty

Přidání nového textu

Nový text se přidává na stejnojmenné stránce. Formulář obsahuje dvě pole. Popisek a text. Pro uložení textu do databáze je nutné vyplnit obě pole. Při nevyplnění jednoho z polí se objeví chybová hláška a nový formulář, kde jsou vyplněna již zadaná data. Popisek je čistě informační a slouží ke snadnějšímu rozlišení textů.

V popisku ani textu se v žádném případě nesmí vyskytnout řetězce „^#POPIS#^“ a „^#TEXT#^“, které slouží jako oddělovací značky.



Obr.: 6.6 Zadání nového textu

Správa textů již zadaných

Po kliknutí na odkaz „úprava textů“ se zobrazí tabulka s náhledy již zadaných textů bez HTML formátování. Texty lze smazat, či editovat. Mazání textů funguje stejně jako u kontaktů.

Uživatel je po požadavku na smazání kontaktu vyzván, zda-li si přeje daný kontakt opravdu smazat a po kliknutí na potvrzovací tlačítko se tak učiní. Po stisknutí tlačítka editace se načte již formátovaný text do formuláře, který je stejný jako při zadání nového textu.

1	TAU	editovat
The Tau home world. As the eldest Sept, Tau from here are considered especial...		
smazat		
2	VIOR'LA	editovat
Vior'la orbits a binary star and translates as "hot-blooded". This is a noto...		
smazat		
3	D'YANOI	editovat
Meaning "twin moons" and isolated for many years from main body of the Tau Em...		
smazat		
4	Dal'Yth	editovat
A very cosmopolitan world, where trade is valued as much as conquest. Tau fro...		
smazat		

Obr.: 6.7 Náhledy textů

Odesílání elektronické pošty

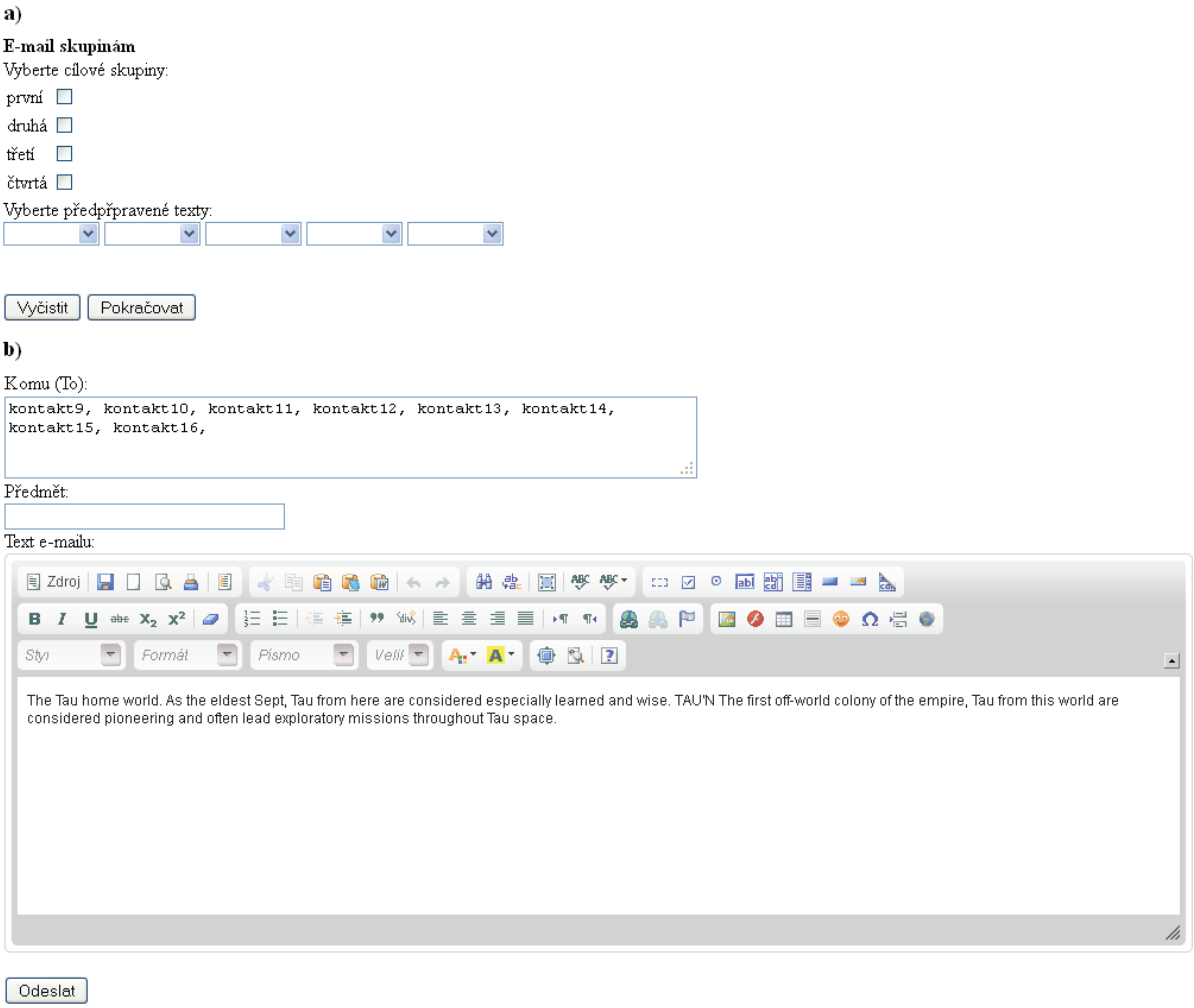
E-poštu lze odeslat z předpřipravených textů nebo novou, čistou.

E-mail skupinám s předpřipravenými texty

K psaní tohoto e-mailu se uživatel dostane po následování odkazu „Vytvoření e-mailu z textů“. Formulář obsahuje prvky pro vybrání skupin. Jejich zaškrtnutím se e-mail odešle právě jim. Zaškrtnout jich lze libovolné množství. Následuje pět roletových menu pro vybrání předpřipravených textů. Ty se po odeslání požadavku vloží do stejného editačního okna, jaké bylo k vidění například u přidávání nových textů.

Po stisknutí tlačítka „Pokračovat“ se objeví formulář pro odeslání e-mailu, do kterého lze dopsat příjemce pošty, předmět a editovat text. Po stisknutí tlačítka „Odeslat“ se e-mail odešle.

B)

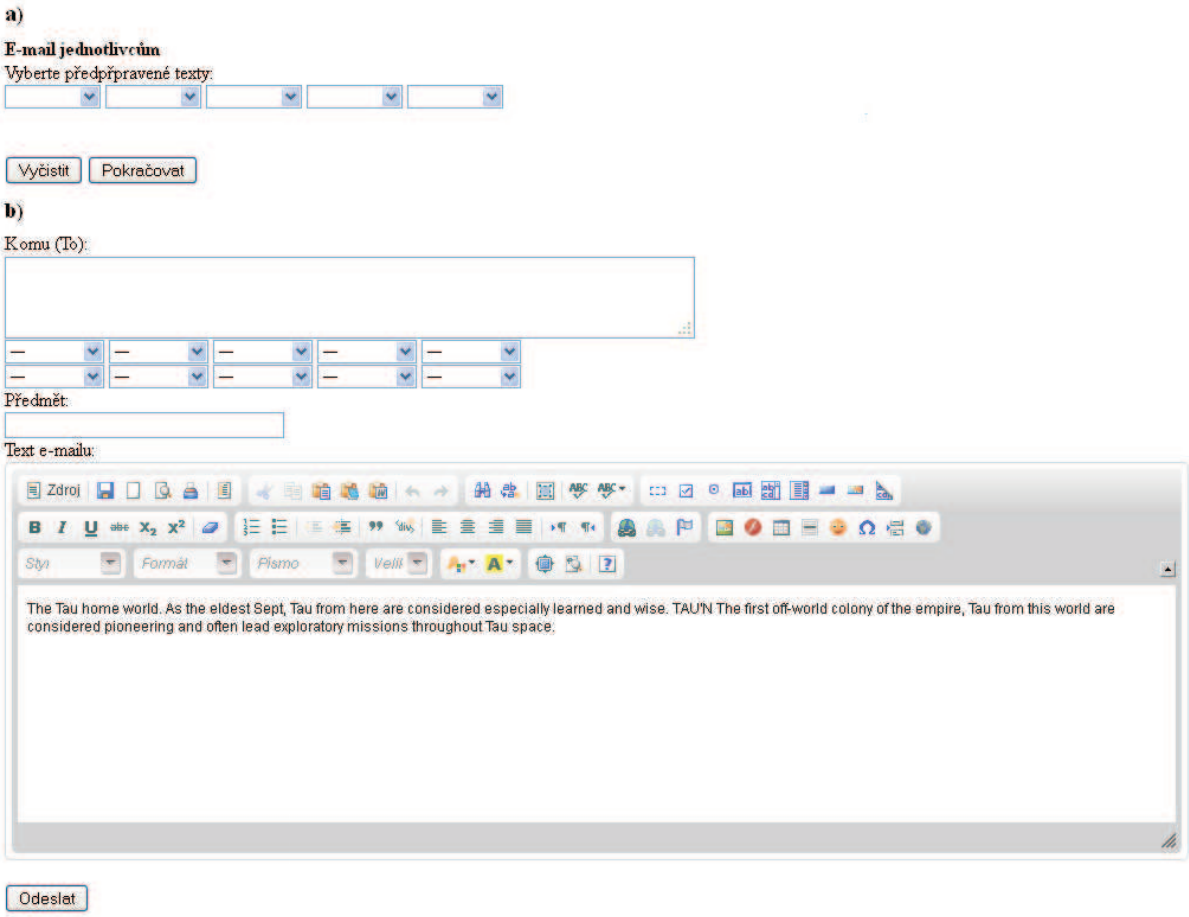


Obr.: 6.8 a) Formulář pro výběr skupin a textů. b) Formulář pro editaci e-mailu.

E-mail jednotlivcům s předpřipravenými texty

První část formuláře se nachází na stejné stránce jako u e-mailu skupinám. Je odlehčen o pole výběru skupin. Po vybrání textů se rovnou přikročí k psaní e-mailu.

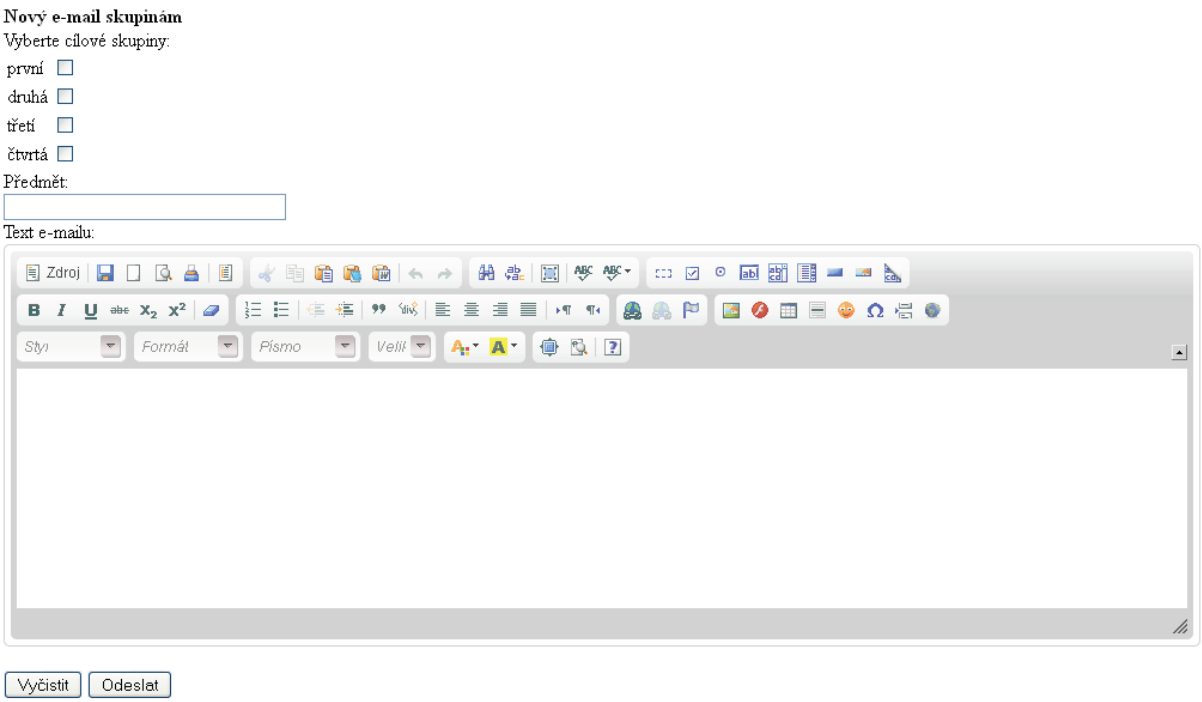
V následujícím okně lze buďto psát e-mailové adresy do okna komu ručně, oddělné čárku, či si vybrat deset příjemců z roletových menu. Systém není ošetřen proti duplicitnímu vybrání jednoho kontaktu.



Obr.: 6.9 a) Formulář pro výběr textů. b) Okno pro odeslání e-mailu.

Nový e-mail skupinám

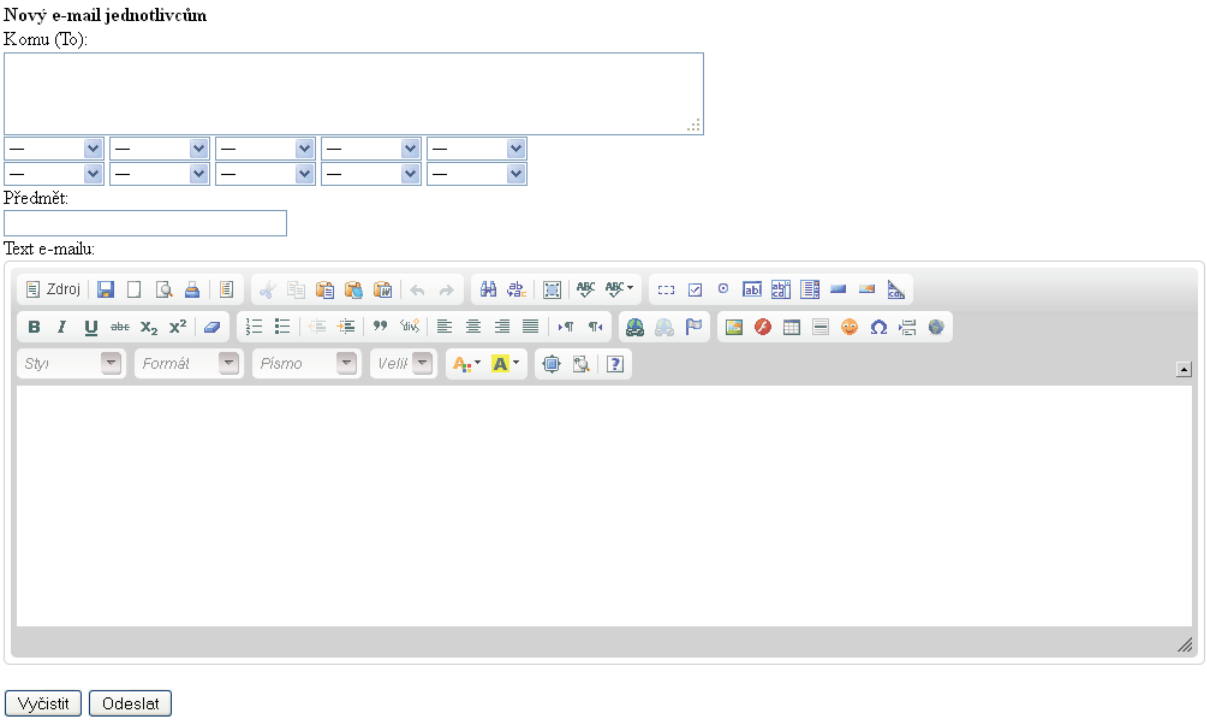
Slučuje vlastně oba formuláře z odeslání e-mailu skupinám s předpřipraveným textem. V poli skupin se vyberou cílové skupiny. Pole předmět ni samotný text není potřeba vysvětlovat.



Obr.: 6.10 Okno pro odeslání nového e-mailu skupinám

Nový e-mail jednotlivcům

Analogicky s bodem 6.6.3 tato možnost kombinuje dva formuláře do jednoho. V tomto případě z e-mailu z textu jednotlivcům.



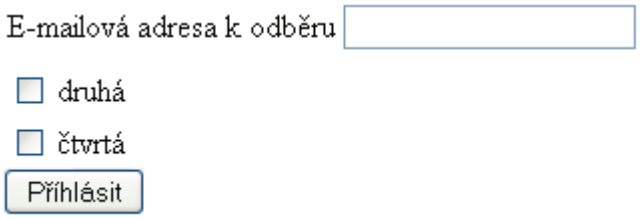
Obr.: 6.11 Formulář pro odeslání nového e-mailu jednotlivcům

„Externí“ skripty

Zde jsou uvedeny skripty, které pracují se soubory mimo složku projekt, či jsou mimo tuto složku umístěny.

Přihlášení se k odběru newsletteru

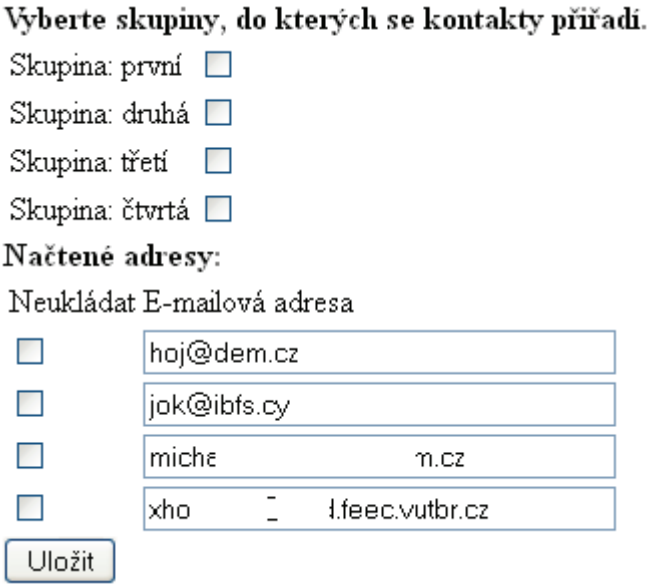
Kód tohoto skriptu se umístí do webové stránky. Do vstupního pole zájemce zadává svoji adresu a při tom si vybere, pokud je jich více, kterou skupinu newsletterů chce odebírat. Vstupní pole s adresou je ošetřeno tak, aby bylo možné do databáze přidat pouze korektní e-mailovou adresu.



Obr.: 6.12 Vstup formuláře pro přihlášení k odběru novinek

Procházení textových databází

K této funkci se uživatel dostane přes odkaz „načíst adresy“ v menu. Pokud je skript schopen načíst ze souboru adresy, pak se zobrazí v okně, které vypadá jako na obr. 6.13.



Obr.: 6.13 Formulář pro přidání načtených kontaktů

V souboru nacteni_adres.php je třeba na 21. řádku změnit v kódu:

```
$soubor = fopen("vstup.txt", "r");
```

vstup.txt za relativní adresu souboru, ve kterém se nacházejí vypsané e-mailové adresy. Jinak skript nebude korektně fungovat.

Zaškrtnutím polí u skupin se všechny přidávané kontakty do této skupiny přiřadí. Zaškrtnutím pole „Neukládat“ způsobí to, že e-mailová adresa v příslušném řádku se neuloží.

7. ZÁVĚR

Při tvoření této práce bylo využito několik programů. Zdrojový kód systému byl psán v programu Pspad. Funkce programu byla testována v balíku XAMPP verze 1.7.3 pro 32bitové operační systémy Windows.

Během praktické části, bylo využito mnoho teoretických poznatků. Ponejvíce však interakce mezi kódem napsaným v jazyce PHP a webovými formuláři. K tomu účelu bylo napsáno nejvíce skriptů.

V praktickém řešení této práce bylo naprogramováno:

- Tvorba a mazání skupin
- Přidávání a mazání kontaktů
- Práce s předpřipravenými texty
- Externí přidávání kontaktů a prohledávání textových databází
- Odesílání elektronické pošty

Tvorba i mazání skupin fungovaly správně. Bylo upuštěno do přejmenování skupin z důvodů zbytečnosti. V pozdější fázi práce se formuláře pro přidání a odebrání skupin daly na jednu stránku.

Přidávání kontaktů pracuje na stejných principech jako přidávání skupin. Proto s jeho programováním nebyl problém. Ten se vyskytl při tvoření formuláře pro editaci kontaktů. Původní myšlenka byla ukládat všechny změny naráz. Pro zjednodušení kódu byl pro každý prvek databáze kontaktů (kontakt a pole skupin) vytvořil zvláštní formulář. Funkčnost aplikace se nezměnila, jen byla přidána tlačítka a je potřeba ukládat změny pro každý kontakt zvlášť.

Jeden z největších problémů, co se vyskytl při psaní, byl wysiwyg editor textu. Moje původní idea, že zvládnu tento editor naprogramovat v PHP, byla mylná. Po konzultaci s vedoucím mi bylo doporučeno několik editorů, ze kterých jsem vybral ten nejpokročilejší. Ten se po krátké době podařilo zprovoznit.

Externí skripty byly první, ve kterých bylo využito regulérních výrazů. Hlavně prohledávání databází by bez těchto funkcí nebylo možné naprogramovat.

Funkce pro odesílání pošty v domácích podmínkách taky fungovaly. Není důvod, proč by na serveru, na kterém je PHP a dovoleno odesílat elektronickou poštu neměly být též funkční.

Do budoucna je velký prostor pro zlepšení. Bylo by potřeba ověřovat více zadaná data. Dále bych chtěl zapracovat na grafické stránce práce a nezůstat jen u základního vzhledu. Co jsem nestihl naprogramovat, byl archiv odeslaných e-mailů. To je další věc o kterou by šel tento systém rozšířit.

Závěrem chci říci, že tuto práci jsem si vybral hlavně proto, že jsem se chtěl již dlouho naučit programovat v PHP. Byť jsem se nenaučil na takovou úroveň, abych byl schopen i tento projekt naprogramovat lépe a nejsem se svým výkonem z části spokojený, jsem rád že jsem si jej vybral a naučil se něco nového.

8. CITACE

[1] JANOVSKÝ, Dušan. Jak psát web : o tvorbě internetových stránek [online]. c2001, 12. Prosince 2010 [cit. 2010-12-17]. Formuláře. Dostupné z WWW: <http://www.jakpsatweb.cz/html/formulare.html>. ISSN 1801-0458.

[2] KEBRT, Michal. Interval [online]. 03.01.2001 [cit. 2010-12-17]. Regulární výrazy v PHP 1. Dostupné z WWW: <http://interval.cz/clanky/regularni-vyrazy-v-php-1/>.

[3] KEBRT, Michal. Interval [online]. 01.02.2001 [cit. 2010-12-17]. Regulární výrazy v PHP 2. Dostupné z WWW: <http://interval.cz/clanky/regularni-vyrazy-v-php-2/>

[4] LACKO, Luboslav. PHP 5 a MySQL 5 : Hotová řešení. vydání první. Brno : Computer Press a.s., 2007. 320 s. ISBN 978-80-251-1695-1.

[5] LERDORF, Rasmus , et al. PHP Manual [online]. 1997, 3 December 2010 [cit. 2010-12-15]. Dostupné z WWW: <http://cz.php.net/manual/en/>

[6] Php. In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, 26 February 2002, last modified on 19 November 2008 [cit. 2010-12-15]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Php>

[7] Copyleft. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2 November 2001, last modified on 3 March 2010 [cit. 2010-12-17]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Copyleft>.

[8] Hašovací funkce. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 27.7.2010, last modified on 26.2.2008 [cit. 2011-05-24]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Ha%C5%A1ovac%C3%AD_funkce>.

[9] RŮŽIČKA, Pavel. *Http://interval.cz* [online]. 21.11.2002 [cit. 2011-05-24]. Začínáme používat sessions v PHP. Dostupné z WWW: <http://interval.cz/clanky/zaciname-pouzivat-sessions-v-php/>.

9. ZKRATKY

Zkratka	Popis
PHP	Hypertext Preprocessor
HTML	HyperText Markup Language
SQL	Structured Query Language
MySQL	My Structured Query Language
URL	Uniform Resource Locator
ASCII	American Standard Code for Information Interchange
DES	Data Encryption Standard
PCRE	Perl Compatible Regular Expressions
PEAR	PHP Extension and Application Repository
RFC	Request for Comments
MD	Message Digest
SHA	Secure Hash Algorithm
WYSIWYG	What You See Is What You Get