Bakalářská práce

2006

Jiří Suchan

Pedagogická fakulta Jihočeské univerzity
Katedra informatiky

Poštovní server v PHP

bakalářská práce

Jiří Suchan

České Budějovice 2006

Poděkování

Tímto děkuji PaeDr. Petru Pexovi za poskytnutí užitečných rad, informací souvisejících s tématem a objektivních postřehů během tvotby této bakalářské práce.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pouze s využitím literatury a zdrojů uvedených v části Použité zdroje.

……

Jiří Suchan

Obsah

1. Úvod

1.1. Předmluva

Internet byl dříve čistě akademickým médiem, postupně se šířícím také mezi veřejnost. Prošel bouřlivým vývojem, kdy ve svých počátcích uživatelům nabízel pouze statický obsah, a s nástupem nových technologií nyní také interaktivní obsah. V dnešní době se stále více vzhlíží k službám, které Internet nabízí, čímž zaručuje také větší flexibilitu a rychlost komunikace. To s sebou přináší výhody, především to, že tyto služby jsou dostupné odkudkoliv a lze je používat bez nutnosti vlastnit specializované nástrojů či software.Těžiště internetové komunikace je již dlouhou dobu především email, a vypadá to, že ještě dlouhou dobu jím zůstane. To však neznamená, že svými vlastnostmi vyhovuje a dostačuje pro všechny úkony. Stále vzrůstající podíl emailové komunikace tvoří spam, falešné nabídky, žádosti či hroby například v podobě phisingu. To vše činí emailovou komunikaci méně důvěryhodnou, spolehlivou a pohotovou. Existují tedy i flexibilnější řešení, než je zmiňovaný email. To s sebou však přináší nevýhody v podobě nutnosti dalších registrací, zapamatování si registračních údajů, ověřování registrací, potvrzování emailů, orientace v novém prostředí, odlišné ovládání služeb a další požadavky, podléhající aktuálním potřebám uživatele.

Jako řešení této situace, odstraňující tyto nedostatky, se jeví sloučení množství různých služeb na jedno místo. Tato práce si neklade být dokonalým emailovým klientem nabízející spoustu služeb a vlastností pro práci s emailem. V tomto období v dnešní době kralují AJAX aplikace.

1.2. Použité zkratky

1.2.1. Seznam použitých zkratek

AJAX - Asynchronous JavaScript and XML (Asynchronní JavaScript a XML)

CSS - Cascading Style Sheets (kaskádové styly)

DB - databáze

DOM - Document Object Model (objektový model dokumentu)

DTD - Document Type Definition (definice typu dokumentu)

HTML - HyperText Markup Language (hypertextový značkovací jazyk)

ODBC – Open dataBase connection standard (standard pro připojování k DB)

OOP - Object-oriented programming (objektově orientované programování)

PHP - PHP: Hypertext Preprocessor (hypertextový preprocesor)

SQL - Structured Query Language (strukturovaný dotazovací jazyk)

XHTML - eXtensible HyperText Markup Language (rozšiřitelný hypertextový značkovací jazyk)

XML - eXtensible Markup Language (rozšiřitelný značkovací jazyk)

1.2.2. Význam zkratek jinde nevysvětlených

AJAX - je obecné označení pro technologie vývoje interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich znovunačítání. Na rozdíl od klasických webových aplikací poskytují uživatelsky příjemnější prostředí, ale vyžadují použití moderních webových prohlížečů. Tyto aplikace využívají následujících technologií:

- (X)HTML a CSS pro prezentaci informací
- DOM a JavaScript pro zobrazování a dynamické změny informací
- aplikační rozhraní XMLHttpRequest pro asynchronní výměnu dat s webovým serverem (typicky je užíván formát XML, ale je možné použít libovolný jiný formát včetně HTML či prostého textu)

DOM - je objektově orientovaná reprezentace XML nebo HTML dokumentu. DOM je aplikační rozhraní umožňující přístup či modifikaci obsahu, struktury, nebo stylu dokumentu, či jeho částí.

DTD - Jazyk pro popis struktury XML dokumentu. Struktura třídy nebo typu dokumentu je v DTD popsána pomocí popisu jednotlivých značek a atributů. Popisuje jak mohou být značky navzájem uspořádány a vnořeny. Vymezuje atributy pro každou značku a typ těchto atributů.

SQL – standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích. Patří mezi deklarativní programovací jazyky, což v praxi znamená, že kód jazyka SQL nepíšeme v žádném samostatném programu (jako by tomu bylo např. v Javě), ale vkládáme jej do jiného programovacího jazyka (např. MySQL). Se samotným jazykem SQL můžeme pracovat pouze v případě, že se terminálem připojíme na SQL server a na příkazový řádek bychom zadávali přímo příkazy jazyka SQL.

XML – Jazyk vyvinutý a standardizovaný konsorciem W3C. Umožňuje snadné vytváření konkrétních značkovací jazyků pro různé účely a široké spektrum různých typů dat. Jazyk je určen především pro výměnu dat mezi aplikacemi a pro publikování dokumentů. Lze jím popsat strukturu dokumentu z hlediska věcného obsahu jednotlivých částí, nezabývá se sám o sobě vzhledem dokumentu nebo jeho částí. Prezentace dokumentu (vzhled) se potom definuje připojeným stylem. Další možností je pomocí různých stylů provést transformaci do jiného typu dokumentu, nebo do jiné struktury XML. Je nástupcem původního jazyka určeného pro publikování – HTML. Ten již přestal vyhovovat především svou složitostí, která vznikla jeho postupným rozšiřováním. Jazyk XML nemá žádné předdefinované značky (tagy, názvy jednotlivých elementů) a jeho syntaxe je podstatně přísnější, než HTML.

2. Použité technologie

2.1. Rozdělení podle umístění klient/server

2.1.1. Jazyky použité na straně klienta

Popis

Technologie jazyků na straně klienta (client-side) spočívá na stažení jeho zdrojového kódu či objektu, jako součásti požadavku na server, do počítače uživatele, kde se s ním následně pracuje. Klientské jazyky lze dále rozdělit na několik skupin.

Skriptovací jazyky

Jedná se o jazyky, kterémohou modifikovat chování okna prohlížeče, či jeho prvků. Nejpopulárnějším jazykem z této kategorie je bezesporu JavaScript. Dalším zástupcem v této kategorii je Jscript, což je v podstatě modifikovaná podoba jazyka Javascript, a pochází z dílny Microsoftu. Od tohoto výrobce pochází i jazyk VBScript, který je založený na Visual Basicu, proto také zkratka VBScript. Jazyky tohoto výrobce jsou podporovány ovšem jen v jeho prohlížeči Internet explorer, proto jejich rozšíření není tak masivní jako je tomu v případě Javascriptu.

Kvůli dosažení maximální kompatibility se používá většinou pouze JavaScript, Java a Flash.

Ostatní jazyky

Patří mezi ně applety Javy nebo objekty, ke kterým je vyžadován dodatečný plugin, např. Flash.

Výhody

Hlavním přínosem je možnost okamžité reakce na požadavek uživatele, bez nutnosti opětovného načítání dat ze serveru.

Při prvním stažení skriptu na stranu klienta jej již není znovu třeba stahovat.

Nevýhody

Obecná nevýhoda všech těchto jazyků je doba, kdy klient musí čekat na stažení všech dat, pokud jsou tato data objemnější. Přitom ne všechna stažená data jsou vždy využita.

Nevýhody skriptovacích jazyků

Využití těchto jazyků naráží na problém s kompatibilitou v různých prohlížečích. Tyto jazyky prošly dlouhým vývojem, různí vývojáři implementovaly proprietární funkce, jež nejsou vzájemně ve všech prohlížečích vždy dostupné. Tedy skripty, které fungují v jednom prohlížeči, nemusí fungovat v dalším. Při programování tak musí vzniknout několik různých skriptů pro jednotilvé prohlížeče, a to je překážkou při dalším vývoji nových verzí, přinejmenším to znamená větší časovou náročnost pro naprogramování a odladění těchto rozdílných verzí.

Protože se jedná o nekompilované jazyky, lze zobrazit jejich zdrojový kód, a tak může kdokoliv zkopírovat výsledek dlouhého a nákladného vývoje těchto skriptů.

Nevýhody ostatních jazyků

Pro spuštění java appletů či objektů vytvořených ve Flashi je zapotřebí prekvizity v podobě pluginu v prohlížeči (Flash) nebo podpory programovacího jazyka na straně klienta (Java applety). Tyto požadavky svou velikostí mnohonásobně převýší velikost samotného objektu určeného k použití. V případě potřeby instalace jazyka Java pro běh appletu, při nedostupnosti instlačního balíku v lokálním umístění klienta, je kvůli jeho velikosti téměř vyloučena další práce s ním, a uživatel musí obsah appletu oželet.

2.1.2. Programovací jazyky na straně serveru

Označovány též jako server-side jazyky. Tyto jazyky pracují tak, že klient odešle serveru svůj dotaz, ten je zpracován aplikací patřičného jazyka. Výhodou je nezávislost napsané aplikace na použitém operačním systému klienta, výsledkem je dokument XHTML. V této kategorii existuje několik programovacích jazyků z nichž každý je vhodný na určitý druh projektu.

2.2. Použité jazyky

2.2.1. PHP

Účel

Programovací jazyk, který je díky širokému množství nabízených funkcí vhodný pro zpracování široké škály úloh, ať se jedná o zpracování řetězců, XML dokumentů, výpočetní matematické operace či spolupráci s databází.

Vznik

PHP vzniklo roku 1994 jako nástroj pro osobní využití jednoho z jeho autorů. Mezi veřejnost se začalo významněji šířit až od své třetí verze, která byla vydána v roce 1998.

Popis

Jedná se o interpretovaný skriptovací jazyk, to znamená že k spouštění skriptů je nutný extérní program (tzv. interpreter), který je vykoná. Vytvořeno bylo pro spolupráci s webovým serverem Apache. Ten, stejně jako PHP, je nabízen pod open source licencí zaručující komukoliv přístup k jeho zdrojovým kódům. Díky tomuto přístupu lze jeho prostředí modifikovat podle potřeb, stejně tak i mnoho porpůrných knihoven a nástrojů. I díky tomuto bylo možno tyto nástroje portovat na většinu platforem, čímž je zajištěna široká podpora umožňující provoz tohoto jazyka na serverech.

PHP umožňuje procedurální i objektově orientovaný přístup k programování. Znalost OOP může být výhodou, není však nutnou podmínkou. PHP patří mezi jazyky, kde není nutné předem definovat typ proměnných, navíc jakákoli proměnná může kdykoli změnit svůj typ. To dává při psaní skriptů určitou volnost, může to být však na úkor přehlednosti kódu. Jazyk je šířen jako open-source, může být tedy modifikován podle potřeb, stejně tak i mnoho podpůrných knihoven a nástrojů.

V současnosti jsou se vyvíjí dvě řady verzí jazyka (4. a 5.). Jeho pátá verze přinesla lepší podporu pro OOP, a také nové vlastnosti. V minoritních verzích také probíhají změny výraznějšího charakteru, ať už v chování metod, či způsobu implemetace vlastností.

Z těchto důvodů, kdy je chování PHP v jeho různých verzích páté řady, nevyzpytatelné, a také kvůli jeho ne příliš velkému rozšíření na serverech, je preferována

verze nevyužívající nových vlastností, které nejsou zpětně kompaktibilní s předchozí, čtvrtou verzí.

Nasazení PHP 5 na serverech v ČR, květen 2006 [2]

Verze	Počet	Podíl [%]
PHP	87084	100
PHP 3	606	1
PHP 4.0	316	0
PHP 4.1	5400	6
PHP 4.2	1224	1
PHP 4.3	41794	48
PHP 4.4	22844	26
PHP 5.0	5218	6
PHP 5.1	9957	11

Výhody

Kód PHP může snadno prolínat se statickým obsahem zbytku dokumentu. Lze tak snadno vytvořit stránky obsahující jak dynamický kód, tak i statické XHTML.

Jedná se o nekompilovaný jazyk, jeho zdrojový kód lze kdykoliv upravovat bez nutnosti vlastnit speciální vývojové nástroje.

Exceluje také svou rychlostí v porovnání s konkurečními technologiemi. Je snadno dostupné, a díky jeho příznivé licenční politice je nabízeno na drtivé většině serverů pro webhosting.

PHP má výbornou schopnost se nativně připojit k mnoha rozličným databázím, i díky schopnosti využití standartu ODBC. Obecně obsahuje knihovny pro práci s mnoha rozličnými účely. Lze snadno pracovat s obrázky, PDF dokumenty, síťovými protokoly aj.

Nevýhody

Protože se jedná o nekompilovaný jazyk, přináší to s sebou nevýhodu, že kdokoliv s přístupem k serveru, kde jsou umístěny skripty, může zobrazit jejich obsah. Také rychlost, v porovnání s kompilovanými skripty rozhraní CGI je nižší.

Pojmenování metod v PHP není konzistentní. Existují metody, které mají odděleny víceslovné názvy podtržíkem, objevují se i metody, kde jsou víceslovné názvy psány v celku. Stěžuje to práci programátorovi, protože zapamatování názvů těchto metod není intuitivní, v případě použití editoru bez zvýrazňování syntaxe je nutná časté konzultování manuálu s popisem jednotlivých metod. Tato nevýhoda je umocněna jejich velkým množstvím, často je totiž dostupno mnoho metod pro vykonávání podobných akcí. Pro příklad některé metody používané k výstupu: echo, print, printf, sprintf, fprintf, vprintf, vsprintf, vfprintf, fwrite, fputs.

PHP neobsahuje jmenné prostory, které by vytvářely logické umístění těchto metod.

Syntaxe

Ukázka PHP kódu:

```
<?php
class testovaci{
     var $message;
     var $promenna;

function testovaci($promenna=5){
     $this->promenna = $promenna;
     $this->message = 'konstruktor PHP verze 4.X';
}

function __construct($promenna=5){
     $this->promenna = $promenna;
     $this->message = 'konstruktor PHP verze 5.X';
}
```

```
function verze(){
     return $this->verze;
}

function test($param)
     if($this->promenna >= $param){
          return $this->promenna;
}else{
          return $param.' je menší než '.$this->promenna;
}
}//konec tridy

$test = new testovaci(5);
echo $test->verze;
echo $test->test(8);
?>
```

Tento kód vytvoří instanci třídy 'testovaci', kde je jako argument předáno číslo 5, které bude použito jako argument u konstruktoru.

Třídy jsou označeny klíčovým slovem *class* následované názvem třídy. Za jejím názvem může následovat klíčové slovo *extends* znamenající dědičnost z jiné třídy, která následuje za tímto slovem. Vícenásobná dědičnost není podporována. Ve verzi 5 také přibyla podpora rozhraní. To se u třídy uvede klíčovým slovem *implements* s názvem rozhraní.

Konstruktory u PHP verze 4.x mají shodný název jako její třída. Tento přístup se ukázal nevyhovující, například při přejmenování třídy nebo volání rodičovského konstruktor, proto byly v páté verzi změněny na jednotný název __construct.

Metody se označují klíčovým slovem function následovaným názvem jejím názvem a seznamem parametrů v kulatých závorkách. Blok kódu funkce je ve složených závorkách. Metodu lze zavolat prostým napsáním názvu funkce s odpovídajícím počtem parametrů, či bez parametrů, pokud jsou jim v deklaraci metody přiřazeny výchozí hodnoty. Přetěžování metod není podporováno.

Proměnné se označují znakem dolaru ($). V jejich názvech mohou být použita čísla, písmena a podtržítka, číslicemi však jejich název nesmí začínat. Proměnné není nutné před jejich použitím deklarovat. Svůj typ mohou také v průběhu skriptu měnit.

Bloky kódu, který je určen ke zpracování jazykem PHP, jsou ohraničeny značkami '<?php' na jeho počátku a '?> ' na konci. Toto označení je označením uzlů <? a ?> v XML, tím pádem také v XHTML, pro uzly zpracovávající instrukce. Ve starším HTML lze použít zkrácené značky <? a ?>. Jednotlivé příkazy v kódu se vzájemně oddělují středníky.

2.2.2. XHTML

Účel

XHTML je značkovací jazyk používající definované značky (tagy) použité k vytváření a formátování dokumentů pro webové stránky využívající strukturu XML dokumentů.

Vznik

Jazyk vznikl převedením staršího jazyka HTML do podoby odpovídající podmínkám tvorby XML dokumentů se zaručením zpětné kompaktibilitou. XHTML bylo navrženo standardizační organizací W3C.

Popis

Jedná se o značkovací jazyk. To znamená, že se v něm používají speciální značky nazvané tagy, které určují vznam textu v dokumentu. Dokument se řídí definicí podle souboru DTD. Tento formát se stává nejpopoužívanějším formátem dokumentů na internetu. [1]

Je zakázáno odesílat dokument s MIME typem text/html, ale je nutno ho odesílat s MIME typem application/xhtml+xml. Po odeslání dokumentu s tímto mime typem je zpracován nikoliv HTML parserem, ale XML parserem, který netoleruje chyby ve formátu dokumentu tak jako je tomu u HTML parseru, který je k chybám tolerantnější. Požadovaného formátu lze dosahnout několika způsoby.

- pojmenování souboru příponou .xhtml
- odesláním MIME typu s hlavičkou pomocí PHP

Výhody

Neobsahuje žádné možnosti formátování, vzhled je tedy definován pomocí CSS, čímž je dosaženo oddělení obsahu od vzhledu. Je tím usnadněno zobrazení na zařízeních, která nepodporují kaskádové styly (PDA, mobilní telefony...)

Nevýhody

Jazyk je náchylný k chybám ve formátu. Není podporován všemi prohlížeči (IE).

Syntaxe

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
     <meta http-equiv="content-type"
     content="application/xhtml+xml; charset=UTF-8" />
     <title>Informační systém</title></head>
</head>
<body>
     <div id="menu">
          <a href="http://domena/zpravy.php">Zprávy</a>
     </div>
     <div id="obsah">
          <h2><b>nadpis</h2></b>   <!--chyba//-->
          <span>přidat do výběru</span>
          <input type="checkbox" value="1"
          checked="checked" />
     </div>
</body>
</html>
```

Každý element je uzavřen do špičatých závorek, bezprostředně po první závorce musí následovat název elementu. Ten může být dále následován atributy, jejichž hodnota je

uzavřena do uvozovek. Hodnotu v uvozovkách musí mít také doplňkové atributy, které se v HTML daly použít samostatně, bez přiřazené hodnoty v uvozovkách, např. atribut 'checked' u některých vstupních prvků formulářů.

Všechny tagy musí být párové. To znamená, že každý tag musí mít svůj startovní i ukončovací element. V případě nepárových tagů se přidá do jejich těla ukončovací znak, tedy například element
 bude vypadat takto:
.

Je zakázáno křížení tagů. To znamená uzavření jednoho tagu dříve, než je uzavřen tag, který je vnořený. Pro ilustraci je takováto chyba na řádku, kde je komentář upozorňující na tento nedostatek. Zde je zahájen tag h2, a je ukončen ještě dříve, než vnořený tag b.

2.2.3. CSS

Vznik

Jazyk CSS byl navržen standardizační organizací W3C. Je určen pro popis způsobu zobrazování (X)HTML a XML dokumentů. Byly vydány zatím dvě verze specifikace CSS1 a CSS2, pracuje se na verzi CSS3.

Účel

Hlavním smyslem je umožnit návrhářům oddělit vzhled dokumentu od jeho obsahu, a tím se přiblížit moderním formátům dokumentů jako je XML.

Popis

Výše uvedených možností měl dosáhnout již jazyk HTML, ale v důsledku nedostatečných standardů a konkurenčního boje výrobců prohlížečů se tak nestalo. Starší verze HTML obsahují celou řadu elementů, které nepopisují obsah a strukturu dokumentu, ale i způsob jeho zobrazení. Z hlediska zpracování dokumentů a vyhledávání informací není takový vývoj žádoucí.

Výhody

Tvorba a údržba stylu je mnohem snažší. Dokumenty mohou pro definici vzhledu použít společný soubor, jehož změna zaručí změnu všech těchto dokumentů.

Používáním kaskádových stylů lze dosáhnout větší kompaktibility i při využití některého z alternativních prohlížečů či zobrazovacího zařízení, které kaskádové styly zpracovávají odlišně či jej zpracovat neumí vůbec.

Nepřímo je také dosaženo kratší doby, po kterou je stránka načítána. Dokument obsahuje pouze data nesoucí informace, takže pokud máme styl již jednou stažený a uložený v lokální paměti, není třeba jej stahovat znovu.

Pomocí kaskádových stylů je také možno napsat styly pro hlasový syntetizátor nebo hmatovou čtečku Braillova písma.

Nevýhody

Hlavní nevýhodou CSS je zatím stále špatná podpora v majoritních prohlížečích. Různé prohlížeče interpretují stejný CSS kód jinak a je někdy velmi obtížné jej napsat tak, aby se na všech (resp. na několika vybraných) prohlížečích výsledek zobrazil stejně.

Syntaxe

```
h1{
     margin: 5px;
     font-size: 12pt;
}

p.stred{
     text-align: center;
     line-height: 10pt;
}
```

Styl definující vzhled se skládá z posloupnosti pravidel. Každé pravidlo určuje vzhled konkrétnímu elementu nebo jejich skupiny v dokumentu. Pravidlo začíná tzv. selektorem, který specifikuje skupinu elementů, pro něž bude styl použit. Celý seznam je uzavřen ve složených závorkách. jednotlivé deklarace jsou odděleny středníkem.

2.2.4. JavaScript

Účel

Obvykle používán pro rozšíření možností prvků v okně prohlížeče (tlačítka, textová políčka), tvorbu jednoduchých animací a efektů u obrázků.

Vznik

JavaScript byl původně obchodní název implementace tohoto jazyka společnosti Netscape, kde byl nejprve vyvíjen pod názvem Mocha, později LiveScript.

Ohlášen byl společně se společností Sun Microsystems roku 1995 jako doplněk k jazykům HTML a Java. Roku 1997 byl standardizován asociací ECMA (Europen Computer Manufacturers Association), o rok později také organizací ISO (International Standards Organization). Standardizovaná verze JavaScriptu je pojmenována jako ECMAScript. [1]

JavaScript od společnosti Netscape je dodnes velmi populární a nadále vznikají nové verze, ačkoliv společnost Netscape ukončila svou činnost jednak na poli internetových prohlížečů, jednak mezi programovacími jazyky.

I když byl javascript původně pouze doplňkový jazyk, v dnešní době vznikají ryze javascriptové aplikace přesahující svou velikostí 100 kB.

Popis

Javascript je v současné době nejrozšířenější skriptovací jazyk, pracující na straně klienta. Z toho plynou jistá bezpečností omezení, např. nelze pracovat se soubory na disku, aby tím nebylo ohroženo soukromí či data uživatele. Snadno by se mohlo stát, že by byly citlivé údaje odeslány, či by byla možnost naprosto znehodnotit obsah pevného disku.

JavaScript by měl být v používán pouze jako doplněk, bez kterého obsah neztrácí na funkčnosti a použitelnosti. Vhodným příkladem mohou být formuláře, kde se uživatel před jeho odesláním může dozvědět dozví, co zapomněl vyplnit. Avšak ani v takovém případě však nesmí chybět kontrola na straně serveru, která už musí být funkční za všech okolností. Každá akce tvořena javscriptem, ať už nápovědy zobrazené po najetí myši, dynamické menu nebo formátování textu pomocí wysiwig editorů či v dalších případech, by měla mít svou alternativu bez jeho využití.

I přesto se na Internetu nachází mnoho webů, které na JavaScriptu staví a to je v každém případě chyba. Například používání pouze javascriptového menu je nesmysl. Nejen že návštěvníci bez javascriptu nemohou web používat a nedostanou se na žádné jiné podstránky, ale web se stává neindexovatelný také pro vyhledávače a tudíž jako by na internetu vůbec neexistoval.

Důležité je také zvážit, pro koho jsou stránky stránky tvořeny. Pokud jsou určeny pro interní firemní síť a je tedy jasné, že budou zobrazovány všude jedním konkrétním prohlížečem s podporou javascriptu, pak není důvod, proč jej nevyužít. Na internetu však takovouto přesnou cílovou skupinu vymezit nelze, takže je třeba tvořit obsah s ohledem na všechny uživatele.

Výhody

Zaručuje mnohem menší zatěžování serveru, na kterém jsou stránky umístěny, a v prostředí klienta, které je odlišné od prostředí serveru, lze provádět i akce, které na straně serveru nelze.

Nevýhody

Velkou nevýhodou je viditelnost kódu v HTML. Zdrojový kód si může každý snadno zobrazit a tak vlastně "opsat" pracně vymyšlené skripty.

Pokud je objem javascriptového kódu větší, prohlížeč čeká na stažení celého skriptu, i když ne všechna jeho data vždy využije.

Syntaxe

```
function test(value){
     var i = 5;
     if(i > value){
               return i;
     }else{
          return false;
     }
}
```

Metody se definují pomocí klíčového slova function, které je následováno jejím názvem a seznamem argumentů v kulatých závorkách. Kód náležící metodě je ohraničen složenými závorkami.

Proměnné se deklarují pomocí klíčového slova var, následováného jejím názvem. Ten může být tvořen čísly, písmeny, avšak opět nesmí začínat číslem.

Do dokumentu se javascript vpisuje pomocí tagu <script>

2.2.5. MYSQL

Účel

Relační databáze sloužící pro uchovávání dat.

Vznik

MySQL je databázový systém, vytvořený švédskou firmou MYSQL AB. Databáze je k dispozici jak pod bezplatnou licencí GPL tak pod komerční verzí licence.

Popis

Díky snadné implementovatelnosti lze tento databázový systém provozovat téměř na všech platformách, výkonu a také proto, že se jedná o volně šiřitelný software, má v současné době vysoký podíl na poli používaných databázích. Velmi často bývá používána kombinace MySQL, PHP a Apache jako základní software webového serveru.

Databázový systém je založen na tabulkách. Řádky zastupují jednotlivé záznamy, ve sloupcích jsou pak obsaženy jednotlivé hodnoty.

MySQL ve verzi 5 přináší řadu nových vlastností, které byly již delší dobu vereřjností vyžadovány. Mezi ně patří zejména triggery, pohledy či uložené procedury. Její rozšíření je v současnosti nepatrné, proto nejsou pokročilé vlastnosti této verze použity, i z důvodu zpětné kompaktibility ke staším verzím. Pro práci s databází MySQL je použita třída, která zobecňuje práci s databází tak, že využití MySQL není podmínkou a lze použít i jinou databázi. K dispozici je v tuto chvíli však pouze ovladač právě pro MySQL.

Kvůli velmi nízkému zastoupení páté řady na serverech nejsou použity zpětně nekompaktibilní vlastnosti.

Výhody

MySQL je optimalizováno již od svého počátku optimalizováno pro vysokou rychlost. Je nabízeno pod open source licencí, díky čemuž je snadno dostupné. Téměř každý webhosting nabízí k využití právě tuto databázi.

Nevýhody

Optimalizace rychlosti byla na úkor některých zjednodušení, to znamená absenci výše uvedených pokročilých vlastností. Tyto vlastnosti byly však v poslední, páté, verzi doplněny.

Databáze není v některých ohledech dobře navržena - například u sloupců, které mají číselný formát by v dotazu vůbec neměly figurovat uvozovky či apostrofy (WHERE index <= '10'). Při špatně naprogramovaných skriptech, které databázi obsluhují, se můžou objevit potencionálně zneužitelné chyby, ku příkladu velmi častým případem je SQL injection.

Syntaxe

Syntaxe MySQL je obecně shodná se standardem ANSI SQL92. Tedy syntaxe samotného SQL není složitá, je zvolena tak, aby připomínala anglický jazyk. Pro ilustraci je zde uveden krátký příklad.

```
SELECT index,jmeno,predmet
FROM tabulka
WHERE index <= 10
ORDER BY index ASC
```

Příkaz SELECT slouží k výběru hodnot záznamů, jejichž seznam (atributů) je uveden za ním. Podmínky pro omezení výběru následují za klíčovým slovem WHERE, kde lze jako podmínky použít názvy atributů v kombinaci s matematickými operátory (<, >, =) či funkcemi SQL (NOW()...). Klauzule ORDER BY určí podle kterého atributu, a v jakém pořadí budou výsledky řazeny.

3. Jádro systému

3.1. Filosofie jádra

Pro nabídku různých služeb je potřebná flexibilita a rychlost. To znamená mít v jádře to, co všechny služby spojuje a minimum nepotřebných zbytečností, které by jej zpomalovaly. Veškerý funkční obsah je tedy je tedy přesunut do pluginů, které jsou koncipovány jako dědící třídy jádra. Možnosti pluginů budou neomezené a lze tedy těžko určit konkrétní vlastnosti, které budou všechny tyto dědící třídy obsahovat. S jistotou lze říci, že nezbytná je podpora práce s formuláři a databází.

3.2. Adresářová struktura

3.2.1. Obsah kořenového adresáře

- **index.php** - Spouštěcí soubor. Vytvoří instance základních tříd sloužících k načtení konfiguračních souborů, modulů a autorizaci uživatelů.
- **core.php** - Obsahuje třídu core, jejíž instance načte potřebné konfigurační soubory, ovladač databáze, základní třídu pro zobrazení a využitelné moduly.
- **layout.php** - Rodičovská třída pro rozšiřující moduly. Obsahuje metody s předdefinovanými bloky pro výstupní dokument a metody pro zjednodušení práce s některými tagy XHTML.
- **auth.php** - Třída zajišťující autorizaci uživatelů.
- **blank.php** - Třída vytvářená při vykonání neplatných požadavků. Po instalaci je jediným dostupným modulem.
- **config.inc.php** - Soubor obsahující konfigurační direktivy.

3.2.2. Struktura adresářů

- **classes** - Doporučený adresář pro umisťování tříd rozšiřujících modulů.
- **css** - V tomto adresáři jsou jádrem hledány soubory kaskádových stylů a další soubory sloužící pro definici vzhledu.
- **db** - Z tohoto adresáře se jádro snaží načíst třídy sloužící pro práci s databází.
- **files** - Zde jsou hledány skripty pracující na straně klienta.
- **img** - Adresář doporučený k umisťování grafických souborů vyžadovaných třídami.
- **include** - Adresář doporučený k umisťování podpůrných tříd.
- **install** - Obsahuje soubory instalačního průvodce.
- **tmp** - Cesta doporučená k nahrávání souborů pomocí formulářů, v případě potřeby.

3.3. Třídy jádra

3.3.1. Třída auth

Popis

Třída slouží pro autorizaci a přihlašování uživatelů. Proces přihlašování probíhá na základě porovnání uživatelského jméno a heslo zadaného ve formuláři se záznamem uloženým v tabulce uživatelů v databázi. Heslo je zde zašifrováno jednocestnou šifrou, takže i v případě odcizení dat z databáze zůstává znění uživatelova hesla nevyzrazeno. Pokud proběhne autorizace úspěšně, uloží se jednoznačné identifikační informace do session, jsou mu přidělena příslušná přístupová práva, čímž jsou mu zpřístupněno provádět jím přístupné akce. Tuto třídu lze používat i jako statickou v případě potřeby získání některého z uživatelských údajů ze session. Tento přístup je rychlejší, časově a prostředkově méně náročnější než přístup k stejným hodnotám uloženým v databázi.

Proměnné

- string error – řetězec nesoucí informace o nastalé chybě

Konstruktory

- void auth(string $req) – Konstruktor pro verzi PHP < 5. Je zde obsažen pro zachování zpětné kompaktibility, volá konstruktor __construct()
- void __construct(string $req) – konstruktor pro verzi PHP 5. Pokud není zvolena jiná akce, pouze ověří platnost obsahu session se záznamy v databázi.

Metody

boolean check – porovná platnost obsahu session s databází. Pokud obsah nesouhlasí, odhlásí uživatele.

boolean loggedIn – Indikuje stav přihlášení uživatele. Pokud je hodnota definující uživatelská práva větší než 0, vrací true. V opačném případě vrátí false.

boolean login(string $user, string $pwd) – na základě argumentů metodcy, definující uživatelské jméno a heslo, ověří jejich pravost s tabulkou uživatel v databázi. Pokud údaje souhlasí, vytvoří obsah session pro daného uživatele, v opačném případě pouze zapíše neplatný pokus o přihlášení do databáze, a hodnota session se nezmění.

boolean logout() – přepíše obsah session hodnotami odpovídající nepřihlášenému uživateli.

boolenan sessionCreate(int $id, string $user, string $pwd, int $prava) – uloží do session pole s hodnotami jednoznačně identifikující uživatele.

mixed getname(int $id) – pokud je hodnota argumentu větší než 0, vrátí řetězec obsahující jméno uživatele z databáze, u kterého je hodnota primárního klíče rovna hodnotě argumentu. Pokud je hodnota argumentu nulová, vrátí řetězec obsahující jméno aktuálně přihlášeného uživatele, z pole session, kde jsou uloženy jeho identifikační údaje.

Pokud není splněna ani jedna z uvedených podmínek, tedy kdy v session neexistuje prvek pole s jménem uživatele a argument neobsahuje žádnou číselnou hodnotu, je vrácena logická hodnota false.

mixed getHash() – pokud v poli hodnot session existuje prvek, který obsahuje hodnotu uživatelského hashe, je vrácen řetězec s jeho hodnotou. V opačném případě je vrácena logická hodnota false

mixed getId() – pokud je v session uložena hodnota uživatelského id, je vrácena jeho číselná hodnota. Pokud takový prvek v session neexistuje, je vrácena logická hodnota false.

mixed getPrava() – v případě, že je v uživatelské session uložena hodnota, která určuje jeho uživatelská práva, je vrácena tato číselná hodnota. Pokud neexistuje, návratová hodnota bude logického typu, s hodnotou false.

3.3.2. Třída core

popis

Třída core načítá na počátku běhu skriptu všechny potřebné soubory obsahující konfigurační direktivy, a soubory potřebné k běhu systému. Vytváří se zde také spojení s databází, díky kterému lze ověřit, zda je požadavek přicházející od uživatele právoplatný. Pokud ano, je třídou navrácena instance požadované třídy. V opačném případě je navrácena instance třídy s chybovým oznámením.

proměnné

- resource $conn - Obsahuje identifikátor připojení k databázi. Je inicializována v konstruktoru.

konstruktory

- void core( ) - konstruktor pro verzi PHP < 5, pouze volá konstruktor __construct().
- void __construct() – Načítá konfigurační soubory metodou initFiles(), vytváří spojení s databází pomocí třídy db.

metody

- void initFiles() - pomocí funkce require_once načte konfigurační soubor, ovladač databáze, třídu pro autorizaci uživatel a rodičovskou třídu layout.
- object request(string $show, string $action) - ověření požadavku uživatele na základě proměnných určujících požadovaný modul a akci a navrácení instance objektu patřičné třídy

3.3.3. Třída layout

Popis

Rodičovská třída umožňující modulům dědit metody pro práci s formuláři a dalšími prvky výstupního XHTML dokumentu, stejně jako proměnné, které v dokumentu definují některé elementy, či jejich obsah. Ty lze samozřejmě pomocí dědící třídy přepsat.

Proměnné

- string title – řetězec obsažený v tagu <title> XHTML dokumentu
- string modul - modul, který bude systémem načítán jako výchozí, přiřazena hodnota konstanty DEF
- string akce - výchozí akce načítaného modulu, přiřazena hodnota konstanty DEF2
- array skripty - pole obsahující dodatečné skripty pracující na straně klienta
- array styly - kaskádové styly určené k načtení
- array rootmenu - položky menu 1. úrovně
- array submenu - položky 2. úrovně menu
- array error - pole obsahující řetězce popisující chyby vzniklé v průběhu skriptu
- string content - obsah těla XHTML dokumentu

Metody

- string error() - vypíše chyby z instanční proměnné error
- void prepare() - sjednocuje volání metod rootmenu() a submenu()
- void rootMenu() - načte položky menu 1. úrovně, ke kterým má uživatel práva přístupu
- void subMenu() - načte položky patřící do menu 2. úrovně, ke kterým má uživatel práva přístupu
- string head() - vrátí řetězec obsahující počátek XHTML dokumentu
- string body() – vrátí řetězec obsahující záhlaví XHTML dokumentu
- string footer() - vrátí řetězec ukončující tělo XHTML dokumentu
- string drawRootMenu() - vrátí řetězec obsahující XHTML kód menu 1.úrovně
- string drawSubMenu() - vrátí řetězec obsahující XHTML kód menu 2.úrovně
- void headTo(string $where,string $what,string $misc) – přesměruje XHTML dokumentu pomocí PHP funkce Header s direktivou "Location"
- string unHtmlEntities(string $retezec) - Přepíše XHTML entity na jejich zkrácený tvar (např. & na &)
- string cesta(string $show,string $action,string $misc) – vytvoří řetězec pro určitou akci podle argumentů metody
- string odkaz(string $show,string $action,string $misc,string $title,string $hrefTitle,int $level,int $minPrava) - vrátí řetězec tagu <a> podle zadaných parametrů
- string userBox() – vrátí řetězec obsahující XHTML kód přihlašovacího formuláře
- string formular(array $pole) – šablona formuláře, vrací řetězec jeho XHTML kódu
- void draw() - funkcí echo odešle na výstup obsah XHTML dokumentu

3.3.4. Třída db

Popis

Třída je zahrnuta ve snaze zjednodušit práci s databází. Účelem této třídy není konkurovat mocným rozšířením jako je například PDO, či jemu podobné. Svou velikostí, která nedosahuje ani 8 kB, se snaží nenarušovat koncept malého, rychlého jádra. Hlavním přínosem je zjednodušení často prováděných úkonů a navázat se na konkrétní typ databáze.

Pro práci s databází je nejprve nutno vytvořit spojení, to je vytvořeno ve třídě core, jejíž instance je vytvořena při volání skriptu téměř v jeho počátku. Po vytvoření tohoto spojení s databází lze tuto třídu používat i jako statickou. Volání metod v takovém případě probíhá tak, že se napíše název třídy, poté operátor '::' a název metody, tedy například db::insert(). Kvůli tomuto způsobu využití je třída pojmenována velmi krátkým názvem. U tohoto volání lze vidět oblastní operátor (::), který v tomto případě slouží k vypsání statické vlastnosti, metody či konstanty mimo třídu

Ovladače pro databázi jsou uloženy v adresáři db, zvolen je ovladač, který má stejné jméno souboru jako hodnota konfigurační konstanty DB_TYPE ze souboru config.inc.php.

Proměnné

- string $host – DB server
- string $user – přihlašovací jméno k DB
- string $pwd – přihlašovací heslo k DB
- string $base – jméno databáze
- string $charset – znaková sada pro komunikaci s DB

Konstruktory

- void db() – konstruktor pro verzi PHP < 4
- void __construct() – konstruktor pro verzi PHP 5

Metody

- mixed conn() - vytvoří identifikátor připojení k databázi
- boolean test() – vrací vždy true (např. pro zjištění dostupnosti třídy)
- string type() – vrací řetězec určující typ ovladače databáze
- boolean createDB(string $host,string $user,string $pwd,string $name, string $charset) - pokusí se vytvořit databázi, pokud již neexistuje
- boolean createTable(string $name,string $in,string $suff,string $action) – pokusí se vytvořit tabulku v databázi
- boolean update(string $table, array $key, array $values) - provede SQL příkaz UPDATE
- string slashes(string $val, string $act) – přidá řetězci uvozovky
- mixed insert(string $table, array $values) – provede SQL příkaz UPDATE
- mixed insUpd(string $table,array $key,array $values) - vloží nebo upraví záznam v databázi
- mixed del(string $table, array $params) – provede SQL příkaz DELETE
- mixed query(string $query, int $debug, boolean $direct) – použije řetězec jako SQL příkaz databázi
- int rowsCount(resource $resource) – vrátí počet řádek daného zdroje
- mixed error() – vrátí informaci o potencinální chybě
- string makePrefix(string $query) – dosadí prefix k názvům tabulek v četězci
- mixed select(string $select, string $table, array $params, string $type) – vrátí výsledek SQL příkazu SELECT
- string untype(string index,string val,int prvky) – vrátí část zformátovaného řetězce pro SQL dotaz

3.3.5. Třída blank

Popis

Třída blank je určena pro zobrazování chybových hlášení při neprávoplatném požadavku či při chybném pokusu o přihlášení uživatele. Její instance je vytvořena ve třídě core v případě neúspěšného požadavku, či chybném pokusu přihlášení.

Proměnné

- title – překryje obsah proměnné v rodičovské třídě layout.

Konstruktory

- blank(string $req) - konstruktor pro verze PHP < 5. Parametr určuje požadavek na zobrazovaný obsah. Tento konstruktor pouze volá konstruktor pro PHP 5, kerému také předává svůj argument.
- __construct(string $req) – konstruktor pro verzi PHP 5. Parametr určuje zobrazovaný požadavek.

Metody

- badlogin() - zobrazí infromace o nepodařeném přihlášení uživatele
- show() – zobrazí informace o neplatném požadavku uživatele

3.4. Práce s třídou db

pro demonstrační účely použijeme tabulku s třemi sloupci, z nichž první je primární klíč celočíselného typu s vlastností auto_increment (to znamená, že u nově vkládaných záznamů bude jeho hodnota automaticky zvětšena o jedničku) , ostatní jsou typu varchar (mohou obsahovat libovolný řetezec do délky 255 znaků):

- id – int, primární klíč (celočíselná hodnota, nesmí se v tabulce opakovat)
- jmeno – varchar (řetězec libobolné délky, max. 255 znaků)
- prijmeni – varchar (řetězec libobolné délky, max. 255 znaků)

3.4.1. vkládání/upravování záznamu

Uživatel odešle data ze vstupního formuláře, která mají být vložena do zmíněné tabulky v databázi. Běžný postup, kdy se musí rozhodnout, zda záznam, u kterého existuje klíč s danou hodnotou, existuje a bude upraven, či zda se vloží záznam nový, by vypadal takto:

```
$id = intval($id);
$jmeno = addSlashes($jmeno);
$prijmeni = addSlashes($prijmeni);

$res = mysql_query("SELECT Count(id) FROM ".PREFIX."tabulka
WHERE id = '$id' GROUP BY id");

$jeZaznam = mysql_fetch_row($res);

if(!$jeZaznam[0]){
     mysql_query("INSERT INTO ".PREFIX."tabulka VALUES(NULL,
'$jmeno', '$prijmeni')");
}else{
     mysql_query("UPDATE ".PREFIX. "tabulka SET jmeno =
'$jmeno', prijmeni = '$prijmeni' WHERE id = '$id'");
}
```

Na prvním řádku ošetříme vstup proměnné id, tak aby byly eliminovány potencionálně nebezpečné znaky, které by mohly narušit provedení dotazu. Funkce intval() vrátí celočíselnou část z proměnné v jejím parametru. Na dalších dvou řádcích ošetříme proměnné, jejichž obsah je řetězec. V tomto případě se provede úprava pomocí funkce addSlashes() přidávající před potencionálně nebezpečné znaky zpětná lomítka. Poté je třeba otestovat, zda v tabulce již existuje záznam mající u atributu id hodnotu jako stejnojmenná proměnná. Načteme výsledek SQL dotazu do proměnné obsahující výsledný řádek. Pokud je výsledek prázdný, znamená to, že tento záznam neexistoval a můžeme jej vložit. V opačném případě daný záznam upravíme.

S pomocí třídy db lze tuto operaci provést o poznání snadněji:

```
db::insUpd(    ,tabulka’,
               Array(    ,id%i’ => $id),
               Array(    ,jmeno%s’ => $jmeno,
                         ,prijmeni%s’ => $prijmeni));
```

Metoda insUpd má tři parametry:

První je jméno tabulky, se kterou se bude pracovat. V metodě bude k jejímu názvu automaticky přidán prefixový řetězec, pokud je použit.

Druhý parametr je pole obsahující atributy z tabulky, podle kterých se bude porovnávat, zda se bude vkládat nový záznam, v případě, že atribut s danou hodnotou nebude nalezen, nebo se bude upravovat existující záznam s nalezenou hodnotou.

Třetí atribut je pak opět pole s atributy, které se budou vkládány či upravovány.

Každý index pole může být doplněn o definici typu proměnné, která bude patřičně před uložením do databáze uložena. Například řetězec (%s) bude opatřen lomítky před citlivými znaky pomocí funkce addslashes(), celočíselné číslo bude vloženo po ošetření funkcí intval().

Používat lze také MySQL funkce jako je NOW() pomocí suffixu %f, či nechávat řetězce neměnné, v případě potřeby vkládání jiných údajů, které nevyhovují žádnému z výše uvedených (IS NOT NULL atp.).

3.4.2. Výběr dat z databáze

Klasický případ, kdy uživatel potřebuje vybrat data z tabulky. Použijeme opět výše uvedenou tabulku s třemi sloupci id, jmeno, prijmeni. Tentokrát budeme chtít vypsat všechny záznamy, kde je jméno totožné se vstupní proměnnou jméno.

```
$jmeno = addSlashes($id);

$res = mysql_query("SELECT jmeno,prijmeni FROM ".PREFIX.
"tabulka WHERE jmeno = ,$jmeno’");


if(mysql_num_rows($res)==0){
```

```
     echo ,žádný vyhovující záznam nebyl nalezen’;
}else{
     while($data = mysql_fetch_array($res)){
          echo $data[jmeno].’, ‘.$data[prijmeni];
     }
}
```

Opět ošetříme proměnnou jmeno tak, aby nemohla obsahovat potencionálně nebezpečné znaky. Následovně se pokusíme načíst všechny záznamy s příslušnou hodnotou v daném sloupci. Poté porovnáme počet vrácených řádek. Pokud je nulový, vypíšeme oznámení. V opačném případě použijeme cyklus while(), který bude postupně vypisovat data z řádek obsažených ve výsledku dotazu.

Řešení pomocí třídy db bude vypadat takto:

```
$rs=db::select(,jmeni, prijmeni’,
               ,tabulka’,
               Array(,jmeno%s’ => $jmeno),
               ‘arrays’);
if(!$rs){
     echo ,žádný vyhovující záznam nebyl nalezen’
}else{
     foreach($rs as $data){
               echo $data[jmeno].’, ‘.$data[prijmeni];
     }
}
```

Metoda select má čtyři parametry. První je řetězec který bude dosazen ke klauzuli SELECT v SQL příkazu. Dalším argumentem je jméno tabulky, ke kterému se přidá případný prefix. Třetí argument je pole, které definuje obsah klauzule WHERE v SQL příkazu. Index každého prvku pole je použit jako jméno atributu. Lze opět použít znak procenta s označením typu proměnné. Obsah prvku pole je hodnota, kterou bude mít argument přiřazenu. Posledním, nepovinným argumentem je řetězec, který označuje typ návratové hodnoty. Pokud nebude tento argument použit, bude navrácena proměnná typu resource, to jest zdroj výsledku SQL dotazu. Tento argument může nabýt hodnoty ‚row’,

‚array', ‚object' pro navrácení konkrétního typu výsledku SQL dotazu, nebo ty samé v množném čísle pro navrácení pole příslušných typů. K těmto možnostem se ještě přidává zmíněný ‚result' a ‚rowsCount' vracející počet záznamů ve výsledku.

3.5. Struktura databáze

V databázi jsou implicitně vytvářeny tři tabulky. Jejich názvy jsou opatřeny prefixy, v případě potřeby instalace několika paralelní instancí systému.

První z tabulek uchovává základní informace o uživatelských účtech.

3.5.1. Tabulka uživatelů:

- id – primární klíč tabulky, celočíselný, neznaménkový atribut s vlastností auto_increment (tzv. nově vložené záznamy automaticky zvyšují hodnotu tohoto atributu)
- nick – unikátní řetězec s volitelnou délkou. Slouží jako přihlašovací jméno.
- pwd – řetězec obsahující zašifrované heslo jednosměrnou šifrou.
- email – řetězec s proměnlivou délkou, obsahuje email uživatele, používaný např. při ztrátě hesla či ověření pravosti účtu
- prava – neznaménková celočíselná hodnota definující uživatelská práva, nabývá hodnotu od nuly (práva nepřihlášeného uživatele), práva s hodnotou jedna má běžný uživatel, až do devíti (práva administrátora).
- aktivni – celočíselná hodnota určující, zda je uživatelský účet aktivní a je možno se přihlásit.

Další dvě tabulky obsahují seznam tříd, akcí a omezení týkající se přístupových práv, se kterými lze akce vykonávat. První tabulka obsahuje pouze seznam těchto tříd s jejich rozšiřujícími vlastnosmi.

3.5.2. Základní tabulka tříd

- id – primáční klíč tabulky, celočíselný, neznaménkový atribut s vlastností auto_increment (nově vložené záznamy automaticky zvyšují hodnotu v tomto sloupci)
- pozice – celočíselná hodnota určující umístění položek v menu nejvyšší úrovně. Položky jsou řazeny vzestupně.
- query – unikátní řetězec s proměnlivou délkou, označuje název třídy, je stejnojmenný jako soubor, ve kterém je třída označena. Obsah tohoto sloupce je porovnáván s proměnnou $show ze superglobálního pole $_GET, určuje tedy polovinu požadavku uživatele.
- req – pokud neobsahuje hodnotu NULL, pak je při volání této třídy vkládán stejnojmmenný soubor z adresáře classes s příponou php voláním metody require_once. Pokud je obsažená hodnota NULL, soubor se nevkládá, tedy v případě, kdy je požadavek veden na třídu načítanou implicitně (např. auth).
- version – obsahuje celočíselné označení verze třídy, většinou ve formátu YYMMDD s uvozujícími nulami.

Třetí, implicitně vytvářená tabulka, obsahuje informace o dostupných akcích jednotlivých tříd. Tato tabulka obsahuje vazbu na výše uvedenou tabulku, a to přes jejich sloupce id.

3.5.3. Tabulka akcí tříd

- id – spolu se sloupcem ‚action' tvoří primární klíč tabulky, tento sloupec je provázán na tabulku tříd, určuje propojení akce k třídě se stejným id.
- action – obsahuje řetězec označující akci vykonávanou v dané třídě, určenou proměnnou $show, spolu s proměnnou $action ze superglobálního pole $_GET, která je porovnávána s tímto sloupcem.
- min_prava – obsahuje celočíselnou hodnotu, která určuje minimální uživatelská práva potřebná k provedení této akce. Pokud je tedy veden požadavek uživatele veden na příslušný záznam (porovnává se ve třídě core), a uživatel má menší práva, než je obsaženo v tomto sloupci, nemá opravánění tuto akci vykonat, a je mu navrácena instance třídy blank.

- max_prava – obsahuje celočíselnou hodnotu, která určuje maximální práva, se kterými lze danou akci vykonat. Toto omezení se uplatní u akcí, kdy je její vykonání neopodstatěné, například akce pro přihlášení uživatele v případě, kdy už přihlášen je.
- aktivni – obsahuje celočíselnou hodnotu vyjadřující možnost provedení akce
- title – řetězec obsahující popis akce, může být umístěn v atributu title u výsledného elementu v menu.
- href_title – obsahuje řetězec, který tvoří obsah tagu <a> v menu. Může obsahovat i hodnotu NULL pro akce, pro které nemá položka v menu význam (odeslání formuláře)

3.6. Průběh volání skriptu

Volání začíná v souboru ve výchozím souboru index.php. Odtud jsoupostupně volány metody dílčích tříd jádra, které inicializují podstatné části systému, až k volání metody uživatelem požadované třídy a vykreslení.

session_start()

Funkce PHP, která inicializuje data pro sezení. Podpora session je nezbytná, protože je zde používána pro autorizaci uživatelů. Tato metoda musí být vždy volána před odesláním dat na výstup prohlížeči.

require_once ‘core.php’

načte obsah souboru core.php, ve kterém je obsažena stejnojmenná třída.

$core = new core()

Vytvoření nové instance třídy core. V konstruktoru je volána metoda obstarávající načtení obsahu dodatečných konfiguračních souborů, opět pomocí funkce *require_once.* Jedná se o tyto soubory:

- **config.inc.php**: obsahuje konfugurační konstanty pro přístup k databázi, definování výchozího modulu atp.
- **/db/ DB_TYPE .php**: třída pro práci s databází, zvolena na základě konstanty *DB_TYPE* ze souboru *config.inc.php*

- **auth.php**: třída zajišťující autorizaci uživatel a přidělení jejich přístupových práv
- **layout.php**: rodičovská třída *layout*, používána rozšiřujícími moduly

Následně se vytvoří spojení s databází, identifikátor se uloží do proměnné conn:

- **$this->conn = db::conn()**

$auth = new auth()

vytvoří instanci třídy zajišťující autorizaci uživatele. Obsahu session se porovná se záznamy uživatel v databázi. Pokud je obsah session prázdný, budou uživateli přidělena práva odpovádající neregistrovanému uživateli, stejně tak i v případě, kdy hodnoty v session nebudou korespondovat s žádným záznamem v databázi.

$page = $core->request($_GET[‘show’],$_GET[‘action’])

Metodou třídy core se pokusíme zjistit zda je uživatelův požadavek smysluplný, na základě proměnných $show a $action ze superglobálního pole $_GET, a také zda má na jeho vykonání dostatečná práva

- pokud ano, metoda vrátí instanci objektu jím požadovaného modulu
- pokud má nedostačující opravnění, či je dán požadavek na neexistující akci, je navrácena instance třídy *blank* zobrazující informaci o jeho nedostupnosti.

$page->draw()

Metoda odešle, pomocí funkce *echo*, na výstup obsah instanční proměnné content, která obsahuje tělo vygenerovaného XHTML dokumentu, opatřeného také hlavičkou a patičkou, na základě daného požadavku uživatele.

3.7. Instalace

Ve snaze o maximální přizpůsobitelnost systému týkající se nabídky vlastností a komponent, probíhá zvlášť instalace jádra a volitelně každé rozšiřující třídy. S jádrem se nainstaluje automaticky pouze třída umožňující registraci dalších uživatel a třída pro nastavování vlastností systému.

Základní systém je potřeba nakopírovat na FTP server. Poté je k dipozici průvodce s jehož pomocí se nastaví parametry systému určující lokaci umístění na serveru, přístup k databázi a nastavení administrátorského účtu.

Poté již lze systém provozovat, avšak zatím nejsou dostupné žádné rozšiřující třídy. Ty lze nainstalovat po přihlášení administrátora kdy je k dispozici příslušný formulář.

4. Rozšiřující třídy

4.1. Instalační soubory

Rozšiřující třídy lze instalovat ručně či z archivů ve formátu zip. Soubor je zkopírován a rozbalen pomocnou třídou *unzip*. Soubor musí nutně obsahovat 4 soubory obsahující konfigurační a instalační soubory, které musí být na jeho začátku. Toho lze dosáhnout umístěním do adresáře začínajího např. podtržítkem či jinými znaky, které jsou v ASCII tabulce před všemi znaky abecedy, tedy např. do adresáře '_DATA'.

Konfigurační soubory jsou následující:

- INSTALL – tento soubor obsahuje instrukce vykonávané při instalaci, tedy například kód pro vytvoření databázových tabulek či jejich obsahu.
- NAME - obsahuje řetězec, kterým bude označen v systému instalovaný plugin, např. *diskuse*
- REQUIRE - obsahuje závilosti na ostatních nainstalovaných třídách. Může určovat vyžadované či konfliktní třídy.
- VERSION - číselné označení určující verzi instalované třídy. Pro jednotné označení je použito číslování podle datumu ve formátu YYMMDD

Ostatní soubory budou zkopírovány i s jejich adresářovou strukturou. To znamená, že samotná třída bude hledána v adresáři *./classes*, v souboru, který bude mít jméno totožné s řetězcem obsaženým v souboru *NAME*, s příponou *.php*. V adresáři *./css* bude hledán stejnojmenný soubor, s příponou *.css*, v adresáři *./files* pak soubor s příponou *.js*.

4.2. Struktura třídy pluginu

Instance se vytváří v metodě request třídy core na základě jejích parametrů, které označují požadovanou třídu a akci v ní prováděnou. Požadavek se ověří s databází, zda je požadavek dostupný, a zda má uživatel na jeho vykonání oprávnění, podle hodnoty session. Pokud vše proběhne úspěšně, vytvoří se instance třídy s parametrem požadované akce. V opačném případě bude vytvořen objekt třídy blank zobrazující chybové oznámení

o nedostupnosti akce. Konstruktor rozhodující o vykonání akce může vypadat například takto:

```
<?
class diskuse extends layout{

function diskuse($req=DEF2){  /* konstruktor pro PHP < 5 */
     $this->__construct($req);
}

function __construct($req=DEF2){/* konstruktor pro PHP 5  */

     switch($req){  /* určení požadavku */
          case ‘room’:
               $this->content = $this->room($_GET[‘misc’]);
          break;
          case DEF2:
          default:
               $this->content = $this->show($_GET[‘misc’]);
          break;
     }
}
/* obsah třídy */
}
?>
```

Obsahuje konstruktor pro verzi PHP 4, z důvodu zachování kompakbtibility. podle argumentu konstruktoru se následně, např. ve switch, nebo pomocí podmínek if – elseif – else, rozhodne o tom, která akce se vykoná. Návratová hodnota této metody, která by měla tvořit řetězec obsahující tělo dokumentu, se uloží do instanční proměnné content, která bude na konci skriptu vypsána spolu s ostatními částmi dokumentu.

5. Rozhraní systému

5.1. Uživatelský přístup

Systém má společné administrátorské i uživatelské rozhraní, z důvodu toho, že administrátor je také uživatel, avšak oproti ostatním má navíc přístup i k administračním funkcím, které jsou dostupné právě s vyššími uživatelskými právy administrátora.

5.2. Grafická struktura uživatelského rozhraní

Obsah dokumentu je vytvořen pomocí XHTML. K jeho stylování jsou použity kadskádové styly neobsahující téměř žádnou grafiku, pro dosažení optimální rychlosti při načítání obsahu. Použitím CSS je zaručena snadná přizpůsobitelnost rozhraní.

5.3. Navigace

Menu je rozděleno do dvou úrovní. V nejvyšší úrovni jsou uchovány odkazy pro jednotlivé dostupné třídy spolu s dalšími odkazy, které je vhodné mít kdykoliv přístupné, např. menu pro nastavení či odhlášení uživatele. Obsah tohoto menu se mění podle uživatelských práv.

Menu druhé úrovně je tvořeno odkazy pro akce dostupné v aktuálně zobrazované třídě. Některé třídy umožňují zobrazovat položky v tomto menu residentně, bez ohledu na zobrazovanou stránku. Například při doručení nové zprávy je vhodné mít zobrazeno upozořnění na tuto skutečnost kdekoliv, aby uživatel tyto kklíčové informace nemusel zobrazovat nabídce oné třídy, což by jej mohlo vyrušovat od práce.

6. Závěr

Podařilo se vytvořit systém, který může být úspěšně využít k elektronické komunikaci. Umožňuje využívat email, a navíc disponuje také dalšími možnostmi, pro které jsou flexibilnější, rychlejší, ale mají jasně vymezené hranice prostředím serveru, kde je systém provozován.

Systém neklade důraz na vysoké množství vlastností a interaktivity ovládacího prostředí. K těmto účelům lze vhodně použít jiné technologie, kterým serverový jazyk PHP svou podstatou nemůže konkurovat (např. AJAX). Hlavní přínos pro uživatele je nabídka uceleného komunikačního řešení s možností snadné rozšiřitelnosti pomoci přídavných modulů.

Přínosem pro programátora jsou předpřipravené prvky, které jsou velmi často používané, systém se snaží práci s nimi zjednodušit a zkrátit tak čas potřebný k vývoji dalších prvků.

7. Použitá literatura

Internetové zdroje:

[1] www.wikipedia.org

[2] www.root.cz

www.php.cz

www.interval.cz

www.thedailywtf.com

php.vrana.cz

www.jakpsatweb.cz

www.kosek.cz

www.linuxsoft.cz

www.dgx.cz

www.w3c.org

Použitá literatura:

Ullman, L. PHP a MySQL, Computer Press, 2004

Zeldman, J. Tvorba webů podle standardů, Computer Press, 2005