VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

MODULOVÝ REDAKČNÍ SYSTÉM S VYUŽITÍM TECHNOLOGIE AJAX

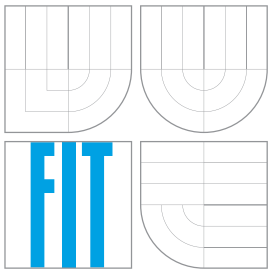
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAN HLAVIČKA

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

MODULOVÝ REDAKČNÍ SYSTÉM S VYUŽITÍM TECHNOLOGIE AJAX

MODUL MANAGEMENT SYSTEM WITH USING AJAX TECHNOLOGY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAN HLAVIČKA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR ČÁSTEK

BRNO 2009

Abstrakt

Práce je rozdělena do dvou částí. První diskutuje o webových technologií používaných v dnešní době a webových standardech. Podrobněji se zaměřuje na technologii Ajax a její využití. V druhé části se práce věnuje analýze, návrhu a implementaci plně objektového, modulárního redakčního systému.

Abstract

This work is separated into two section. First section discusses about web technologies and standards using at this time. In more detail is focused on technology called Ajax and it's usage. In second section work is focused on analysis, design and implementation full object-oriented, modulary editorial system.

Klíčová slova

webové technologie, webové standardy, Ajax, CMS, modulový systém, OOP, návrh, implementace

Keywords

web technologies, web standards, Ajax, CMS, modul managment system, OOP, design, implementation.

Citace

Jan Hlavička: Modulový redakční systém s využitím
technologie AJAX, bakalářská práce, Brno, FIT VUT v Brně, 2009

Modulový redakční systém s využitím technologie AJAX

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Petra Částka.

.....
Jan Hlavička
19. května 2009

Poděkování

Rád bych využil příležitost k poděkování panu Ing. Petru Částkovi za jeho vedení a odborné připomínky.

Obsah

Kapitola 1

Úvod

V dnešní době snad už každý ví, co je to internet, k čemu slouží a jak se k němu připojí. Poskytovatele internetových služeb se předhánějí v nabídkách a díky velké konkurenci se kvalita internetových služeb nejenom pro domácnosti neustále vyvíjí a zkvalitňuje. Situace s cenami za internetové služby jsou podobné, díky tomu ve většině dnešních moderních domácností přístup k internetu existuje.

To samozřejmě nahrává do karet samotným webovým aplikacím, jelikož se na poli marketingu stávají velice silným hráčem. Obchodní zástupci, ale i programátoři, hledají neustále nové a nové způsoby, jak své produkty odlišit od konkurence a nabídnout svému klientovi něco navíc. To má za důsledek vznik stále nových technologií, které ve své atraktivitě, interaktivitě, ovladatelnosti a kvalitě, už dokáží konkurovat tzv. *offline* aplikacím, tj. aplikace, které ke svému chodu nepotřebují připojení k internetu. Navíc mají výhodu stále aktuální verze aplikace a minimální nároky na uživatelův systém. Uživateli v podstatě stačí webový prohlížeč, který je ve všech nejpoužívanějších operačních systémech naprostým standardem.

V této práci se podrobněji zaměřím na webovou technologii Ajax. Popíšu i jiné, často používané, webové technologie a také se budu věnovat praktické části této práce, tedy aplikaci nazvané *Weblymp*.

Ajax je zkratka pro Asynchronous JavaScript and XML, což je termín, označující webovou technologii, která je schopna pomocí javascriptu dotazovat na webový server bez nutnosti načtení celých stránek. Pomocí této technologie lze webové stránky navrhnout více interaktivně a zpříjemnit uživateli práci s obsahem webových stránek. Mezi nejčastější využití technologie Ajax bezesporu patří kontrola formulářů a doplňování slov. Více se zabývám technologií v sekci 4 *Ajax podrobněji* na straně 13.

V kapitola 2, nazvané *Webové technologie* na straně 4, popisuji dnešní webové technologie. Krátce nastíním historii vývoje webových technologií a věnuji se některým, dle mého názoru nejdůležitějším, webovým technologiím. Další kapitola má název *Moderní přístupy k webovým aplikacím* na straně 10. V této kapitole píši o některých zásadách a standardech, kterým je třeba při vývoji webových aplikací věnovat pozornost. Nastíním problematiku CMS *(Content Managment System)* aplikací v porovnání s aplikacemi programované *na míru* a také zde popíši problematiku vyhledávačů a v dnešních dnech velice často opakované SEO - *Search engine optimalization*. Webové technologii Ajax se podrobně věnuji v kapitole nazvané *Ajax podrobněji* na straně 13 a v poslední kapitole *Návrh, implementace a popis aplikace* na straně 17 se věnuji praktické části této práce.

Kapitola 2

Webové technologie

Termínem *webová technologie* se označují nástroje a postupy pro tvorbu a prezentaci webové aplikace. Může se jednat o programovací jazyk, framework, serverové aplikace nebo kombinaci některých nástrojů, např. Ajax. Webových technologií je velké množství, zaměřím se na ty, které považuji za zásadní a nejpoužívanější v dnešních webových aplikacích, jako je samotné (X)HTML, CSS nebo nejoblíbenější programovací jazyk na webové aplikace - PHP. Při tvorbě webové aplikace by si každý programátor už při návrhu měl rozhodnout, jaké webové technologie bude používat. V nejčastějších případech bude přístup k webové aplikaci pomocí prohlížeče, tedy využití systému World Wide Web [3], který jako standard pro zobrazování informací využívá technologie (X)HTML. V dnešních dnech se dle mého názoru nejčastěji používá skriptovací PHP a pro uložení dat databáze MySQL, což se odráží i na hostingových službách, kde kombinace PHP a MySQL je v drtivé většině nabídek.

2.1 Historie

Historie webových aplikacích jde ruku v ruce s historií internetu a informačních technologií celkově. První myšlenky využití počítačů pro komunikaci publikoval **Vannaver Bush** (1890-1974) v Americkém vědeckém časopise *The Atlantic Monthly* už v roce 1945. Na jeho práci přímo navazoval **Theodor Holm Nelson** (narozen 1937), který v roce 1963 poprvé použil termín *hypertext* a v roce *1965* jej zveřejnil. První nástroj k tvorbě hypertextu prezentoval vynálezce počítačové myši **Dr. Dougles C. Engelbart**.

Počítačová síť, tedy komunikace počítačů mezi sebou, má počátky v roce 1968 v Národní výzkumné laboratoři ve Velké Británii. Instalace této sítě se ale nikdy nerozšířila a zůstala pouze na pomezí jedné budovy. Ministerstvo obrany USA finančně podpořilo výzkum agentury ARPA, která v roce 1969 vynalezla experimentální síť, kterou pojmenovala ARPANET. Internet byl až do poloviny osmdesátých let omezen pouze na vládní a vojenské organizace, tak se ani příliš nerozvíjel.

Do Evropy se rozmach internetu dostal hlavně díky švýcarskému institutu pro jaderný výzkum CERN, kdy v roce 1980 **Tim Berners-Lee** využil myšlenku hypertextu pro usnadnění sdílení informací mezi výzkumníky. V roce 1989 vlastnil CERN největší internetový server v Evropě a v listopadu 1990 **Tim Berners-Lee** předvedl první prototyp web serveru, pojmenoval ho *httpd* a zanedlouho, 6. srpna 1991, na adrese http://info.cern.ch/, spustil první webové stránky. První prohlížeč byl zároveň i prvním WYSIWYG (what you see is what you get) editorem, který nesl název *WorldWideWeb*, ale brzy byl přejmenován na *Nexus*. V roce 1984 bylo k internetu připojeno pouze 1000 uživatelů, v roce 1992 to bylo již

více než milion počítačů, které byly připojeny na internet. V roce 1992 byl taky započat vývoj prvního grafického prohlížeče pod názvem *Mosaic*. Celý prohlížeč byl psán na půdě *NCSA* (National center for supercomputing applications) a na vývoji se podíleli **Mark Andreessen** a **Eric Bina**. První verze grafického prohlížeče byla zdarma uvolněna 22. dubna 1993. Koncem roku byl prohlížeč už i pro verze Apple Macintosh a Mircrosoft Windows. Z prohlížeče *Mosaic* v budoucnu vzniká velice populární *Netscape*, vyvíjeno stejnou skupinou vývojářů, která se ale přejmenovala na *Netscape Communications*.

V roce 1993 nastal velký rozmach internetu, byl vyvinut standard WWW a již existovalo kolem 50 serverů podporující WWW. Od roku 1993 do 1995 se zdvojnásobil počet uživatelů k internetu, odhaduje se, že již v roce 1995 je připojeno na 20 miliónů počítačů a v roce 2000 už bylo připojeno více než 300 miliónů počítačů.

Důležitým datem je rok 1994, kdy vznikla instituce nazvaná *WWW Consorcium* (W3C), která se podílí na rozvoji WWW a v pozdějších letech definuje několik standardů pro různé webové technologie. Jejím ředitelem je tvůrce WWW **Tim Berners-Lee**.

2.2 Přehled webových technologií

Webové technologie, které se dnes používají je celá řada. V posledních letech se na internetu objevují stále častěji interaktivní aplikace, aplikace, které streamují videa, virtuální prohlídky a mnoho dalších. Každá taková aplikace vyžaduje používání jiných technologií. Technologie, které budu popisovat, jsou většinou zdarma a možná i proto je jejich oblíbenost vysoká.

Všechny technologie jsou na tvorbu nebo prezentaci WWW, tedy takové aplikace, které ke spuštění používají internetový prohlížeč, který je již dnes, troufám si říct, na všech osobních počítačích. Není vzácností, že se internetové prohlížeče instalují i do přenosných a mobilních zařízení. Osobně považuji právě tyto přístroje za velice marketingově atraktivní a věřím, že v následujících měsících a letech budeme svědky stále větší optimalizace aplikací právě pro tato zařízení. Operátoři mobilních služeb nabízejí stále výhodnější mobilní připojení k internetu, ale instalace WiFi do zařízení otvírá možnosti i pro vysokorychlostní připojení. Ve větších městech začíná být naprostým standardem připojení zdarma na několika místech, takže dostupnost k internetu je z mobilního zařízení každým dnem snažší a prostor pro vývoj je obrovský.

V dalších podkapitolách popíšu základní programovací a značkovací jazyky pro tvorbu webových stránek a nástroje, kterými lze docílit interaktivity či přehrávání videí.

2.2.1 HTML, XHTML

HTML (HyperText Markup Language) a XHTML (Extensible HyperText Markup Language) jsou naprostým standardem pro zobrazování webových stránek. A v dnešní době nejpoužívanější jazyk pro prezentace na internetu pomocí webových prohlížečů. Historie vývoje sahá do 80. let minulého století a jejím hlavním tvůrcem je **Tim Berners-Lee**. HTML během svého vývoje prošlo několika verzemi, poslední a nejzásadnější, dnes stále používaná, je verze HTML 4.01. XHTML je dalším stupněm vývoje.

Jedná se o značkovací jazyky pro tvorbu hypertextových dokumentů. Ve své definici obsahuje různé značky pro tvorbu tabulek, seznamů, obrázků, odkazů, odstavců a spoustu dalších elementů vhodné pro prezentaci textu. Verze XHTML vznikla hlavně z důvodu kompatibility. Kombinuje HTML a XML, které zaručuje rozšiřitelnost se zachováním kompatibility se staršími verzemi prohlížečů, tedy alespoň podle W3C a jeho standardů. Problém je,

že téměř každý internetový prohlížeč v dnešní době má své specifika a chyby, takže vývojáři webových aplikací stále musí trávit spoustu času nad testováním. Více o validitě webu a kompatibilitě píšu v kapitole 3.4 Validní výstupy na straně 11. Dalším důvodem, proč byl XHTML vyvinut, je oddělení struktury a obsahu dokumentu od formátování a stylování. Standardy staršího HTML mají ve svých definicích spoustu atributů, které určující grafické vlastnosti, ale ty v případě zpracování nebo vyhledávání v dokumentu nejsou žádoucí. Proto podle standardů pro XHTML lze pouze strukturalizovat obsah a jeho grafické prvky se vkládají do externích souborů pomocí CSS.

HTML i XHTML je rozděleno do hlavních dvou bloků, které jsou ohraničeny elementy `<head></head>` a `<body></body>`.

V tagu `head` jsou prezentovány META informace o konkrétní webové stránce a uživatel je standardně nevidí. Tyto informace využívají prohlížeče pro indexování a získání informací. O vyhledávačích píšu více v kapitole 3.5 Vyhledávače a SEO na straně 12.

Obsah dokumentu, který návštěvník dané webové stránky má obvykle zobrazený, je definovaný mezi tagy `body`. Celý dokument v html by měl být strukturovaný a dobře navržený. V dnešní době se jako grafický stavební kámen používá tag `div`, což nebylo a není vždy pravidlem, nicméně hlavně ve verzi XHTML odborníci tento přístup maximálně doporučují. Problém je ale s již už zmíněnou rozdílnou interpretací jazyka různými prohlížeči, ale i verzemi prohlížečů. Někteří programátoři věnující se HTML a tzv. *rozřezu webu*, což je v podstatě přenesení grafiky do HTML značek s využitím CSS, nedají dopustit na rozvržení webu pomocí tabulek, tedy pomocí skupině tagu `table, th, tr, td`. Tento způsob osobně považuji ale za zastaralý. Odborníci se shodují, že tabulky jsou v definici XHTML pro výpis dat a nikoliv pro stavbu grafiky celé stránky.

V dnešní době je většina webových stránek generována pomocí jiných programovacích jazyků, respektive překladačů jazyků. Tyto překladače generují výstupní kód a zpravidla jim bývá právě HTML nebo XHTML. Pomocí správné kombinace těchto překladačů se statickými informacemi v HTML, lze docílit velké interaktivity, modulárnosti i flexibilnosti.

2.2.2 CSS

HTML i XHTML jako takové pouze definují strukturu a obsah dokumentu a vzhled výsledného dokumentu se upravuje pomocí technologie CSS (Cascading Style Sheets).

První definici, z které vychází dnešní CSS, definoval, tehdy ještě CHSS (Cascading HTML Style Sheets), **Håkon Wium Lie** v roce 1994 a v prosinci 1996 bylo oficiálně představeno CSS level 1 a v květnu roku 1998 byl veřejně představen CSS level 2.

Vývoj CSS a HTML, potažmo XHTML, je velice úzce spojen. Vývojáři spojení kolem W3C, kteří definují standardy nejenom pro tyto jazyky, se snaží o co nejjednoznačnější definice a tím ulehčit programátorům webových aplikací práci. Velkým problémem je zpětná kompatibilita se staršími prohlížeči, proto jejich úpravy v definicích standardu se v praxi odráží až po určité době, která nebývá zrovna krátká.

CSS by se mělo oddělovat do samostatných souborů. Může být součástí HTML dokumentu, ale není to doporučované. Jednotlivé styly lze vpisovat přímo do HTML tagů. Jejich priorita bývá většinou vyšší, než definování stylu pro daný tag v externím souboru. Již z principu oddělení textu a grafiky, bych osobně tento způsob nikdy nedoporučil a např. pro určení priority v CSS použil atributu `!important;`.

Dnešní webové aplikace by měly dbát na variabilnost svých návštěvníků a technologie CSS je naprosto ideální pro zajištění co nejširší kompatibility. Dnešní uživatelé internetu používají nejrůznější prohlížeče a je více než vhodné, aby ve všech prohlížečích byla grafická

prezentace obsahu shodná, minimálně bezchybná. Jednotlivé prohlížeče proto upravily své překladače, aby vývojáři byli schopni definovat speciální vlastnosti pouze pro daný prohlížeč. Pro prezentaci uvedu následující kód:

Kód 1 ukázka css kódu a rozdílností v prohlížečích

```
div#menu {
    color: black;
    #color: yellow;
    _color: blue;
}
```

Jak je vidět v kódu 1 je parametr `color` pro tag `div` s `id=‘‘menu’’` definován třikrát, pouze s rozdílným prefixem. Jedná se o parametr barvy a každý prohlížeč takto zapsaný kód interpretuje jinak. Jako základ je parametr bez jakéhokoliv prefixu, tedy barva černá. Pokud je před parametrem uveden znak `#`, tak prohlížeč *Internet Explorer 6 i verze 7* interpretuje jako prioritnější hodnotu a znak _ před atributem *Internet Explorer 6* upřednostní i před `#`. Jinými slovy, například v prohlížeči *Mozilla Firefox*, bude výsledný tag `div` v barvě černé, v *Internet Explorer 6* bude modrý a v *Internet Explorer 7* bude žlutý. Jak lze vidět, rozdílnost není pouze v různých rodinách prohlížečů, ale i u jednotlivých verzí. Tyto rozdíly jsou markantní obzvláště u prohlížeče od firmy *Microsoft*, jelikož starší verze příliš nedbaly na standardy definované W3C a v nových verzí se již interpretace HTML i CSS ke standardům přiblížila.

2.2.3 Javascript

Javascript je naprostým základem pro interaktivní weby. V jeho názvu část slova *java* je pouze z marketingových důvodů a s programovacím jazykem *Java* nemá kromě jména a podobné syntaxe nic společného. Syntaxe jazyka je podobná jazykům C/C++, Java atd. Je to jazyk objektový a velmi důležitou vlastností je, že se jazyk spouští na straně klienta. Tzn, že se nejdříve od serveru stáhne veškerý obsah webové stránky, včetně zdrojových kódů pro JavaScript a jeho interpretace nastává až v prohlížeči klienta. S tím jsou spojená jistá rizika, proto má JavaScript spoustu omezení, například nemůže pracovat se soubory. Javascript je i základem pro Technologii Ajax, kdy pomocí JavaScriptových objektů lze kontaktovat server.

Historie JavaScriptu začíná ve společnosti *Netscape Communications* a vyvíjel jej **Brendan Eich** pod názvem *Mocha*. Později byl přejmenován na *Live Script* a nakonec nesl finální jméno známé dnes, *JavaScript*. Důležitým faktorem, proč se tato technologie přejmenovala právě na *JavaScript*, je také to, že v roce 1995 kdy byl *JavaScript* implementován do prohlížeče *Netscape* byla přidána podpora *Java Technology*.

Významným důvodem proč je JavaScript tak silný je, že umí pracovat s DOM strukturou html stránky. Tím lze docílit téměř jakýchkoliv změn v uživatelském prohlížeči bez zásahu interpretačního jazyka na straně serveru jako je *PHP* nebo *ASP*. Díky JavaScriptu je možné přidávat do webových aplikací různé dynamické prvky, které lze využít v galerii, menu položkách, ale např. v reklamních bannerech.

Tato technologie je velice kvalitní a silná, přesto v různých článcích a fórech JavaScript kritizují. Problematika rozdílné interpretace technologií v prohlížečích se nevyhnula ani JavaScript. Z tohoto důvodu často napsaný kód přesně podle referencí jazyka funguje správně

pouze na některých prohlížečích, rozdíly v interpretaci jsou převážně v pojmenování atributů.

Kód 2 ukázka JavaScriptu ošetřující rozdílné pojmenování atributů v prohlížečích

```
function getElementClass(elm) {
    cn = (tr.getAttribute(,,class‘‘)) ? ,,class‘‘ : ,,className‘‘
    return tr.getAttribute(cn);
}
```

Z ukázky kódu 2 je vidět, že JavaScript nejdříve zjistí, zda je nastavený pro element `elm` parametr `class`. Pokud takovou definicí nezná, zvolí parametr `className`. Tímto jednoduchým kódem lze docílit získání atributu `class` pro různé typy prohlížečů. Parametr `class` je ve většině prohlížečů implementovaný, ale v `Internet Explorer` se musí používat parametr `className`.

Naštěstí v tomto jazyce je napsáno spousta frameworků, které usnadňují a zrychlují práci s JavaScriptem. Osobně považuji za velice silný nástroj, který používám i v projektu, opensource *Dojo Toolkit* [8]. Jeho pomocí lze docílit velice kvalitních a dobře vypadajících efektů.

2.2.4 PHP

Význam slova PHP se postupně s vývojem měnil. V roce 1995, kdy bylo první oficiální vydání, PHP bylo zkratkou pro *Personal Home Page*. Při přepsání paresearu tohoto jazyka v roce 1998, vznikla verze PHP3 a nastalo přejmenování, které se používá do dnešní doby. PHP tedy znamená: „PHP: Hypertext Preprocessor“.

Jedná se o jeden z nejoblíbenějších a nejpoužívanějších programovacích jazyků pro vývoj webových aplikací. Dokazuje to procento poskytovatelů nabízející své služby s podporou této technologie.

Ve verzi PHP5 byl přepracován model objektového programování, který na oblíbenosti jenom přidal. Zdaleka není dokonalý, ale k efektivnějšímu programování rozhodně přispívá. Tento jazyk má velkou podporu ze strany jednotlivých programátorů, díky tomu existuje již spousta řešení na různé problémy.

Většina OpenSource projektů, dotýkající se nějakým způsobem webových aplikací, na svoji implementaci využívá tento programovací jazyk.

V jazyce samotném je zaimplementováno obrovské množství funkcí a jelikož je překladač napsaný v programovacím jazyce C, jsou často funkce velice efektivní.

Velkou výhodou této technologie je kombinace s technologií (X)HTML. Snadno lze totiž kombinovat psaní HTML tagů a v případě nějaké proměnné oblasti, lze využít právě PHP.

Tento jazyk je velice benevolentní, proto i málo zkušení programátoři jsou schopni rychle napsat fungující kód, ale má to svá úskalí. Ne každý kód je stejně efektivní a snadno lze špatně napsaným kódem zahltit celý server. Dalším problémem je také bezpečnost, bez potřebných zkušeností a znalostí programátor ve své implementaci neošetří všechny možnosti a zanechá v programu *„bezpečnostní díru“*, kterou případný útočník najde. V posledních verzích se vývojový tým PHP hodně zaměřuje právě na bezpečnost a jeho snaha je přimět programátory používat základní nastavení PHP takové, aby minimalizovali možné nedostatky.

PHP považuji za tak zásadní, že se dle mého názoru zasloužil velkou měrou o rozvoj webových aplikací po celém internetu.

2.2.5 Flash

Flash je koncepčně odlišná technologie od ostatních a jeho využití není pouze ve webových aplikacích. Tato technologie přidává možnost do webových projektů zapojit různé druhy médií, včetně jejich streamování. Nejčastější použití v praxi je na reklamních bannerech nebo např. server YouTube, kde je technologie použita na streamování videí.

Pomocí Flashe lze naprogramovat celý web, ale není to moc časté. Více se využívá pro grafické prvky nebo některé webové hry. Možnosti technologie jsou ale obrovské a v každé nové verzi přibývá mnoho funkcí, které ještě možnosti rozšiřují.

2.2.6 ASP.NET

ASP.NET (Active Server Pages) od firmy Microsoft je dalším stupněm vývoje ASP. Jeho velkou výhodou je .NET Framework a CLR (Common Language Runtime). Programátoři proto mohou aplikace psát v jakémkoliv jazyce podporující CLR, např. Visual Basic NET, JScript.NET, C# aj.

Vývoj takových aplikací probíhá v *Microsoft Visual Studio*. Výhodou je to, že spousta věcí je již naprogramována a programátor má ušetřeno spoustu práce. Tato technologie je placená a osobně ji považuji za méně oblíbenou než PHP.

Kapitola 3

Moderní přístupy k webovým aplikacím

V dnešní době existuje velké množství technologií, kterými lze tvořit webové aplikace. Proto se v této kapitole zaměřím na obecnější pohled na věc, ne na konkrétní způsoby v jednotlivých technologiích. Každá technologie má své výhody i nevýhody, a proto je potřeba vždy před začátkem vývoje webové aplikace rozhodnout, co má obnášet a jak působit na návštěvníky. Určitě se použijí jiné technologie na server s interaktivními hrami a jiné na eshopy nebo prezentace firmy. Co bývá častým problémem v neprofesionálních řešeních je právě volba dané technologie. Programátoři se snaží často předvést své schopnosti tam, kde spíše návštěvník hledá jasné a srozumitelně strukturované výstupy.

3.1 Srozumitelnost a přehlednost webových aplikací

Než se začne vytvářet webová aplikace nebo webová prezentace, je vždy potřeba mít jasno, pro koho je výsledek určen. Pokud se bavíme o webových aplikacích a stránkách, je vždy potřeba vědět, co je standardem a snažit se ho v co ho dodržovat. Návštěvníci daného webového serveru vždy ocení více přehlednost, strukturovanost a srozumitelnost, než červeně křiklavé texty s nejrozmanitější škálou barev v pozadí.

Každý web by měl mít k dispozici uživatelům známé a sympatické nástroje, jako je mapa webu, navigátor nebo například vždy přístupný odkaz na domovskou stránku. Možnou výjimkou mohou být specifické weby pro odlišné účely nebo s použitím netradičních technologií.

3.2 Redakční systémy, CMS

Umístit své prezentace na internet dnes už není žádný problém ani pro naprostého začátečníka. Existuje mnoho serverů, které nabízejí své prostory a služby pro seberealizaci. Pokud se ale od prezentace očekává určitá kvalita, je vždy lepší obrátit se na profesionály v oboru. Existují mnohá řešení a vždy je třeba mít co nejjasnější představu o určení webu a jeho budoucím obsahu.

V současnosti už málokdo řeší své internetové prezentace statickými stránkami, které je potřeba znovu a znovu ručně nahrávat na servery, aby došlo k aktualizaci dat zobrazené uživatelům. Pokud má být webová aplikace umístěna na internet standardní a podobná většině ostatních, tedy prezentace textů, galerie, novinky nebo třeba ankety, je vhodné

využít již některý ověřený systém. Je již z čeho vybírat a například možnosti v opensource projektech jsou obrovské.

Redakční systémy a CMS jsou ideálním řešením právě pro tyto účely, tedy jednoduché i složité prezentace. Jejich modulárnost nabízí návštěvníkům velkou flexibilnost a při kvalitní správě webu lze docílit dobrých výsledků. Síla redakčních systémů a CMS spočívá v jednoduché editaci a správě obsahu, formulářů, anket a všeho, co daný systém nabízí. Taková aplikace musí být schopna dodržovat aktuální standardy a měla by administrátorům co nejvíce ulehčovat práci. Naprosto nedílnou součástí takových systémů jsou WYSIWYG editory, které pomáhají generovat správné výstupy textů. Problémy můžou nastat v případě velkého projektu se speciálními požadavky, kde je vyžadována velice netradiční funkce. Dobrý CMS je dostatečně obecný na to, aby implementace nového modulu nebyla problém. Takové řešení není vždy ideální. Příkladem může být eshop. Jeho náročnost a složitost je natolik velká, že je vhodné řešit tento problém individuálně. Není to ale dogma. Rozhodujícím faktorem je dostatečná obecnost systému, využití vhodných technologií a samozřejmě jeho rychlost.

3.3 Řešení na míru

Pokud má být výsledek specifický, odlišný od běžných projektů, je dobré vytvořit řešení na míru.

Vhodným příkladem jsou složité speciální aplikace jako např. intranetové systémy, internetové bankovnictví, sázkové kanceláře nebo online hry s rozhraním přístupným přes webový prohlížeč. V těchto příkladech by programátoři měli přistupovat k řešení naprosto stejnému jako je návrh offline aplikací v kompilovaných jazycích. Rozhodně tím nechci říct, že je potřeba psát všechny knihovny znova, ale jádro aplikace by mělo být přizpůsobeno potřebám dané aplikace. Nikdy nebude pro internetové bankovnictví vhodné používat jádro systému vyvinuté pro redakční systém nebo CMS. V tomto konkrétním případě se musí celá aplikace zaměřit především na dostatečnou bezpečnost a důvěryhodnost.

Jelikož jsou tyto aplikace velice specifické, dochází často k využívání netradičních technologií. Ve velkých projektech není výjimkou použití jiných či upravených serverových služeb, vhodných pro danou aplikaci.

Protože se jedná o specifické služby, jsou vyžadovány netradiční nástroje a uživatel je nucen pro správný chod nainstalovat nestandardní nástroj, který mu přístup umožní.

3.4 Validní výstupy

Několikrát jsem se již zmínil o problémech různých interpretací jednotlivých technologií prohlížeči. Nejznámější z těchto problémů je okolo technologií CSS a JavaScript. Při využívání těchto technologií je programátor vždy povinnen svoji práci velice často a precizně kontrolovat a testovat ve všech používaných nebo alespoň nejvíce používaných prohlížečích.

Validní webová aplikace či validní webová stránka je taková stránka, která dodržuje dohodnuté standardy v dané specifikaci. Každá stránka musí na svém úvodu specifikovat, jakou standardizaci implementuje, v jaké verzi a jakém jazyce. Jak takový validní dokument vypadá lze nalézt na stránkách W3C.

Zjistit, zda jednotlivé stránky nebo styly jsou validní, lze pomocí různých validátorů. Před releasem stránky by měl ten, kdo stránku vyvíjí, ji analyzovat právě takovým validátorem.

S příchodem XHTML se situace trochu zlepšila a obzvláště při dodržování validních výstupů v jednotlivých technologiích lze eliminovat rozdílnost. Ne vždy lze vystačit pouze se základními a hlavně validními postupy. Někdy nelze jinak, než sáhnout po řešení, které není validní. Trochu si protiřečím, ale ukázka v kódu 1, kdy pomocí CSS lze jednotlivým prohlížečům definovat jiné vlastnosti, je nevalidní. Co by ale opravdu každá aplikace, kterou zkušení programátoři vyvíjejí, měla splňovat, je validní (X)HTML, které v některých případech ovlivní i výsledný pohled vyhledávače na danou webovou stránku.

3.5 Vyhledávače a SEO

Aby byla webová aplikace přístupná a dohledatelná pro veřejnost, musí být někde zaevidovaná. O to se starají vyhledávače, které indexují všechny weby jim přístupné a poté je svým návštěvníkům přidávají do výsledku hledání.

Existuje spousta kriterií, které ovlivňují výsledky vyhledávaní a v posledních letech se díky službám SEO (search engine optimalization) rozrůstají marketingové *„trháky“*, které slibují umístění v prvních pěti výsledcích. Majitel takového webu neusiluje vždy o takové prvenství. Obzvláště u internetových obchodů je to výsledek velkou měrou odrážející se na počtu prodaných kusů v daném internetovém obchodě.

SEO je často těmi, kteří alespoň trochu znají tento výraz, překrucováno. V dnešních době obchodní zástupci nabízejí služby SEO a slibují okamžité umístění na prvních místech ve vyhledávačích. Často jim chybí argumentace k dosažení výsledku. SEO rozhodně nesmí být chápáno jako jednorázová činnost. Dá se *optimalizovat* výstup webových stránek tak, aby byl pro vyhledávače co nejjasnější, ať už pomocí správných meta informací, nadpisů nebo využití přepisu url a využívat tzv. *přátelské url*. Osobně považuji umístění ve vyhledávačích jako jednoduchý násobek: *kolikrát je stránka vidět * jak kvalitně je stránka vidět*. Jinými slovy, nejdůležitější pravidla a optimalizace jsou takové, aby reference, tedy zpětné odkazy na stránky, byly co nejčetnější. Tak vyhledávače mají častý přístup k webovým stránkám a násobek *kolikrát je stránka vidět* roste. Při přístupu na stránky je potřeba mít stále kontrolované a optimalizované výstupy, nejenom meta informace, ale i samotný obsah. Důležité je dodržovat i správnou strukturu.

Hodnotící systém vyhledávačů se často mění a dnes už téměř každý důležitý vyhledávač nabízí možnosti pro reklamní umístění, ale finančně náročné. Pokud má být internetová aplikace úspěšná, neobejde se v dnešní době bez značné investice, ať už pro zkvalitnění výstupu a umístění ve vyhledávačích nebo jiné marketingové činnosti, podporující známost webu.

Kapitola 4

Ajax podrobněji

Ajax, někdy uváděn jako AJAX (Asynchronous JavaScript and XML), je skupina webových technologií, které umožňují asynchronně v pozadí bez přerušení komunikovat se serverem, ať už za účelem zápisu dat nebo změny aktuálního obsahu. Využívání této technologie vede k zatraktivnění a zinteraktivnění webové aplikace a zkvalitní webových služeb díky asynchronnímu módu. Základním stavebním kamenem je objekt *XMLHttpRequest*, v jehož názvu je XML, ale nemusí být vždy nutné technologii používat.

Někdy je slovo Ajax používáno na místech, kde se rozhodně o tuto technologii nejedná. Uživatelé i programátoři se domnívají, že stačí pomocí javascriptu změnit částečně obsah webové stránky a používají tím technologii Ajax. Naopak, tato technologie nemusí změnit naprosto nic, může například asynchronně na pozadí ukládat na server aktuální rozestavení panelů uživatele a obsah stránky nechat beze změn. O technologii Ajax se jedná pouze tehdy, pokud prohlížeč na pozadí komunikuje se serverem a přitom nemusí měnit obsah webových stránek.

4.1 Historie

Termín Ajax existuje oficiálně od roku 2005. Asynchronní spojení mezi prohlížečem a serverem existuje už od poloviny devadesátých let minulého století, ať už v technologiích jako je *IFrame element* v HTML představený v *Internet Explorer* v roce 1999 nebo *Java applets*, které už v roce 1996 uměly komunikovat asynchronně se serverem na pozadí prohlížeče. 5. dubna 2006 W3C vydalo první koncept *XMLHttp* objektu, pro definování základního standardu této technologie.

4.2 Ajax z pohledu uživatele

Z uživatelského hlediska je tato technologie velice výhodná. Pokud je vhodně a kvalitně implementována, je i časovým přínosem. Hlavní výhodou je, že komunikace se serverem probíhá na pozadí, uživatel nemusí být přerušován od své práce. Velmi vhodný je příklad s anketou. Uživatel zvolí možnost a zaktualizuje se pouze oblast s anketním lístkem, nikoliv celý obsah. Problém může nastat, pokud uživatel má standardně vypnutý JavaScript. Bez této technologie algoritmy napsané pro Ajax fungovat nebudou a webová stránka s tím musí počítat a nabízet uživateli alternativu. Nevhodným řešením je pouze oznámení uživateli, ať si JavaScript povolí. V případě mobilního prohlížeče tuto možnost mít ani nemusí.

4.3 Ajax z pohledu programátora

Nejdůležitější co si musí programátor uvědomit je fakt, že nemusí mít uživatel vždy povolený JavaScript. Proto je vždy nutno implementovat dvojí řešení. Dobrý příklad jsou formuláře, kdy pomocí Ajaxu již v průběhu plnění formuláře lze definovat, zda je zadaná hodnota platná, validní nebo jedinečná. Pokud má uživatel JavaScript vypnutý, tyto validace neprobíhají a je potřeba na to myslet při zpracování formuláře. V tomto příkladě lze také demonstrovat výhody Ajaxu oproti samostatnému JavaScriptu. Validovat pole lze už na úrovni JavaScriptu. Zjistit pomocí JavaScriptu, zda je hodnota formuláře zadaná a splňuje kritéria např. emailové adresy, není problém. Velkou výhodou Ajaxu je fakt, že na validace dané položky stačí napsat jedna funkce v interpretačním jazyce, např. PHP, která se volá jak při ukládání nebo zpracování celého formuláře, tak při editaci dané položky.

Už v úvodu jsem řekl, že nejdůležitější prvek technologie Ajax je *XMLHttpRequest*, pomocí kterého se řídí veškerá asynchronní komunikace se serverem. Trochu problém je jeho implementace v různých prohlížečích. Nejjednodušším způsobem, ale i základním, přesto dostatečně efektivním, je získávání objektu pomocí funkce v kódu 3.

Kód 3 Získání objektu pro asynchronní přístup z prohlížeče na server

```
function ajaxCreateHttp() {
    var xmlHttp;
    try  {
        // Firefox, Opera 8.0+, Safari
        xmlHttp=new XMLHttpRequest();
    } catch (e) {
        // Internet Explorer
        try {
            xmlHttp=new ActiveXObject(,,Msxml2.XMLHTTP‘‘);
        } catch (e) {
            try {
              xmlHttp=new ActiveXObject(,,Microsoft.XMLHTTP‘‘);
            }  catch (e) {
               alert(,,Your browser does not support AJAX!‘‘);
               return false;
            }
        }
    }
    return xmlHttp;
}
```

Z kódu 3 je vidět, že pro získání objektu jsou použity výjimky. Díky komentářům lze i zjistit, že rozdílná implementace je hlavně v *Internet Exploreru*, kde je místo *XMLHttpRequest* objektu implementován objekt *ActiveXObject*. Naštěstí je to jediný rozdíl a práce se získaným objektem je už identická.

Pomocí tohoto objektu lze komunikovat se serverem, posílat parametry i volit typ metody mezi *GET* a *POST*. Při zaslání dotazu server vrátí odpověď, která se na úrovni JavaScriptu zpracovává. Odpověď může být ve formátu prostého textu nebo XML, záleží na implementaci. Obecně platí, že pokud chceme měnit bloky kódů už vytvořeného podle (X)HTML pravidel, je vhodné použít prostý text, který není potřeba zpracovávat, pouze se

vymění za předešlý. Pro zpracování různých dat je vhodné použít strukturované XML, které lze pomocí různých funkcí zpracovávat. Odpověď ze serveru může nést pouze informaci, zda daná operace proběhla úspěšně a programátor nemusí už tuto informaci zobrazovat uživateli. Opět záleží na situaci a vhodnosti.

4.4 Výhody a nevýhody technologie Ajax

Výhody i nevýhody této technologie už byly lehce naznačeny. Mezi hlavní výhody bezesporu patří:

- Zinteraktivnění a větší komfort webových aplikací a stránek
- Snížení zátěže na straně server
- Zkvalitnění nabízených služeb
- Automatické ukládání zpracovávaných dat uživatelem

Jednotlivé výhody se prolínají a záleží na jejich interpretaci, principiálně se dá shrnout do dvou hlavních – komfort a menší zátěž. Vždy je mnohem příjemnější zjistit, zda uživatelské jméno je již použito při jeho zadávání, než několikrát odesílat celý formulář. To samozřejmě ovlivňuje i zátěž na straně serveru.

Mezi hlavní nevýhody patří asi problematika ohledně SEO. Pokud některý obsah a klíčová slova jsou zobrazena pouze po nějaké interaktivitě uživatele, může nastat, že je vyhledávač vůbec nemusí zaindexovat. Tato chyba se dá eliminovat na úplné minimum a spíše záleží na znalostech a kvalitě programátora. Další nevýhodou je již zmíněná potřeba řešit vždy dvě možnosti. Zda uživatel má podporu všech technologií pro Ajax či nikoliv. I zde záleží na tom, jak se programátor s problémem vypořádá. Je důležité a vhodné řešit problematiku již na úrovni návrhu a implementovat systém tak, aby se nemusel psát dvojí kód.

Tato technologie obecně má více méně jenom výhody. Platí zde pravidlo všeho moc škodí. Pokud se bude technologie používat naprosto všude, nevýhod bude přibývat každou další funkcí. Vždy je potřeba mít na mysli to, k čemu má výsledek sloužit a zohledňovat standardy, na které jsou uživatelé zvyklí.

4.5 Využití technologie Ajax v praxi

Využití Ajaxu v praxi je stále častější. Na nejznámějších a největších portálech, jako je Google, Seznam a dnes už i Centrum, je využití znát na první pohled. Uživatel si může dynamicky spravovat vzhled, obsah i pozice obsahu. Další velice časté použití lze vidět v různých vyhledávacích formulářích. Při psaní dochází k našeptávání a uživateli je nabídnut seznam možných slov, která může vyhledávat. Ověřování formulářů, řazení, stránkování a mnoho dalších možností. Možností, kde lze tuto technologii použít, je nepřeberné množství a spíše jsou omezené znalostmi a možnostmi programátora.

Často se tato technologie uplatňuje na místech, kde je zbytečná. Její implementace zabírá mnoho času a ne vždy je výsledek ideální. Při návrhu aplikace by si měli vývojáři stanovit cíle, zda chtějí vytvořit webovou interaktivní aplikaci, kde využití Ajaxu nemá omezení nebo zda chtějí vytvořit webovou stránku s podporou Ajaxu, kdy uživatel nemusí přemýšlet nad nestandardním ovládáním.

4.6 Konkurence

Přímého konkurenta tato technologie nemá. Samozřejmě záleží na pohledu srovnání. Tato technologie využívá jiné technologie, proto například přímé srovnání s PHP nebo ASP není zcela správné.

Možné srovnání by mohlo být s technologií Flash nebo SilverLight. Nicméně nalézt spravedlivá kritéria srovnání je téměř nemožné. Všechny technologie mají jiné určení. Ajax má výhodu v tom, že téměř všechny prohlížeče mají již v sobě integrované nástroje pro jeho podporu, navíc koncept stránek se neliší od již známých standardů. Aplikace ve Flashi a v SilverLightu mají mnohem blíž k typickým aplikacím na osobním počítači, ale můžou být prezentovány přes web. Jejich grafické možnosti jsou téměř neomezené a s každou novou verzí jsou sofistikovanější. Tyto technologie v sobě zahrnují i nástroje na streamování médií, což Ajax nezvládne. Ajax, Flash, SilverLight a jiné technologie nabízejí uživateli interaktivní přístup k webovým stránkám, ale rozhodně se nedá říci, že některá technologie je lepší než jiná. Každá technologie se hodí na jiné druhy implementace a hlavně se dají kombinovat, což mnohdy bývá nejsilnější a nejkvalitnější výsledek.

Kapitola 5

Návrh, implementace a popis aplikace

Tato kapitola se věnuje praktické části bakalářské práce. Úkolem bylo naprogramovat plně objektový, modulární redakční systém s využitím technologie Ajax. Postupně se budu věnovat analýze, vyjmenuji použité nástroje při implementaci, obecně popíšu samotnou aplikaci, návrh aplikace i její implementaci.

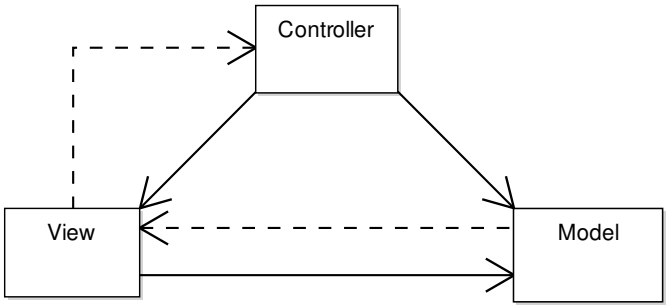
Jako název aplikace jsem zvolil *WebLymp*. Toto slovo je odvozeno od slov *Web* a *Olymp*. Slovo Olymp je z toho důvodu, že server, na kterém je uložen repozitory k svn, má název Zeus.

5.1 Analýza

Před samotným návrhem aplikace a následnou implementací jsem zanalyzoval zadaný úkol. Snažil jsem se vytyčit mé hlavní cíle a zvolit vhodný návrhový vzor. Hlavní kritéria, která při analýze vyplynula a byla je třeba zahrnout do samostatného návrhu jsou:

- plně objektový systém
- modulárnost
- snadná rozšiřitelnost a upravitelnost jádra systému
- template systém
- obecný systém ukládání dat
- vícejazyčnost

Systém musí být plně objektový. Musí být i modulární, bylo potřeba navrhnout takový systém, který bude schopný sám sebe rozšiřovat pomocí zásuvných modulů. Důraz při návrhu musí být kladen na oddělení samotného jádra systému od jeho rozšiřitelných funkcí. Systém musí být moderní a lehce přizpůsobitelný novým trendům a standardům a tedy jeho jednotlivé části jádra by měly být na sobě nezávislé a jednoduše upravitelné bez možnosti ovlivnění funkčnosti jiné části systému. Systém musí být velice jednoduše aplikovatelný na různé grafické návrhy a být v tomto směru velice flexibilní. Jeho vlastnosti musí být takové, aby jednotlivé prvky byly editovatelné stejně snadno jako celý koncept grafického návrhu.



Obrázek 5.1: Návrhový vzor: Module-View-Controller

Pro kvalitní objektový výsledek a obecnou kompatibilitu systému je potřeba navrhnout základní třídu tvořící objekt, který bude schopný sám sebe ukládat a takový objekt pouze rozšiřovat velice obecnou definicí. Za tímto účelem bylo nutno vytvořit obecný systém na ukládání dat.

Díky definování si základních požadavků systému jsem se mohl rozhodnout pro vhodný návrhový vzor, který by mi byl vodítkem pro samotný návrh aplikace. Návrhový vzor, který jsem použil jako základní vodítko je Module-View-Controller a je zobrazený na obrázku 5.1. Jedná se o oddělení několika na sobě nezávislých vrstev:

- datová vrstva
- aplikační vrstva
- výstupní vrstva

Návrhový vzor mi dovoluje jednotlivé vrstvy upravovat bez ovlivňování jiných. Již při prvních definování požadavků jsem chtěl vytvořit template systém. O veškeré grafické výstupy se stará *výstupní vrstva*, která je v systému implementována jako template systém. Zjistil jsem, že je velice výhodné oddělit samostatné vrstvy pro práci s databází. V případě změny typu databáze nebo kombinací některých typů nebude potřeba většího zásahu do systému, pouze se naimplementuje nová knihovna pro přístup k datům v jednotlivých databázích. Jádro systému a jednotlivé moduly jsou implementovány jako *vrstva aplikační*.

5.2 použité nástroje

Při výběru použitých nástrojů jsem se rozhodoval celkem jednoznačně. Všechny zvolené nástroje musí být zdarma a s dobrou dokumentací. Jak ale popisuji v kapitole 2 vzhledem k obecné preferenci programovacího jazyka *PHP* jsem i já zvolil tento jazyk jako hlavní. Z tématu bakalářské práce vyplývá, že v systému existuje podpora pro technologii *Ajax*. Systém má oddělenou výstupní vrstvu, využívající technologie *XHTML* s kombinací *CSS*.

V systému používám v současném stavu tři extérní knihovny:

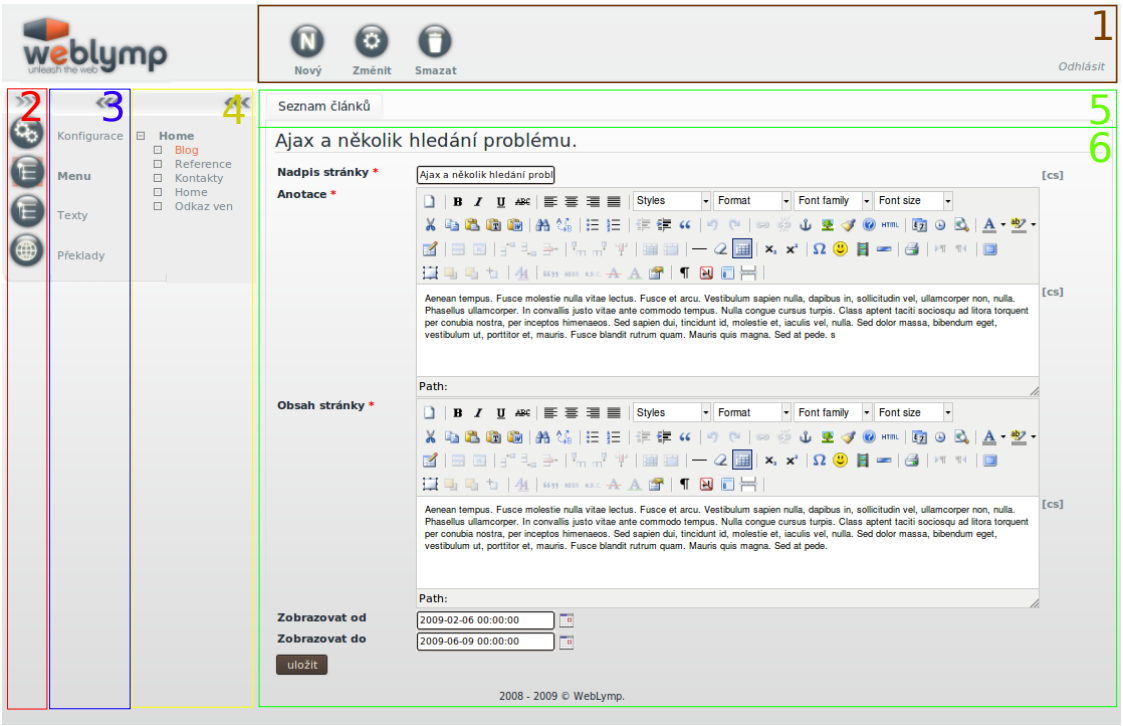
1. DatePicker - JavaScriptový výběr datumů [10]
2. Dojo Toolkit - JavaScriptový toolkit [8]
3. TinyMCE - WYSIWYG editor [7]

5.3 obecný popis

5.3.1 Základní uživatelský popis

Aplikace je rozdělena na administrační a veřejnou část. Obě části jsou kresleny přes template systém a jejich prvky nejsou tedy pevně dané a můžou být jednoduše změněny.

Veřejně přístupná část je obsahově plně editovatelná. Samozřejmě záleží, jaký template je použit a jak je implementován. Template by ale měl implementovat všechny možnosti systémy.



Obrázek 5.2: Ukázka administračního rozhraní WebLymp

Administrační rozhraní je navrženo tak, aby byla maximalizována oblast, kde probíhají jednotlivé editace. Na obrázku 5.2 je ukázka administračního rozhraní. V hnědé oblasti číslo 1 je ovládací panel pro prvky v levém ovládacím panelu, který je ve žluté oblasti číslo 2. Horní ovládací panel nemusí být vždy přístupný, záleží jaký modul se edituje. V ukázce je modul menu, horní panel tedy slouží pro editaci jednotlivých položek menu. V oblasti číslo 1 se také v pravé části nachází odkaz na odhlášení ze systému.

Panely s čísly 2, 3 a 4 jsou základními panely celé administrace. Jsou rozděleny do tří částí s tím, že části v oblastech 3 a 4 se dají libovolně schovat podle potřeb uživatele. v červené oblasti číslo 2 jsou vždy ikony jednotlivých modulů, které jsou v modré oblasti číslo 3 popsány. Žlutá oblast číslo 4 nemusí být zobrazena, opět záleží na definici modulu, zda má k dispozici rozšiřitelný panel.

V oblastech 5 a 6 probíhá samotná editace s tím, že oblast 5 je nepovinná a jsou zde zobrazeny záložky, které jsou plně definovatelné v jednotlivých modulech. Oblast 6 je již plně věnována jednotlivých editacím a zde je prostor pro výstupy jednotlivých modulů.

5.3.2 Popis funkcí

V administrační části jsou k dispozici na výběr čtyři hlavní moduly, které mají své pozice v ovládacím panelu na levé straně. Jedná se o tyto moduly:

- Konfigurace
- Menu
- Texty
- překlady

Konfigurace

V modulu *Konfigurace* jsou tři záložky, pomocí kterých lze volit obsah v tomto modulu.

- Konfigurace webu
- Práce s databází
- O weblympu

V první záložce *Konfigurace webu* je už podle názvu jasné, že se zde konfigurují jednotlivé možnosti samotného systému a konkrétního webu. Každý modul může mít definované své konfigurační hodnoty a podle toho jsou zde nabízeny. Názvy jednotlivých položek odpovídají jejich funkčnosti. Systém má svoji základní skupinu konfiguračních dat, která je nazvaná obecně *config*. Jsou zde nastavení pro některé META informace k webu a hlavně možnost zapínání modulů povolených pro jednotlivé weby. Zapnutím nebo vypnutím modulů se může ovlivnit i nabídka nabízených typů obsahu pro jednotlivé položky menu. V současné chvíli je v systému implementován pouze jeden administrační účet, ale v dalších verzích by tento modul měl být pro normální administrátory skrytý a měl by k němu přístup pouze ten, kdo web instaluje.

Další záložka je *Práce s databází*. Při její inicializaci proběhne kontrola základních tabulek, které jsou nezbytné k funkčnosti webu. Pokud dané tabulky chybí, systém je automaticky vytvoří. V této záložce je taky správa nad jednotlivými tabulkami definovaných v jednotlivých modulech. Každý modul může mít definici svých tabulek a podle zapnutých modulů se zde objevují. Pokud tabulka v databázi chybí, systém ji nabídne vytvořit, tato funkce je naprogramovaná přes *Ajax callback*. V současném stavu systém neumí mazat tabulky a ani pracovat s upgrady tabulek, je ovšem připravený k takové implementaci.

V poslední záložce jsou zobrazeny informace o aktuální verzi WebLympu a autoru. V návrhu je zde prostor pro vytvoření další položky pro automatické zjišťování na externím serveru aktuální verzi systému a případný její upgrade.

Menu

Menu je základní modul v systému. Pomocí tohoto modulu se tvoří stromová struktura celého webu. Pokud není vytvořena ještě žádná položka, systém ji při prvním pužití modulu nabídne vytvořit.

Formulář, pomocí kterého se edituje tato položka se opakuje často v celém systému. Je to základní, editovatelný formulář pro *DbObject* (definovatelný objekt, který umí sám sebe

uložit do databáze). Více o DbObjectu je napsáno v kapitolách 5.4 Návrh aplikace a 5.5 Implementace na stranách 22 a 27. Jednotlivé Menu položky se editují, mažou a přidávají pomocí tlačítek v horním panelu. Při editaci položek jsou k dispozici následující hodnoty:

- Název položky
- Typ obsahu
- Maska

Název položky je vícejazyčná položka, lze editovat jednotlivé jazyky pomocí přepínače vpravo od položky. Typ obsahu definuje, jaký obsah bude mít daná položka k dispozici. Počet typů obsahů, které jsou zde k dispozici, jsou ovlivněny zapnutými moduly. V současném stavu jsou k dispozici 4 typy obsahu:

1. Seznam článků
2. Webová stránka
3. Interní odkaz
4. Externí odkaz

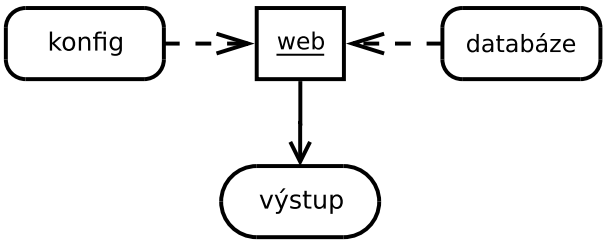
Seznam článků je seznam webových stránek s možností definovat anotace a období, v kterém je článek přístupný. Editace článků využívá tabulkového widgetu, který se opakuje v celém systému. Tento widget může definovat callbacky na jednotlivé položky, případně i Ajaxové, jako je tomu v definici databází v modulu Konfig. Tato tabulka je nadefinována spolu s příkazy na přidávání, editování a mazání jednotlivých článků. Celá tabulka je Ajaxová, v případě velkého množství článků se objeví pager, který je rovněž implementován za pomocí Ajaxu. Samotná editace článků probíhá přes formulář *DbObjectu*. K editování textových částí je použit WYSIWYG editor, konkrétně *TinyMCE* [7].

Dalším typem obsahu je *Webová stránka*. Tento obsah je nejzákladnější a je to jednoduchá prezentace textu, její editace je podobná článkům v seznamu článků, její pole jsou však omezeny o položky, které nejsou potřeba.

Poslední dva typy obsahů jsou *Interní odkaz* a *Externí odkaz*. Jedná se pouze o odkazy, které někam směřují. Interní odkaz se edituje výběrem jiné položky v systému menu pomocí formuláře *DbObjectu*. Externí odkaz ke své editaci využívá textové pole, do kterého lze napsat normální adresa ve formátu *http://www.adresa.cz*.

Při editaci položky menu je kromě názvu položky a typu obsahu k dispozici položka *Maska*. Tato hodnota definuje, která maska, neboli template, bude použit pro vykreslení. Tímto mechanizmem lze docílit rozdílné grafiky pro jednotlivé položky menu nebo různá volání modulů podle aktuální pozice položky. Jednotlivé template se ukládají do adresáře webu a systém už sám nabídne jeho zobrazení v tomto formuláři. Vhodné použití je hlavně na rozdílnost stránek typu *Home* a ostatních podstránek. V konfiguraci webu lze nastavit standardní masku pro web, která bude použita jako základní. Z výběru ve formuláře, kromě konkrétních masek a hodnoty *„standardní“* je ještě hodnota *„Stejný jako nadřazený“*. Tato možnost použije pro vykreslení masku zděděnou od rodičovské položky.

V tomto modulu je největší prostor pro další rozšiřování. Částečně připravený je i návrh přípon, kdy bude k objektům, které mají možnost ke svému obsahu připínat další obsah, definovaný jako přípona, zobrazena editace i právě těchto přípon. V této části bude využito právě technologie *Ajax*.



Obrázek 5.3: Obecné rozvržení komponent

Texty

Tento modul je velice jednoduchý, jedná se o editaci statických textů, které jsou použity při vytváření template pro systém. Každý takový text má svůj identifikátor, který je volán v template, a hodnotu. Samozřejmostí je vícejazyčnost. Pro editaci položek je zde použita Ajaxová tabulka.

Překlady

Tento modul je pouze pomocný a případný administrátor by jej k dispozici nikdy neměl. Jedná se o editaci všech textů, které systém má definováno, a jejich lokalizace ve všech dostupných jazycích. Jsou to texty, které lze nalézt v administračním rozhraní systému, ať už se jedná o chybové hlášky, názvy jednotlivých položek nebo i názvy modulů. Pro správnou funkčnost je potřeba mít nastavené práva na soubory s možností zápisu.

5.4 Návrh aplikace

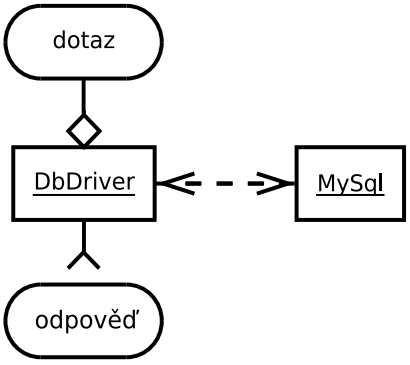
Při návrhu aplikace jsem postupoval hlavně s ohledem na oddělení částí a vycházel jsem z návrhového vzoru *Module-View-Controller*, viz obrázek 5.1. Při postupu jsem tedy oddělil části pracující s databází, vykreslovací a část, která je jádrem celé aplikace.

Z obrázku 5.3 je patrné, že hlavní komponentou, která se stará o funkcionalitu webu je komponenta nazvaná *web*. Komponenta sama analyzuje uživatelovi požadavky, o nastavení se stará komponenta *konfig* a data z databáze připravuje komponenta *databáze*. Po inicializaci komponenta *web* zavolá příslušný template a uživatelem zvolená akce je vykreslena. O to se stará *template systém*, tedy komponenta *výstup*.

5.4.1 Databáze

Přístup k databázi je tvořen přes jednu třídu nazvanou *DbDriver*. Tato třída nepracuje přímo s databází, je pouhým rozhraním k přístupu do databáze za účelem jednoduché výměny typu databáze a editování pouze této vrstvy. Při inicializaci si vytvoří všechna připojení k databázi, které reprezentují jednotlivé objekty. V současné chvíli systém implementuje pouze jedno připojení a to připojení k databázi typu *MySql*.

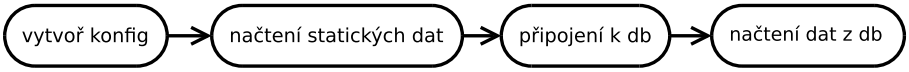
Při dotazu na databázi je tedy v systému potřeba vždy dotazovat komponentu *DbDriver*, která komunikuje s aktuální databází a vrací odpověď. Schéma dotazu na databázi je na obrázku 5.4.



Obrázek 5.4: Průběh dotazu na databázi

5.4.2 Konfig

Při návrhu této komponenty bylo potřeba vyřešit problém priorit. Konfig pracuje s databází a některé hodnoty načítá právě z databáze, na druhé straně, aby bylo možné připojit se k databázi, je potřeba mít již načtené některé statické konfigurační data potřebné k autentizaci a výběru databáze.



Obrázek 5.5: Jednotlivé kroky při vytváření konfigu

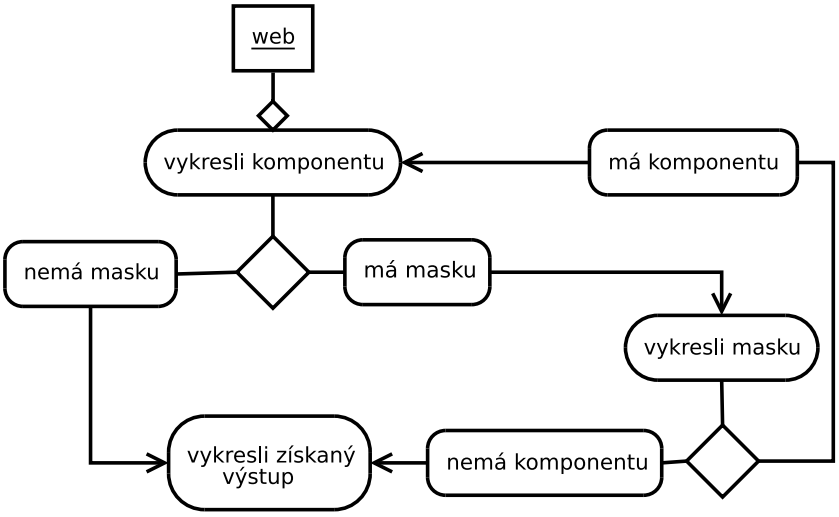
Tento problém jsem vyřešil dvoufázovým načítáním konfigu. Samotné vytváření konfigu je třífázové, ale pouze dva kroky jsou získávání konfiguračních hodnot. Všechny kroky jsou zobrazeny na obrázku 5.5. Při inicializaci webu se jako jedna s prvních akcí vytvoří konfig. Ten při svém vytváření nejdříve načte konfigurační data ze statického souboru u webu, projde všechny moduly, zjistí, zda mají definované nějaké konfigurační soubory, pokud ano načte jeho defaultní hodnoty. Poté se vytvoří připojení k databázi, ta již využívá konfigu k načtení autentizačních hodnot. Po vytvoření připojení k databázi konfig inicializuje svoji poslední část a to načtení hodnot z databáze, tyto hodnoty přebíjejí hodnoty defaultní z definic modulu.

Zmínil jsem zde, že moduly mohou definovat konfigurační hodnoty. Tyto hodnoty jsou editovatelné přes modul *Konfig*. Ukázku definic proberu v kapitole 5.4.4 Moduly.

5.4.3 Výstup

Další základní komponentou aplikace je komponenta, která se stará o vykreslování výstupu. Tuto roli má na starosti *Template Systém*. Při návrhu komponenty jsem se snažil o co největší editovatelnost výstupů bez jakýchkoliv zásahu do jiných komponent systému. Princip template systému je zobrazen na obrázku 5.6.

Slovně popsáno, první inicializaci kreslení podnítí komponenta *web*, ta rozhodne, která maska bude jako základní, podle aktuálního obsahu. Při vykreslování může dojít k situaci, že template systém nalezne nějakou systémovou komponentu jako je např.: položka menu, vyhledávací box, navigátor aj. V takové situaci template systém zavolá vykreslení na komponentu a zde se proces opakuje až do té doby, než jsou všechny komponenty při vykreslování masky zpracovány a vykresleny.



Obrázek 5.6: Princip fungování template systému

Při vykreslování mají masky k dispozici všechny hlavní komponenty web, databáze i konfig, pomocí kterých lze získávat informace ze systému pro případné rozhodnutí v algoritmu.

V základní masce webu jsou definovány také *callbacky*, pomocí kterých lze v některých místech volat funkce jednotlivých modulů. Jejich implementace je vysvětlena v kapitole 5.5.6 Callbacky na straně 33.

5.4.4 Řídící vrstva – komponenta Web

Jednoznačně nejdůležitější vrstva celé aplikace je vrstva řídící. Částečně zahrnuje již popsaný konfig, ale její hlavní jádro tvoří třída, která je pojmenována *web*.

Tato třída má odkazy na všechny důležité prvky v celém systému včetně databáze a konfigu. Jedná se o systém modulový, proto je řídící vrstva rozšiřitelná jednotlivými moduly a některé často opakující se prvky, jako jsou tabulky, mají své *widgety*.

Uživatel nebo administrátor musí někde definovat obsah, pro tuto funkci je navržený *DbObject*, který díky své definici a flexibilnosti je základním stavebním kamenem každého webu v tomto systému.

Základní třída

Třída *web* v sobě neskrývá žádné složité algoritmy, pouze spravuje pořadí všech inicializací, má v sobě uloženy všechny odkazy a je součástí každé další třídy v systému. Její nejdůležitější role je analýza uživatelských informací. Tato třída rozezná, zda se jedná o Ajax volání, administrační část nebo veřejnou část. Je zde implementován také systém na tvoření a analýzu url adresy. Všechny adresy tvořené v systému musí projít přes tuto funkci.

Tuto třídu rozšiřuje třída *admin*. Třída je zděděná a pouze upravuje některé funkce tak, aby bylo v celém systému jednoznačné, v které části webu se právě nachází uživatel. Administrační část má složitější masku a musí dodržovat některé pravidla. Důležité pravidlo je priorita vykreslování. Proto je systém získávání a vytváření masky lehce odlišný od základního ve veřejné části webu.

Knihovny

Knihovna je v systému takový prvek, který je neodseparovatelný od systému a tvoří jeho jádro. Každá obecná funkce musí být implementována v knihovně. Všechny moduly i widgety můžou s přítomností dané knihovny vždy počítat a není potřeba řešit různé možnosti. Jako knihovna je také implementována třída *web*, *konfig* nebo *DbObject* a třídy z nich zděděné. Knihovny deklarují základní funkce pro moduly a widgety a všechny komponenty musí být z oněch tříd zděděny. Mezi nejdůležitější knihovny patří:

- base - implementace základních funkcí a tříd pro systém
- data - definování datových typů pro systém, využívaných hlavně v DbObjectu
- array - práce s poli, implementace Quick Sort nad asociovaným dvojrozměrným polem
- html - definice HTML výstupů. Samotné template tyto funkce využívat nemusí, jednotlivé komponenty ale ano.
- js - definování funkcí pro práci s JavaScriptem a Ajaxem
- module - deklarace a definice základní třídy pro jednotlivé moduly
- widget - deklarace a definice základní třídy pro jednotlivé widgety
- template - template systém
- web - základní řídicí třída

Toto je pouze výčet některých knihoven v systému implementovaných. Základní knihovny byly navrženy ještě před samotnou implementací, v průběhu implementace se rozšiřoval nejenom jejich počet ale i knihovny samotné.

Widgety

Widget byl definován při návrhu jako prvek systému pro často opakující se komponenty nebo složitější komponenty, které se nedají implementovat v masce. Widget je objekt, kterému se při incializace předá nastavení, které zpracuje spolu s přiřazenými daty a při zavolání vykreslí. Widget může využívat všechny knihovny a je považován jako jádro systému, tedy každý widget je přítomen ve všech verzích aplikace.

Důležitá část každého widgetu je nastavení. Při definici widgetu se definuje základní nastavení, které je použito pro případ nedefinováni požadavků při inicializaci. Pokud jsou předána nějaká inicializační data ke konfiguraci, systém data spojí s definicí widgetu. Tím je docíleno vždy kompletní nastavení i možnost úprav jednotlivých widgetů. Je to vhodné pro použití často proměnných prvků jako jsou styly, počty nebo typy. Příkladem je počet prvků na jedné stránce v tabulce nebo definování stylu pro záložky.

Mezi nejčastěji používané widgety patří:

- formuláře
- menu
- tabulky
- záložky

Každý widget musí být zděděn ze základního definovaný v knihovnách. Ten již v sobě implementuje mechanizmy pro komunikaci se systémem nebo práci s nastavením jednotlivých widgetů.

Moduly

Moduly jsou na sobě nezávislé, mohou se však nvazájem rozšiřovat. Každý modul, podobně jako widgety, musí být zděděn z hlavní třídy modul, která v sobě definuje základní funkce pro komunikaci s řídící třídou. Moduly můžou při implementaci využívat widgetu i knihoven, nesmí však spoléhat na jiné moduly. V praxi to znamená, že zapnutí jakékoliv kombinace modulů musí fungovat.

Moduly se dělí do dvou základních skupin:

1. Moduly s vlastním administračním rozhraním
2. Moduly bez rozhraní, skryté nebo rozšiřující

Toto základní rozdělení definuje, zda modul disponuje funkcemi na vykreslování v administrační části nebo pouze rozšiřuje definici jiných modulů. Například modul Menu má administrační rozhraní, patří tedy do první skupiny modulů. Oproti tomu modul Webpage je pouze rozšiřující a přidává definici nového DbObjectu. DbObject podle své definice má vždy zařazení. V případě WebPage možnost být přiřazen jako typ obsahu pro položku menu. O definici DbObjectu a DbObjectu samotném píšu v další části.

Zapínání jednotlivých modulů má za následek přidávání funkcí na různá místa, v současné době systém implementuje následující moduly:

- seznam článků - definuje seznam webových stránek s rozšířenými definicemi
- konfig - administrační rozhraní pro nastavení systému
- externlink a interlink - dvojce modulů, rozšiřující typy obsahu
- menu - základní modul pro stavbu struktury webu
- search - modul pro vyhledávání v systému, využívá Ajax
- texts - statické texty
- překlady - administrační rozhraní pro překlady
- webová stránka - základní modul rozšiřující typ obsahu

DbObject

DbObject je základním prvkem pro obsah. Každý *DbObject* má vlastní definici, která jej popisuje, zařazuje a přidává proměnné. Z definice takového objektu určuje, zda může být objekt přiřazen k menu položce, jestli může jako přípony přijímat jiné objekty nebo jestli může být k jiným objektům přiložen jako přípona. Jednotlivé definice se dají kombinovat podle potřeb daného objektu.

Každý DbObject má ve své definici pole proměnných. Jednotlivé typy proměnných jsou definovány v knihovně *data*. Základní typy proměnných, které DbObject může přijmout jsou:

- string
- text
- link
- number
- boolean
- date

Toto jsou jednotlivé základní typy, které jsou implementovány v systému. Každý základní typ má svoji tabulku v databázi. V systému můžou vznikat i typy odvozené, které mají definované jiný editační formulář, ale ukládá se vždy do jedné ze základních tabulek.

Každý takový datový typ má definovaný svůj editační formulář. Při editaci DbObjectu se postupuje tak, že se z definice zjistí všechny proměnné a vykreslí se. Tím vzniká vždy stejné prostředí pro editaci jednotlivých objektů a uživatel si nemusí zvykat na různé způsoby editace.

Úzce spojená komponenta s *DbObject* je komponenta, který se stará o registraci, nahrávání a ukládání objektů. Jeho název je *DbObjectManager*. Registrovat objekty se musí, aby systém zjistil, které objekty můžou být přiřazeny k menu nebo jako přípona. Tato komponent úzce spolupracuje s komponentou *DbDriver* tedy komponentou zajišťující datovou vrstvu. Některé funkce v *DbDriver* jsou implementované speciálně pro potřeby *DbObject*.

5.5 Implementace

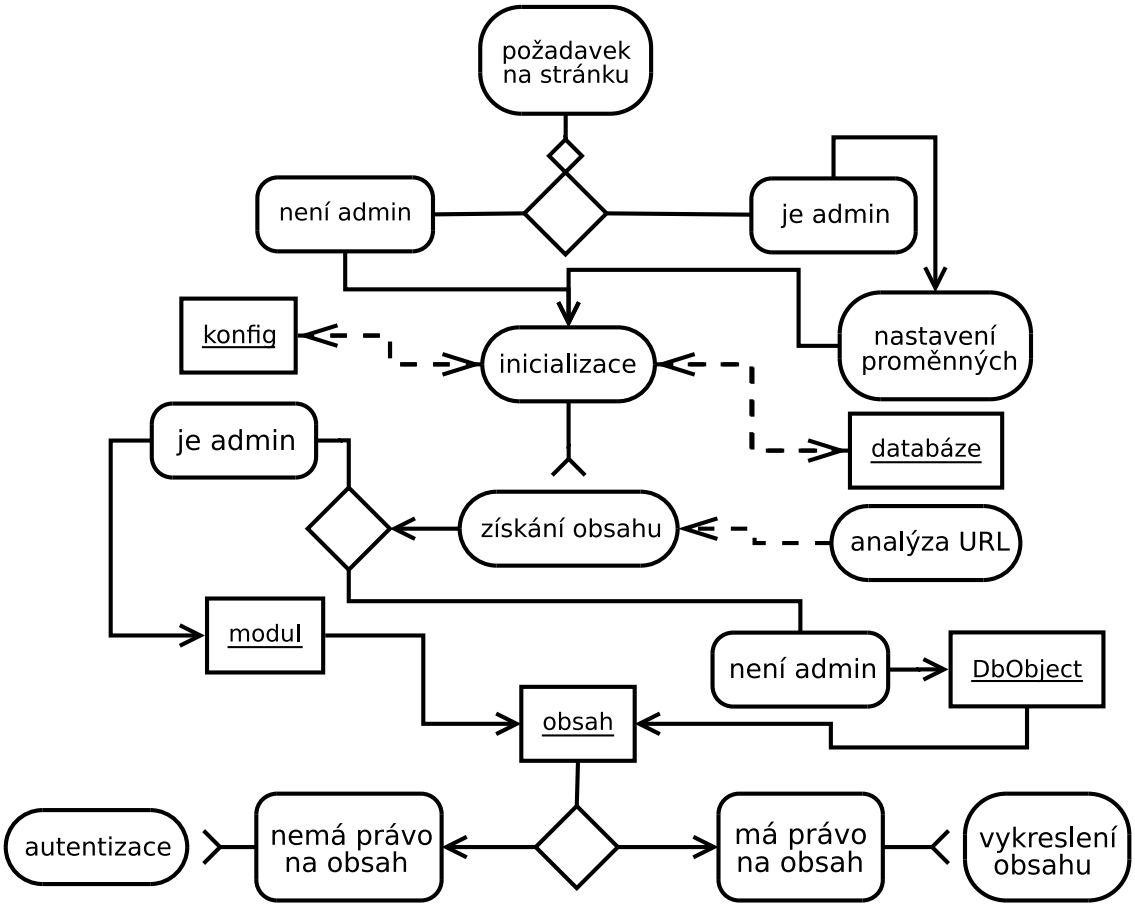
V této kapitole popíšu vybrané komponenty systému a jejich implementaci a nastíním i implementaci systému jako celek. Systém má v souborech shodnou hlavičku, která definuje kdy byl daný soubor vytvořen i jeho poslední modifikace. Celá implementace a vývoj systému je revizován pomocí SVN. Pro základní administraci jsem implementoval jednoduchý skript v programovacím jazyce *python*, který zjistí aktuální číslo revize, zjistí zda jsou v revizi změny a případně upraví informace v souboru, který informuje v záložce *O weblympu* v modulu *Konfig* o aktuální verzi systému. Verzování dodržuje pravidlo: `milestone-version-datum`. Milestone a verze jsou definovány vždy trackovacím systémem. Při implementaci používám Trac [5].

5.5.1 Systém komplexně

Běh systému může mít v současné podobě tři způsoby:

1. veřejná část
2. administrační část
3. ajax dotazy

Každé části odpovídá jeden soubor, konkrétně `index.php` pro veřejnou část, `admin.php` pro část administrační a o ajax dotazy se stará `ajax.php`. V každém souboru je jako komponenta *web* zvolena jiná třída, `wl_web`, `wl_admin` respektive `wl_ajax`. První dvě třídy `wl_web` a `wl_admin` jsou téměř identické, `wl_admin` pouze definuje speciální masku a nastavuje proměnné aby bylo jasné, že se program nachází v administračním režimu. `wl_ajax` je trochu odlišný. Nedochází k vykreslení webu ale podle proměnných předaných přes technologii Ajax se volají žádané funkce, které vrací obsah. Celý web je ale zinicializovaný a tak jeho funkčnost je identická s normálním módem.



Obrázek 5.7: Schéma systému

Z diagramu na obrázku 5.7 je patrné, jak systém funguje. Při požadavku uživatele systém rozhodne, do které skupiny patří a nastaví správnou komponentu web. Inicializace

probíhá v konstruktoru třídy, kde také proběhne získání přístupu k databázi a konfiguraci. V inicializaci systém taky načte všechny moduly, které má v aktuálním nastavení k dispozici.

Po řádné inicializaci dochází k získání a následném vykreslení daného obsahu. Systém zjistí, jaké jsou parametry v URL a podle nich dohledá správný obsah. Také kontroluje, zda má na daný obsah přihlášený uživatel právo, respektive, jestli je obsah veřejný nebo vyžaduje autentizaci. Podle toho systém rozhoduje, zda obsah vykreslí nebo vyžádá autentizaci uživatele.

V systému je spousta specifik a tohle je pouze hrubý nástin funkčnosti, některé jeho důležité součásti popisuji v dalších sekcí této práce.

5.5.2 Widgety

Widgety jsou implementovány jako objekty. Každý widget vychází ze základní třídy, kde je implementován systém nastavení widgetu. Nastavení se merguje. Každý modul může definovat své základní nastavení jako asociované pole hodnot. Proměnná, která defaultní hodnoty nese, musí být definovaná v třídě widgetu s názvem `options`. Při vytváření widgetu v systému se pak do konstruktoru může předat nastavení pro widget. Systém mergování je zobrazen v kódu 4.

Kód 4 Ukázka mergování nastavení ve widgetu

```
var $options = array();

function __construct($options = array()) {
    ...
    $this->options = array_merge($this->options, $options);
    ...
}
```

Kód 4 je úsek kódu ze základní třídy `wl_widget`. Jak je patrné, pro mergování nastavení je použita nativní funkce PHP `array_merge`. Pole nastavení je asociované pomocí hodnoty typu `string`, takže pokud v poli `$options` exituje shodná hodnota jako v obecném poli v definici třídy, je považována za prioritní.

5.5.3 Moduly

Moduly, podobně jako widgety, mají základní třídu, z které musí být odvozeny. Velice důležitou funkcí v základní třídě `wl_module`, je funkce `skipAdminMenu()`. Tato funkce je definována již v základní třídě a její návratová hodnota je `false`. Z názvu je patrné, že tato funkce rozhoduje, zda bude modul přiřazen jako hlavní a vykreslen v nabídce hlavních modulů v administrační části systému.

Pokud má modul své vlastní administrační rozhraní, musí mít grafickou reprezentaci v podobě ikony v adresáři s webem.

Každý modul může mít definované své vlastní konfigurační data nebo tabulky v databázi. K těmto potřebám jsou dva soubory v adresáři modulu a to `dbdef.php` a `config.php`. V obou případech je definice pomocí asociovaných polí.

O zpracování a prezentaci těchto dat se stará modul *konfig*. V tomto modulu jsou také definovány základní tabulky i konfigurační data pro celý systém. Tabulky databází jsou definovány vždy jako pole s asociací, která odpovídá názvu tabulky a jejich hodnoty jsou

opět asociované pole s asociací názvu sloupce, jako hodnoty jsou již atributy jednotlivých sloupců. Každá taková definice má speciální hodnotu `__meta__data__`, která definuje meta informace o tabulce, jako její verzi a případné upgrady, které slouží pro porovnání s aktuálními tabulkami v databázi a případný upgrade. Ukázka takové definice je v úseku kódu 5.

Kód 5 Ukázka definice databázové tabulky object

```
'object' => array(
    // field name
    'oid' => array (
        'type'       => 'int',
        'primary'    => true,
        'null'       => false,
    ),
    'cid' => array (
        'type'      => 'varchar',
        'length'    => 255,
        'null'      => false,
    ),
    'modify' => array (
        'type'      => 'timestamp',
        'null'      => false,
    ),
    'create' => array (
        'type'      => 'timestamp',
        'null'      => false,
    ),
   '__meta__data__' => array (
        'version' => 1,
        'upgrades' => array(
        )
    )
),
```

Definice konfiguračních dat funguje na stejném principu pomocí asociovaných polí. Jako hodnoty jsou vždy typ, defaultní hodnota a kvantita, pro případné vícenásobné zadávaní dané hodnoty. Pro některé typy jako je `list` je potřeba definovat i zdroj dat. Hodnota `list` je výčet daných hodnot editovatelných pomocí *checkboxu*. V úseku kódu 6 je předvedena jednoduchá definice pro nastavení modulů.

Funkce `_getModules` musí být definována v daném modulu jako *statická* a musí vracet vždy výčet všech možností, v uvedeném případě výčet všech dostupných modulů.

Pokud modul neimplementuje administrační rozhraní, není potřeba třídu dále programovat, pokud nejsou žádné funkce potřeba. Pokud daný modul přidává nové definice *DbObjectu*, musí být deklarován ve stejném souboru a při inicializaci modulu zavolat registraci objektu.

Pokud je modul definován s administračním rozhraním, má k dispozici několik funkcí, které pomáhají s přidáváním nejenom grafických prvků a v neposlední řadě všechny do-

Kód 6 Ukázka definice konfiguračních hodnot

```
$cfg_data = array(
...
'modules' => array('type'=>'list', 'source'=>'_getModules',
            'quantity'=>'multiple', 'default'=>array('config')),
...
);
```

stupné widgety. Za zmínění stojí funkce `addTab`, která přijmá dva parametry, hodnotu a popis. Zavoláním této funkce se zajistí přidání nové záložky v administračním rozhraní s daným popisem. Důležité je ošetřit, aby nebyly definovány dvě záložky se stejnou hodnotou. Další funkcí je `showPanel`, jejíž výstup, pokud je definový, systém přiřadí k levému panelu. Tento mechanizmus je využit například v modulu *menu*, kde je v oblasti levého panelu vykreslena stromová struktura menu položek.

5.5.4 Ajax

Technologie Ajax má v systému svoji vlastní JavaScriptovou knihovnu, kde jsou funkce na vytváření *xmlHttp* objektu. V této knihovně je definována funkce `ajaxProcess`, která jako přijmá jako parametry identifikátor nějakého elementu v DOM dokumentu a callback funkci pro získání dat. Funkce si poté sama zpracuje všechny procesy spojené s Ajax voláním a obsah elementu nahradí získanými daty.

Pro získání dat existuje třída `wl_ajax`, která je zděděna z `wl_web`. Tato třída již byla popsána v sekci 5.5.1 Systém komplexně na straně 28.

Ajax je použit v tabulkovém widgetu. Pomocí Ajaxu probíhá řazení a stránkování tabulky přes jakýkoliv obsah. Můžou zde být definována i Ajax callback volání na každý řádek tabulky, které dynamicky provádí změny v definovaných funkcí. Této vlastnosti je například využito při vytváření databázových tabulek v modulu *konfig*.

Speciální třídy jsou definovány pro modul *vyhledávání*. Pomocí této technologie probíhají v pozadí dotazy na databázi a vracejí výsledek *„našeptávače“*. Uživatel v případě psaní fráze do pole pro vyhledávání dostane od systému možnost volby již dříve jinými uživateli zadané, úspešně nalezené, výrazy. Třídy jsou definovány odděleně z důvodu co největší rychlosti provedeného dotazu. Každý dotaz probíhá po změně textu ve vyhledávacím formuláři, proto jsem se rozhodl implementovat speciální třídy, které nemusí být neustále znovu inicializovány jako normální stránka v systému.

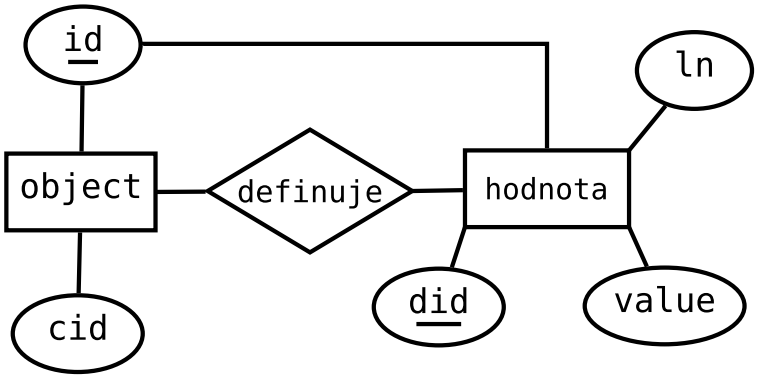
5.5.5 DbObject

Význam *DbObjectu* byl již popsán v předešlých kapitolách. Každý DbObject v systému musí mít svou definici. Definice se skládá z několika proměnných, s tím, že některé jsou povinné a musí se při definování nového objektu vždy zadat, některé musí být jedinečné. Definice je realizována jako statické pole dat, zde jsem ovšem narazil na problém v PHP.

Objekt je děděn z rodiče, který definuje všechny důležité funkce pro jeho chování. Problém ale je právě se statickými metodami a vlastnostmi v PHP, přistupuje se k nim pomocí klíčového výrazu `self::`. Kvůli zapouzdření aplikace jsem definoval funkci `getDef` jako statickou a v případě volání z jiných funkcí na objekt vždy vrátí definici rodičovského objektu, nikoliv aktuálního objektu, na kterém funkci volám. Tento problém je popsaný v

dokumentaci PHP a v dalších verzí jej řeší nová klíčová slova. V mém případě to řeším vždy předefinováním funkce v jednotlivých *DbObjectech*, tak docílím vždy vrácení správné definice.

V předešlých kapitolách jsem zmínil, že o ukládání a nahrávání objektů, tedy o komunikaci s databází, se stará třída *DbObjectManager*. Kromě toho objekty také registruje do systému. Díky definici *DbObjectu* DbOBjectManager vždy ví, která hodnota je uložena v jaké databázi. V definici hodnoty u *DbObjectu* se zadává typ, který vychází z nějakého typu základního popsaném v knihovně *data*. Tento základní typ má definovanou tabulku v databázi. DbObjectManager si tedy před uložením nebo načtením nějaké hodnoty DbObjectu zjistí, jakého typu hodnota je, a poté jej podle identifikátoru vyhledá v databazí. Na ER diagramu na obrázku 5.8 je zobrazeno schéma uložení hodnot v databázi. V tabulce *object* jsou uloženy všechny objekty v systému, jejich atributy jsou *cid* a *id*.



Obrázek 5.8: ER diagram uložení DbObjectu v databázi

cid je identifikace, o jakou definici DbObjectu se jedná. Identifikátor objektu je společný i u všech hodnot, které může DbObject definovat. Pro hodnoty jsou v databázi různé tabulky podle typu hodnoty, v současném stavu se jedná o následující tabulky:

- date - ukládání hodnot s datumem
- link - ukládání odkazu na jiné objekty
- number - ukládání hodnot typu číslo
- string - ukládání textových řetězců
- text - ukládání textových bloků

Z diagramu také vyplývá, že u hodnot jsou také identifikátory pro jazyk. Nicméně ne všechny hodnoty potřebují takový identifikátor, proto u některých tabulek není tento identifikátor přítomen.

Při registraci DbObjectu do systému DbObjectManager prochází definici a podle definice zařazuje objekt. V ukázce kódu 7 je definice DbObjectu.

Tato definice musí být napsána vždy uvnitř třídy jednotlivých *DbObjectů*. Prvních několik hodnot definuje objekt obecně. Hodnota `label` definuje označení třídy a musí být jedinečná, `cid` je jednoznačný identifikátor v systému, který je použit i při ukládání do databáze. Další tři hodnoty, které můžou nabýt hodnot pravda nebo nepravda zařazují objekt v systému, definují, zda může přijmout jiné objekty, zda může být přijmutý k jiným

Kód 7 ukázka definice DbObjectu, část definice položky menu

```
static $def = array(
  'label'        =>'db_object wl_menu',
  'cid'          =>'wl_menu_item',
  'canattached'  =>true,
  'isattachable' =>true,
  'ismenuable'   =>false,
  'localename'   =>'wl_module_menu',
  'fields' => array(
    'title'=>array(
       'type'=>'string',
       'require'=>false,
       'ml'=>true, //multilang
       'hidden'=>false,
       'require'=>true,
    ),
   'content' => array(
        'type'=> 'content',
        'ml'=>false,
        'hidden'=>false,
        'require'=>true,
        '_getHtmlForm'=>'wl_menu_item::_formContent',
    ),
    'children'=>array(
        'type'=>'objectlist',
        'require'=>'false',
        'hidden'=>true,
    ),
    ...
  )
);
```

objektům a jestli může být definován jako položka menu. Proměnná `localname` definuje třídu, z které se převezme lokalizační objekt. Většinou se jedná o třídu modulu, ve kterém je *DbObject* definován. Následují definice hodnot, které *DbObject* obsahuje. Jedná se o asociované pole, které ve svých hodnotách nese název a typ. Můžou být definovány i další hodnoty, které jsou použity při editaci DbObjectu, např. zda je položka povinná, skrytá nebo jestli jestli může mít vícejazyčné hodnoty. Lze definovat také funkci, která bude použita pro vykreslení, tato definice má identifikátor `_getHtmlForm` a v ukázce je v definici hodnoty `content`.

DbObject lze editovat voláním funkce `edit` na daný objekt. Systém s využitím widgetu formuláře vykreslí editační formulář pro editaci objektu.

5.5.6 Callbacky

Aplikace má definovaný systém callback volání. Každý callback musí mít definovaný *hook*, na který se registrují funkce. Hlavní využití callbacku je při vykreslování, například meta

data, konkrétně nalinkovaných souborů s JavaScriptem nebo CSS. Systém má definovaný hook `html-start`, který postupně volá všechny registrované funkce v okamžiku otevření tagu `head` v html dokumentu. tím lze docílit nalinkování potřebného javascriptu v jednotlivých modulech.

Při registraci funkce jsou povinné tři parametry, `hook`, `func`, `obj`, první je název definovaného hooku, druhý je název volané funkce a třetí je objekt, ve kterém je funkce definována. Jako poslední, nepovinný parametr registrace callbacku, přijímá pole hodnot. Pole je poté předáno jako parametr volané funkci.

Kapitola 6

Závěr

V první části této práce jsem obecně popsal některé standardy tvorby webových aplikací, popsal jsem některé zásadní webové technologie a podrobněji jsem se věnoval Technologii Ajax.

V další části jsem se věnoval návrhu a implementaci projektu. Projekt považuji za velmi kvalitně navržený a díky zvolenému návrhovému vzoru velmi dobře rozšiřitelný a editovatelný.

V projektu se na některých místech využívá technologie Ajax. Její přínos hodnotím kladně. Na místech, kde je implementována zrychluje práci a zlepšuje přehlednost aplikace.

Celý systém má kvalitní objektový návrh a je plně modulární. Deklarace nových modulů jsou časově nenáročné. Implementuje zajímavé widgety, které zrychlují programování celé aplikace.

Implementace nového webu jsou velice jednoduché a zkušený programátor ji zvládne velice rychle.

Původní analýza a návrh byl splněn a vzhledem ke kvalitnímu výstupu je považuji za kvalitní a správné.

Ve vývoji aplikace budu pokračovat a technologii Ajax využiji na mnoha dalších místech. Mezi prvními dalšími kroky bude implementace souborového systému a možnosti tvorby DbObjectů typu galerie a soubory. Zaměřím se také na úpravy WYSIWYG editoru pro pohodlnější ovládání a implementace pluginů, pro lepší a jednodušší práci s textem.

Literatura

[1] *Cascading Style Sheets [online]*. nadace WIKIMEDIA, Květen 2009, [cit. 5.5.2009]. Dostupné na World Wide Web: http://en.wikipedia.org/wiki/Css

[2] *JavaScript [online]*. nadace WIKIMEDIA, Květen 2009, [cit. 6.5.2009]. Dostupné na World Wide Web: http://en.wikipedia.org/wiki/JavaScript

[3] *World Wide Web*. nadace WIKIMEDIA, Květen 2009, [cit. navštíveno 5.5.2009]. Dostupné na World Wide Web: http://en.wikipedia.org/wiki/World_wide_web

[4] *XML (Extensible Markup Language) [online]*. nadace WIKIMEDIA, Květen 2009, [cit. 6.5.2009]. Dostupné na World Wide Web: http://en.wikipedia.org/wiki/Xml

[5] Edgewall Software: *The Trac*. Dostupné na World Wide Web: http://trac.edgewall.org/

[6] MCCONNEL, S.: *Doknalý kód, umění programování a techniky tvorby software*. Computer Press, a.s., 2006, ISBN 80-251-0849-X.

[7] Moxiecode Systems AB: TinyMCE, [online]. [cit. 12.5.2009]. Dostupné na World Wide Web: http://tinymce.moxiecode.com

[8] The Dojo Foundation: Dojo Toolkit, [online]. [cit. 12.5.2009]. Dostupné na World Wide Web: http://www.dojotoolkit.org

[9] W3C: HyperText Markup Language [online]. [cit. 5.5.2009]. Dostupné na World Wide Web: http://www.w3.org/MarkUp/

[10] www.frequency-decoder.com: Unobtrusive Date-Picker Widget, [online]. [cit. 12.5.2009]. Dostupné na World Wide Web: http://www.frequency-decoder.com/2006/10/02/unobtrusive-date-picker-widgit-update/

[11] www.webdesign.paysoft.cz: Historie internetu [online]. [cit. 10.5.2009]. Dostupné na World Wide Web: http://www.webdesign.paysoft.cz/clanky/2006/historie-internetu

Dodatek A

Obsah CD

Na přiloženém CD nosiči jsou soubory se zdrojovými kódy a grafická maska pro jeden ukázkový web.

Pro instalaci systému je potřeba zkopírovat obsah CD nosiče do adresáře web serveru a ve složce ukázkového webu v souboru localconfig.php nastavit přístup k databázi a heslo pro administrátora, které je hashováno přes PHP funkci SH1. Uživatelské jméno s přístupem do administrační části je vždy `zeus` a nastavené heslo je `test`.

Na CD nosiči se nachází sql soubor pro naplnění databáze testovými daty. Podrobnější informace o instalaci jsou přiloženy na nosiči v souboru `readme`.