TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: B 2612 – Elektrotechnika a informatika
Studijní obor: 1802R022 – Informatika a logistika

WYSIWYG editor pro sazbu validního XHTML

WYSIWYG editor for typesetting valid XHTML

Bakalářská práce

Autor: **Petr Hamtil**
Vedoucí práce: Ing. Pavel Tyl
Konzultant: Ing. Jiří Týř

V Liberci 13. 5. 2008

POZOR důležité!!!

Tato stránka musí být před vytvořením vazby nahrazena originálním zadáním!

Originál zadání práce Obsahuje zadání práce s originálem podpisů (vedoucí ústavu a děkan fakulty). Zadání DP a BP je oboustranné, v případě jednostranného tisku se počítá jako jediná stránka (číslo 2), v případě oboustranného tisku je počítáme jako dvě strany (3 a 4).

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé BP a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum

Podpis

Poděkování

Na tomto místě bych rád poděkoval vedoucímu bakalářské práce Ing. Pavlu Tylovi a konzultantovi Ing. Jiřímu Týřovi za připomínky a cenné rady při vypracovávání bakalářské práce.

Také chci poděkoval rodině a přátelům za veškerou podporu při práci na tomto projektu a psaní práce.

Abstrakt

Cílem bakalářské práce je seznámit se hlouběji s problematikou tvorby XHTML webových stránek prostřednictvím WYSIWYG editorů vytvořených v programovacím jazyce JavaScript a dalšími příslušnými technologiemi, dále prozkoumat stávající řešení a pokusit se vytvořit a otestovat vlastní WYSIWYG editor.

Při studiu technologií využitelných pro WYSIWYG editor jsem se zaměřil na prozkoumání moderních metod a technologií, které jsou vytvořeným editorem využívány.

Při popisu WYSIWYG editorů jsem se zaměřil především na popsání principu, na kterém pracují.

Vytvořený WYSIWYG editor byl koncipován pro snadnou rozšířitelnost o nové funkce a kompatibilitu s nejpoužívanějšími prohlížeči.

Abstract

The goal of this work is to familiarize with the issues of XHTML web pages creating by WYSIWYG editors programmed in JavaScript language and other appropriate technologies, to analyze existing solutions and to create and test own WYSIWYG editor.

Among tasks related to WYSIWYG editors I focused on analysis of state of the art methods and technologies used by these editors.

Within describing WYSIWYG editors I focused especially on their work principles.

Final WYSIWYG editor was conceived for a simple extension of new functions and maximal compatibility with nowadays most frequently used browsers.

Klíčová slova

WYSIWYG editor, JavaScript, WWW, XHTML.

Keywords

WYSIWYG editor, JavaScript, WWW, XHTML.

Obsah

Seznam obrázků

Seznam termínů a zkratek

CSS Cascading Style Sheets – jazyk pro popis způsobu zobrazení stránek napsaných v jazycích HTML, XHTML nebo XML.

DOM Document Object Model – objektově orientovaná reprezentace XML nebo HTML dokumentu.

DTD Document Type Definition – jazyk pro popis struktury XML případně SGML dokumentu. Omezuje množinu přípustných dokumentů spadajících do daného typu nebo třídy.

GPL GNU General Public License – licence pro svobodný software.

HTTP HyperText Transfer Protocol – internetový protokol určený původně pro výměnu hypertextových dokumentů ve formátu HTML.

LGPL GNU Lesser General Public License – varianta licence pro svobodný software GNU GPL, která ale na rozdíl od GPL umožňuje spojování s nesvobodným kódem.

MPL Mozilla Public License – licence pro svobodný software používána pro zdrojový kód vydávaný Mozilla Corporation.

MSIE(IE) Microsoft Internet Explorer (Internet Explorer) – proprietární grafický webový prohlížeč společnosti Microsoft.

OS(S) Open Source (Software) – počítačový software s otevřeným zdrojovým kódem.

PDF Portable Document Format – souborový formát vyvinutý firmou Adobe pro ukládání dokumentů nezávisle na softwaru i hardwaru, na kterém byly pořízeny.

PHP Rekurzivní zkratka PHP: Hypertext Preprocessor (původně Personal Home Page) – skriptovací programovací jazyk určený především pro programování dynamických internetových stránek.

TDD Test-Driven Development – programování řízené testy, což je jedna z metod programování.

W3C World Wide Web Consortium – mezinárodní konsorcium jehož členové společně s veřejností vyvíjejí webové standardy pro World Wide Web.

WHATWG The Web Hypertext Application Technology Working Group – pracovní skupina snažící se o návrh nových technologií umožňujících autorům psát a nasazovat webové aplikace mnohem snadněji pomocí rozšíření existujících technologií.

WML Wireless Markup Language – značkovací jazyk založený na jazyce XML umožňující tvorbu online dokumentů pro mobilní zařízení.

WWW World Wide Web – označení pro aplikace internetového protokolu HTTP.

WYSIWYG What You See Is What You Get – označuje způsob editace dokumentů v počítači, při kterém je verze zobrazená na obrazovce vzhledově totožná s výslednou verzí dokumentu.

XHTML eXtensible HyperText Markup Language – značkovací jazyk pro tvorbu hypertextových dokumentů v prostředí WWW vyvinutý W3C.

XML eXtensible Markup Language – obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C.

Úvod

Je 21. století a neexistuje nikdo, kdo by dnes nebyl ať přímo či nepřímo závislý na jednom z nejrozsáhlejších informačních médií zvaném Internet. Tato celosvětová síť poskytuje snadnou a rychlou možnost, jak získávat a poskytovat informace, komunikovat s přáteli nebo organizacemi a to prakticky odkudkoliv na světě.

Jednou z nejoblíbenějších služeb Internetu je World Wide Web nebo také WWW či jednoduše web. Pro tuto službu se používají webové prohlížeče, které se připojují k serverům a získávají kód webových stránek, jenž následně svému uživateli interpretují do srozumitelnější nadefinované podoby. Pokud chce ovšem uživatel nějaké webové stránky vytvořit a přispět do celosvětové informační sítě, situace je pro něho komplikovanější.

Aby uživatel nemusel věnovat několik dní či týdnů studiu problematiky tvorby webových prezentací, je pro něho nezbytné mít kromě nástroje pro zobrazení webových stránek také nástroj pro jejich vytváření. V ideálním případě by tento nástroj měl být součástí webového prohlížeče, aby nebylo potřeba doinstalovávat dodatečný software. Zároveň by také měl mít pro uživatele snadné ovládání a mělo by být hned vidět, jak bude ve výsledku výtvor vypadat.

Bakalářská práce se skládá ze čtyř částí. Zprvu jsou popsány technologie, metody a nástroje použité při práci na editoru. V druhé části jsou vybrány některé existující a používané JavaScriptové WYSIWYG editory a vysvětlen princip na jakém takovéto editory fungují. Třetí a čtvrtá část je věnována praktické stránce bakalářské práce, vývoji a otestování vlastního JavaScriptového WYSIWYG editoru.

1 Technologie, metody a nástroje

1.1 WYSIWYG editor

Zkratka WYSIWYG je akronym anglické věty „What you see is what you get“, neboli v češtině „Co vidíš, to dostaneš“.

Tímto akronymem se označuje metoda pro práci s dokumenty na počítači, která umožňuje uživateli snadno upravovat dokument v podobě, v jaké bude vypadat výsledek interpretovaný na výstupu z daného programu. Například u nejčastějších WYSIWYG editorů, textových procesorů, se obvykle jedná o výslednou podobu stránky, která bude vytištěna na tiskárně. Příkladem takového editoru by mohl být velmi populární otevřený open source projekt OpenOffice.org, který je jednou z hlavních konkurencí proprietárního produktu Microsoft Word, silně svázaného s monopolními operačními systémy MS Windows.

1.1.1 WYSIWYG při tvorbě webu

WYSIWYG editor pro sazbu validního XHTML[1] tedy uživateli umožní známým jednoduchým a intuitivním způsobem vytvořit webovou stránku ve značkovacím jazyce XHTML.

Mnoho dokumentů vytvořených pomocí WYSIWYG editorů často trpí nepřiměřenou velikostí a nekompatibilitou s některými méně používanými prohlížeči. Takto generovaný kód je sice možné posléze editovat a opravit ručně, ale poté editor ztrácí svou největší výhodu.

Problémy s nepřiměřenou velikostí a nekompatibilitou je způsoben návrhem webu. Původně web nebyl navržen jako vizuální medium, a proto při rozvoji Internetu byly učiněny různé pokusy, jak dát autorům více kontroly nad vzhledem webové stránky, například pomocí CSS[2]. Bohužel se každý webový prohlížeč pokoušel vyřešit problém po svém a mnohé věci, které perfektně fungovaly v jednom prohlížeči, byly jen slabě podporovány v prohlížeči jiném. Důsledkem toho musely být WYSIWYG editory optimalizovány pro několik různých prohlížečů, řešily mnoho problémů a byly velmi komplikované.

[1] Značkovací jazyk pro tvorbu hypertextových dokumentů.

[2] Jazyk pro popis způsobu zobrazení stránek napsaných v jazycích HTML, XHTML nebo XML.

1.1.2 Výhody a nevýhody

Výhodou těchto editorů pro samotného uživatele je fakt, že uživatel nemusí vědět, jak vypadá samotný zdrojový text dokumentu a jaký jazyk pro popis dat je v dokumentu používán. Proto je pro většinu uživatelů, kteří mají zájem o vytvoření nějakého dokumentu, tento přístup tvorby dokumentu z hlediska časové náročnosti a na pochopení nejvhodnější.

WYSIWYG editory ovšem nejsou všemi uctívány a mají mnoho odpůrců, zejména mezi odbornější veřejností. Důvod je prostý, tyto editory jsou zaměřeny na vizuální ovládání, jednoduchost a snadnost pochopení. V některých případech, když člověk potřebuje kvalitní sazbu dokumentu nebo využít veškeré možnosti pro určitý formát dokumentu, WYSIWYG editory nestačí. Čím složitější a náročnější projekt, tím více ovládacích prvků je potřeba, a čím více ovládacích prvků nashromážděných ve hlavních menu a podmenu, tím se práce stává složitější, nepohodlnou a náročnou. WYSIWYG editory také nejsou schopny zpracovávat kvanta dat pro ideální vykreslení a rozmístění textu v reálném čase.

1.1.3 Alternativa TEX

Přestože WYSIWYG editory často usnadňují a urychlují práci s psaním textu, mnoho profesionálů stále používá alternativní způsoby pro vytváření textových dokumentů.

Jedna z nejkvalitnějších a nejpropracovanějších alternativ je typografický systém TEX naprogramovaný Donaldem Knuthem, který byl zhruba před třiceti lety nespokojen s nabízenou nekvalitní sazbou jeho matematické knihy. TeX se prosadil zejména pro psaní vědeckých, zvláště matematických, textů a stal se standardním formátem pro mnoho známých vědeckých časopisů.

Důvodů, proč psát v TeXu je mnoho. V jednom z mnoha článků o TeXu na webové stránce www.root.cz je vypsán seznam několika základních důvodů [9].

Proč tedy psát v TeXu?

- Skvělý typografický výstup.
- Vaše dokumenty půjdou kdykoliv přečíst, nemůže se stát, že se jejich obsah poškodí a nepůjdou otevřít (což se velmi často stávalo se staršími verzemi MS Office).
- Vede uživatele k dobrým návykům strukturování dokumentu.
- Můžete použít svůj oblíbený editor.

- TeX je multiplatformní, existují implementace pro MS Windows, Linux, Mac OS X, BSD systémy a další operační systémy.
- Existuje pro něj spousta šablon, ale také dokumentace.
- TeX je úžasně přizpůsobitelný a programovatelný, nebude vám bránit v rozletu, až se stanete pokročilým uživatelem.

Na rozdíl od WYSIWYG editorů, které zpracovávají a zobrazují text v reálném čase, se TeX kompiluje. Po napsání dokumentů je použit program pro převod textu s TeXovskými příkazy do požadovaného formátu, například přenosný formát dokumentů PDF[3]. To dává TeXu prostor pro mnohem kvalitnější a propracovanější aplikaci různorodých pravidel. Zvládá obrovské množství ligatur a dalších typografických technik, které se aplikují třeba pro zarovnávání do bloku textu, kdy u WYSIWYG editorů je výsledek mnohdy nepoužitelný. TeX zarovnává do bloku defaultně a určuje nejoptimálnější rozložení textu v odstavci a případně sám rozdělí slovo na konci řádku.

TeXové soubory se ukládají do obyčejných textových souborů, které jsou převážně tvořeny textem ve znakové sadě ASCII, Unicode, nebo ISO-8859-2. Což je velká výhoda pro vývojáře, kteří mohou velmi snadno zpracovávat tyto soubory programem či skriptem a vzhledem k tomu, že TeX není interaktivní, je možno jej používat v programu pro generování PDF souborů.

TeX je velmi dobře rozšiřitelný pomocí mnoha knihoven a také je pro něj k dispozici mnoho šablon a stylů.

TeX sám o sobě je velmi nízkoúrovňový, proto Donald Knuth zahrnul do svých plánů makrobalíky, které umožňují psát dokument a nezabývat se problémy s implementací obsahu, rejstříků a podobně. Tato makra jsou vlastně další příkazy TeXu.

Jedním z nejpoužívanějších balíků, který je používán i při psaní této bakalářské práci, je LaTeX.

1.2 XHTML – hypertextový značkovací jazyk

XHTML, neboli „eXtensible HyperText Markup Language“, což je česky přeložitelné jako „rozšiřitelný hypertextový značkovací jazyk“, je značkovací jazyk pro tvorbu hyper-

[3] Souborový formát vyvinutý firmou Adobe pro ukládání dokumentů nezávisle na softwaru i hardwaru, na kterém byly pořízeny.

textových dokumentů v prostředí WWW vyvinutý konsorciem W3C (World Wide Web Consortium).

Tento značkovací jazyk je následovníkem jazyka HTML (HyperText Markup Language), který byl navržen v roce 1990 společně s protokolem HTTP (HyperText Transfer Protocol – přenosový protokol hypertextu) pro jeho přenos po počítačových sítích. V té době byl také napsán první webový prohlížeč, který se jmenoval WorldWideWeb. (Zdrojový kód programu byl v roce 1993 uvolněn jako volné dílo, což z něj dělá svobodný software.)

Vývoj jazyka HTML byl ukončen ve verzi 4.01, které bylo vydané 24. prosince 1999 a měla být následně nahrazena verzemi XHTML založeném na jazyce XML[4].

Tento záměr konsorcia W3C se však nesetkal s dostatečnou kladnou odezvou a negativní hlasy se projevili v nedostatečném přijetí XHTML. Byla založena iniciativa WHATWG, která si dala za cíl vytvořit novou verzi HTML, označovanou jako HTML 5.

V reakci na to, 7. března 2007, byla konsorciem W3C založena pracovní skupina HTML, která má za cíl v roce 2010 uvolnit novou verzi HTML, označovanou jako HTML 5 a tak navázat na ukončený vývoj HTML 4.01. V květnu 2007 se poté W3C dohodla s WHATWG o převzetí doposud vytvořených nových návrhů Web Applications 1.0 a Web Forms 2.0.

Konsorcium W3C tedy momentálně paralelně vyvíjí dva směry a to značkovací jazyk HTML 5 a XHTML 2.

XHTML ve verzi 1.0 je první specifikace konsorcia W3C, která si klade za cíl převést starší značkovací jazyk HTML do podoby, která bude vyhovovat podmínkám vytváření XML dokumentům při zachování zpětné kompatibility. Tato verze existuje ve třech variantách.

Tou nejstriktnější a nejpřísnější verzí je XHTML 1.0 Strict. Hlavní důraz je kladen na osvobození dokumentů od formátovacích značek pro rozvržení stránky. Pro potřeby grafických efektů a rozvržení stránky se předpokládá použití kaskádových stylů označovaných jako CSS.

Pro snadnější přechod z HTML existuje verze XHTML 1.0 Transitional, která umožňuje používat překonané tagy a formátovací prvky zakázané ve striktní verzi. Tato verze je vhodná pro méně násilný přechod na XHTML standardy.

Poslední (třetí) verze je XHTML 1.0 Frameset. Kromě menší striktnosti (jako u Tran-

[4]Obecný rozšiřitelný značkovací jazyk.

sitional verze) přidává podporu rámců. Rámce jsou již v dnešní době z mnohých důvodů zavrhovány a nahrazovány novějšími metodami pro rozvržení obsahu na stránce.

Pro rozlišení verze XHTML či HTML dokumentu se používá definice typu dokumentu DOCTYPE na začátku dokumentu, který obsahuje informaci o použitém značkovacím jazyce a odkaz na soubor jeho pravidel.

Hlavním rozdílem zápisu HTML a XHTML je nutnost, aby všechny tagy byly v jazyce XHTML ukončené. U nepárových tagů jsou definovány tři způsoby ukončení, z nichž se převážně z důvodu zpětné kompatibility doporučuje zápis mezery s lomítkem před koncové znaménko „větší než“. Například zápis pro nový řádek v HTML se zapisuje jako
 a v XHTML jako
.

Dalším zásadním pravidlem je nutnost zápisu všech tagů a atributů malými písmeny, jelikož je jazyk XHTML citlivý na velikost písma a musí se dodržovat zápis v deklaraci DTD[5].

V XHTML je také nutno dodržovat správnou syntaxi a zapisovat atributy tagů do uvozovek. Každý atribut také musí mít definovanou hodnotu, pokud je uveden. Kromě těchto zásadních rozdílů jsou ještě další doporučení a méně častá pravidla.

1.3 CSS – kaskádové styly dokumentu

Zkratka CSS z anglického Cascading Style Sheets, česky tabulky kaskádových stylů nebo pouze kaskádové styly, je používána pro označení jazyku pro popis způsobu zobrazování dokumentů HTML, XHTML či XML. Tento jazyk opět navrhlo a dále vyvíjí konsorcium W3C. V současnosti jsou používány ve verzi 1 a 2, případně 2.1.

Cílem a hlavním důvodem existence kaskádových stylů je úplné oddělení obsahu a struktury dokumentu od jeho způsobu prezentace. Oddělenost definice vzhledu od obsahu a struktury umožňuje jednodušší upravování obsahu bez rizika poškození vzhledu dokumentu, možnost spolupráce více lidí na jednom projektu, kdy se jeden zabývá čistě obsahovou stránkou a druhý jen grafickou.

Jednou z velkých výhod, hlavně pro uživatele, je možnost nezávisle na obsahu kompletně změnit celkový vzhled dokumentu a tak může autor stránek dát uživateli na výběr mezi několika vzhledy nebo určit různá zobrazení pro různá výstupní zařízení. Například vzhled pro zobrazení na monitoru nemusí být ideální pro tištěnou verzi apod.

[5]Definice typu dokumentu pro popis struktury XML (případně SGML) dokumentu.

Pro tvůrce dokumentu CSS přináší především rozsáhlejší možnosti úprav vzhledu, dynamickou práci se vzhledem a konzistentní vzhled všech dokumentů využívajících stejný kaskádový styl.

1.4 JavaScript – klientský programovací jazyk

JavaScript vytvořený Brendanem Eichem z tehdejší společnosti Netscape je objektově orientovaný skriptovací jazyk používaný tvůrci webových stránek nebo aplikací za účelem zvýšení interakce s uživatelem především na klientském stroji.

Svůj název získal tento jazyk především z marketingových důvodů. To se vyplatilo a mnoho lidí si plete programovací jazyk Java se skriptovacím jazykem JavaScript, díky čemuž se JavaScript rychle dostal do povědomí internetové komunity a postupně se propracoval prakticky na všechny webové prohlížeče. Kromě názvu spojuje JavaScript a Javu, ale také jazyky C, C++ podobná syntaxe.

V roce 1997 byl JavaScript standardizován asociací ECMA (Europen Computer Manufacturers Association) a o rok na to ISO (International Organization for Standardisation). Standardizovaná verze JavaScriptu byla pojmenována ECMAScript.

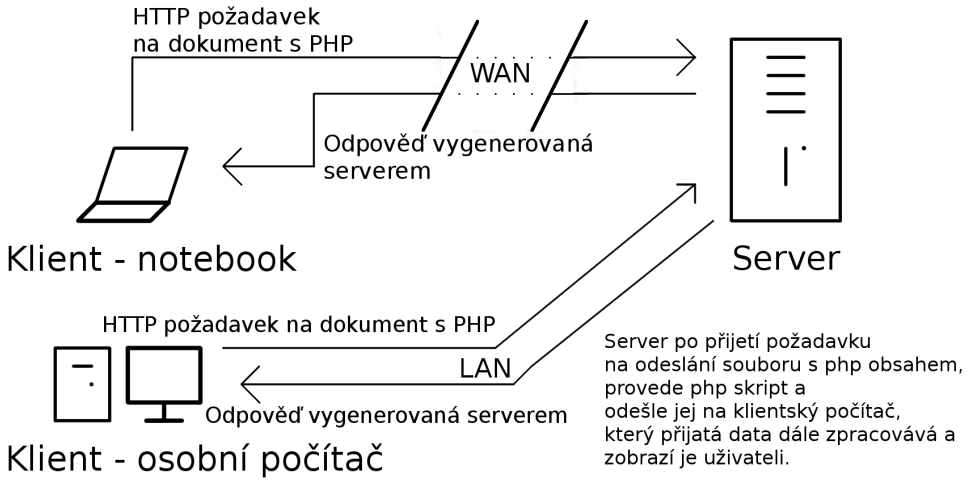
Vzhledem k tomu, že je tento skriptovací jazyk spouštěn převážně až na straně klienta při načítání WWW stránky do prohlížeče, musí být respektována určitá bezpečnostní omezení. Především se jedná o nemožnost pracovat se soubory na klientském stroji, aby se zabránilo nežádoucímu upravování, čtení citlivých souborů (ať už osobních nebo systémových) a nedošlo tak k narušení soukromí uživatele nebo nebyla narušena bezpečnost uživatelova počítače.

Kromě základní syntaxe podobné klasickým programovacím jazykům má JavaScript speciální rozhraní nazývané DOM, což je akronym anglického Document Object Model, česky objektový model dokumentu, které umožňuje přistupovat, modifikovat nebo vytvářet obsah, strukturu či styl dokumentu. Naneštěstí se právě zde nejvíce projevuje jedna z velkých problémů tvorby webových stránek a tou je kompatibilita mezi prohlížeči. Ze začátku měl každý webový prohlížeč vlastní speciální rozhraní pro manipulaci s HTML elementy v JavaScriptu, které byly navzájem nekompatibilní. Nebohý webmaster, tvůrce webových stránek, proto musel vždy ověřovat různými kličkami verzi a typ prohlížeče, aby mohl pro daný prohlížeč spustit speciální kód (pro něho optimalizovaný). Proto se konsorcium W3C odhodlalo ke standardizaci tohoto rozhraní a vznikla specifikace

W3C DOM, která je platformě i jazykově nezávislá.

Specifikace W3C DOM je pro snadnější implementaci do webových prohlížečů rozdělena do několika úrovní označovaných jako DOM. Dále zároveň ještě každá DOM vrstva obsahuje povinné a volitelné moduly. Pro splnění určité úrovně DOM je zapotřebí splnit úrovně předešlé a zároveň tu, která má být splněna a tak bude moci být aplikace označena, že podporuje určitý DOM level. V době psaní této práce existují tři stupně a to level 1, level 2 a level 3.

1.5 PHP – serverový programovací jazyk



Obrázek 1: Zaslání požadavku na PHP stránku klienta na server

PHP je rekurzivní zkratka PHP: Hypertext Preprocessor, původně Personal Home Page. Označuje skriptovací programovací jazyk, určený především pro programování dynamických webových stránek. Začleňuje se přímo do struktury jazyků HTML, XHTML či WML[6], což je velmi výhodné pro tvorbu internetových aplikací. Skripty napsané v PHP jsou prováděny na straně serveru a klient nemá povědomí, jestli byl při vytváření dokumentu tento jazyk použit či nikoliv. Uživatel na klientské straně dostane pouze výsledek činnosti PHP skriptů. Syntaxe jazyka PHP je jako v případě JavaScriptu opět podobná jazykům C a Java. Jazyk je nezávislý na platformě a umožňuje používat rozsáhlé knihovny funkcí pro zpracování textu, grafiky, práci s databází a soubory.

[6]Značkovací jazyk založený na jazyce XML zaměřený na tvorbu dokumentů pro mobilní zařízení.

1.6 AJAX – nový přístup pro webové aplikace

AJAX neboli „Asynchronous JavaScript and XML“ je novou ikonou na poli webového inženýrství. Avšak není to žádný samostatný jazyk nebo nová technologie. AJAX je spíše nový přístup, se kterým přišel jako první v článku AJAX: A New Approach to Web Applications neboli „AJAX: Nový přístup k webovým aplikacím“ v dubnu 2005 Jesse James Garretta.

Prapůvod AJAXu sahá do roku 1998, kdy společnost Microsoft představila novou technologii nazvanou Remote Scripting, ve které v klientském prohlížeči běžel Java applet[7] komunikující se serverem, přičemž tento applet poskytoval služby JavaScriptovým funkcím. Tato technika spolupráce serveru s klientským prohlížečem začala fungovat v MSIE od verze 4 a také v Netscape Navigatoru od verze 4. Microsoft ve svojí páté verzi prohlížeče Internet Explorer představil objekt XMLHttpRequest, který v roce 2000 využil v novém programu Outlook Web Access, který poskytuje webové rozhraní pro přístup k emailům na Microsoft Exchange Server. Objekt XMLHttpRequest je základním stavebním kamenem AJAXu.

Největší zásluhu na zpopularizování a rozšíření AJAXu lze připsat společnosti Google, která vlastní jeden z nejznámějších a nejpoužívanějších vyhledávačů a která AJAX začala využívat ve svých produktech. Zprvu u online emailové služby Gmail, posléze tomu bylo u celosvětových map se službou Google Maps a plno dalších.

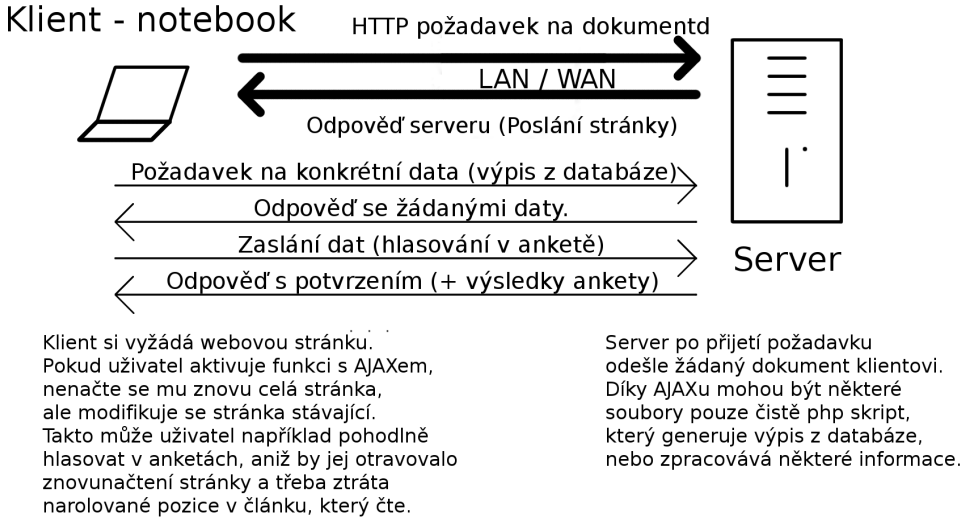
Jednou z hlavních výhod a důsledkem, proč se AJAX tak rychle začal rozvíjet a používat, je odstranění nutnosti znovu načtení a překreslení celé webové stránky při každé akci nebo operacích souvisejících s prohlížením a používáním webových stránek. Na rozdíl od klasického modelu WWW stránek se tedy stránky využívající AJAX nemusí celé znovu načítat.

AJAX totiž umožňuje odeslání informací na server, odkud získá potřebnou odpověď. Následně je pomocí JavaScriptu modifikován obsah zobrazovaných stránek. Uživatel tak není nucen čekat na opětovné načítání stránky a celá operace zobrazení žádaných informací se tak zrychlí. Odesílání a přijímání dat probíhá na pozadí. Uživatel má pocit mnohem větší plynulosti práce, která se dokonce blíží běžným desktopovým aplikacím.

Díky tomuto vylepšení a změně přístupu se snižuje zátěž na webové servery a celou síť mezi klienty a serverem, přes které se museli posílat mnohem větší kvanta dat. AJAX však

[7] Softwarová komponenta, která běží v kontextu jiného programu.

může zvýšit počet vyměňovaných HTTP požadavků, třebaže přenášejí nižší množství dat, proto při nevhodné implementaci zátěž neklesne.



Obrázek 2: Znázornění posílání požadavků klienta na server pomocí AJAXu

Mezi hlavní nevýhody patří změny v paradigmatu používání webu. Tedy změna chování webové stránky. Web se chová jako plnohodnotná aplikace se složitou vnitřní logikou a nikoliv jako posloupnost stránek, mezi kterými se lze navigovat. Největší vliv je vidět při použití tlačítek „Zpět“ a „Další“. Moderní AJAXové aplikace jsou však schopny tyto funkce tlačítek upravit a nasimulovat tak kompatibilní funkčnost se standardním modelem webových stránek.

1.7 TDD – programování řízené testy

TDD je zkratkou z anglického „Test-Driven Development“, tedy „programování řízené testy“. Tato technika vývoje řízeného testy je součástí agilní metody vývoje softwaru nazývané jako extrémní programování.

Hlavním rozdílem oproti klasickému cyklu vývoje softwaru, kdy se stanoví požadavky, návrh architektury, poté se přistoupí ke kódování a následně k testování, je vytváření testů ještě před samotným kódováním.

Klasický přístup vývoje aplikace předpokládá existenci nějakého kódů programů, který se následně testuje a kontroluje. Použité testy kontrolují, jestli se projekt neodchýlil

od zadání a pokud ano, musí se chyba napravit. Nalezení a opravení chyb v pozdějších fázích vývoje, kdy je projekt mnohem rozsáhlejší, může způsobit nemalé problémy.

Oproti tomu TDD vytváří testy před samotným napsáním kódu, který by testy prošel. Testy v TDD nemají za cíl kontrolovat kód, ale spíše představují nástroj pro návrh systému.

Na začátku programování se napíše test, který specifikuje úkol dané části programu. Test musí selhat nebo vůbec neproběhnout, protože zatím nebyl napsán žádný kód. Následně se naprogramuje logika programu, tak aby test alespoň proběhl. Postupnými menšími úpravami kódu a spouštěním testů dojde k upravení kódu do přijatelné podoby, kdy testy projdou. Pokud ještě není program hotový, napíše se další test a pokračuje se v programování. Pokud jsou všechny testy úspěšné a již není potřeba vytvářet další testy, program je hotový, funkční a nemusí se již testovat.

Nevýhodou je, že programování řízeného testu se dá uplatnit jen pro části kódu nezabývající se grafikou a je tedy vhodné především pro programování aplikací či částí kódu přinášející nějaký negrafický výsledek.

1.8 Webové prohlížeče

Webové prohlížeče, též nazývané jako browsery jsou počítačové programy, sídlící na klientském stroji. Když se uživatel rozhodne prohlížet World Wide Webu nebo nějaký dokument v podporovaném formátu jako například HTML, XHTML, XML, spustí jeden z těchto programů a on mu podle svých možností a podporovaných standardů zformátuje a zobrazí dokument.

Prohlížeče umožňují komunikaci se servery sídlícími na Internetu pomocí protokolu HTTP, díky němuž webový prohlížeč přistupuje ke vzdáleným dokumentům, třeba na druhém konci světa.

Mohli bychom je rozdělit do dvou skupin a to na prohlížeče textové a prohlížeče grafické. Textové prohlížeče zobrazují dokumenty jako text a to obvykle s velmi jednoduchým formátováním. Takovými programy jsou například Links nebo Lynx. Na rozdíl od textových umožňují grafické prohlížeče složitější formátování stránky včetně zobrazení obrázků. Lze v nich zobrazit některé externí součástí stránky jako jsou Flashové animace nebo Javové applety, pro které je potřeba do prohlížeče doplnit specializované zásuvné moduly. Mezi nejznámější grafické webové prohlížeče patří Internet Explorer, Mozilla Firefox, Opera, SeaMonkey, Konqueror a Safari.

Jednou z nejdůležitějších částí webových prohlížečů je jádro, také označované jako renderovací jádro. Jeho účel v prohlížeči je vykreslovat webové stránky a je napsáno ve vyšších programovacích jazycích, jako je například jazyk C++. V současnosti nejpoužívanější jádra jsou Trident, Gecko a KHTML.

1.8.1 Jádro Trident

Na jádře Trident, dříve známý pod názvem MSHTML, jsou postaveny prohlížeče Microsoft Internet Explorer, které jsou dodávány společně s operačním systémem Microsoft Windows. Z tohoto důvodu je také toto jádro jedno z nejpoužívanějších na celém světě. Poprvé se objevilo roku 1997 v Internet Exploreru 4.0 a v současnosti je ve verzi 5 součástí Internet Exploreru 7.0.

Dále se toto jádro nachází v prohlížeči Maxthon, který se stal velmi oblíbenou alternativou k Internet Exloreru 6. Maxthon si našel přízeň mezi uživateli především pro své dodatečné vlastnosti jako jsou záložky, ale hlavně pro své lepší zabezpečení než bylo u Internet Exploreru 6.

Ještě bych zmínil prohlížeč Netscape Browser 8.0, který je výjimečný a zajímavý především pro svou schopnost přepínat mezi renderovacím jádrem Gecko, které používají prohlížeče postavené na Mozille a jádrem Trident.

Jako většina produktů společnosti Microsoft je i jádro Trident proprietární nesvobodný software a proto jeho kódy jsou uzavřeny. Známé a pro mnohé vývojáře nemilé vlastnosti Tridentu jsou slabší podpora webových standardů, existence některých nestandardních rozšíření a především u starších verzí slabší zabezpečení.

Kromě webových prohlížečů se jádro Trident, díky své integraci do operačního systému Windows, používá jako renderovací jádro v mnoha dalších komponentách systému. Například je využito pro nápovědu v systému Windows, kancelářský balík Microsoft Office a Windows Explorer. Může jej využívat a také jej využívá řada instalovaných aplikací nepocházejících od společnosti Microsoft.

Aktuálně je vyvíjen Internet Explorer 8, který by měl být vydán ve stabilní verzi koncem roku 2009 a měl by být kladen větší důraz na bezpečnost a webové standardy.

1.8.2 Jádro Gecko

Gecko je druhé nejpoužívanější jádro hned po jádře Trident a jeho obliba neustále roste i díky tomu, že na rozdíl od jádra Trident se Gecko řadí mezi open source projekty a je používáno v mnoha dalších open source projektech. Bylo napsáno v programovacím jazyce C++, pod licencemi MPL, GPL a LGPL. Toto jádro bylo původně vytvořeno firmou Netscape Communications Corporation, ale nyní je vyvíjeno společností Mozilla Corporation.

Gecko nenabízí pouze možnost renderování webových stránek, ale díky svému bohatému API slouží i k vykreslování grafického rozhraní (XUL), využívaného Firefoxem či Thunderbirdem. Není zaměřeno pouze na určitý systém, ale jedná se o multiplatformní jádro. K dispozici je pro Microsoft Windows, Linux, Mac OS X a další.

Z důvodů zpětné kompatibility a podpory webových stránek optimalizované pro staré verze prohlížečů Netscape Navigator a Internet Explorer podporuje Gecko dva režimy vykreslování. Pokud je v dokumentu nalezena definice Doctype, vykresluje se stránka „v režimu platných standardů“. V opačném případě využije tzv. „režim zpětné kompatibility“, kterému se také říká quirks mód.

Mezi významnější programy využívající jádro gecko patří prohlížeče Epiphany, Galeon, prohlížeč Camino určen pro Mac OS, multimediální přehrávač Songbird, poštovní klient Mozilla Thunderbird nebo balík internetovéch aplikací SeaMonkey.

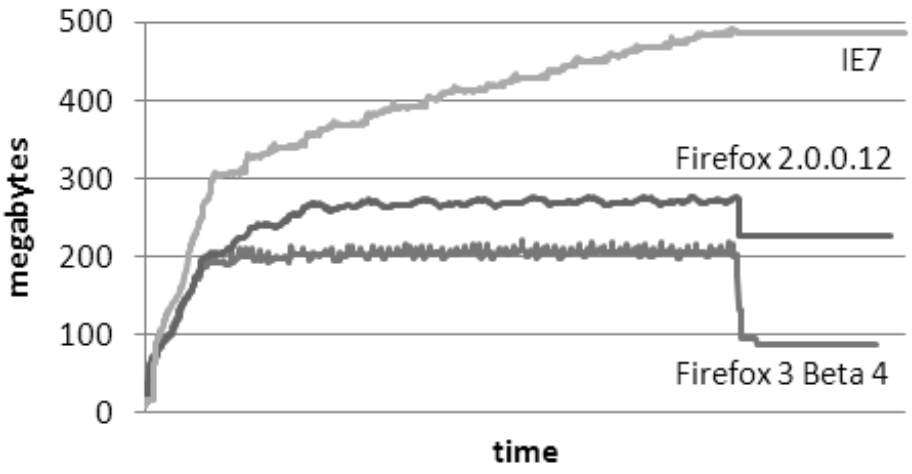
Aktuální verzí je Gecko 1.9. Toto jádro je obsažené v prohlížeči Mozilla Firefox 3.0, která by měla být vydán oproti původnímu plánu v červnu. Vývojáři Firefoxu se v poslední verzi jádra zaměřili především na snížení paměťové náročnosti prohlížeče, což se jim mělo povést oproti starším verzím o jednu třetinu.[8]

1.8.3 Jádro Presto

Renderovací jádro Presto se poprvé objevilo 28. ledna 2003 ve webovém prohlížeči Opera 7.0 od firmy Opera Software, pro operační systém Windows a nahradilo tak dosavadní starší jádro Elektra. Je licencováno obchodním partnerem Adobe Systems a také integrováno do produktu Adobe Creative Suite. Opera je distribuována jako freeware.

Největší předností tohoto jádra a tedy také Opery je snaha důsledně dodržovat standardy konsorcia W3C. Spolu s dodržováním standardů, dobrým zabezpečením, rychlostí

[8] zdroj: http://blog.pavlov.net/2008/03/11/firefox-3-memory-usage/



Obrázek 3: Snížení paměťové náročnosti poslední verze Firefox 3

a technologií Small Screen Rendering se prohlížeč Opera (ve speciálně upravené podobě) dostal na vedoucí pozici jako prohlížeč pro „chytré“ mobilní telefony a PDA.

1.8.4 Jádro KHTML a WebKit

KHTML je renderovací jádro, které je vyvíjené v rámci projektu KDE, což je desktopové prostředí pro operační systém Linux. Napsáno je v jazyce C++. Uvolněno je pod licencí LGPL a v KDE se nachází od verze 2.

Využívá jej webový prohlížeč Konqueror, což je hlavní prohlížeč používaný v desktopovém prostředí KDE a je na něm založen také prohlížeč Swift a RSS čtečka Akregator.

Na základě jádra KHTML bylo částečně vytvořeno společností Apple jádro WebKit, konkrétně renderovací engine WebCore, a je použito pro webový prohlížeč Safari určený především pro operační systémy Mac OS X a Apple iPhone. Jako KHTML je napsán v jazyce C++ a používá licenci LGPL.

2 WYSIWYG XHTML editory

Myšlenka na vytvoření WYSIWYG editoru, který by byl součástí webových prohlížečů není nová. Ve skutečnosti byl WYSIWYG HTML editor součástí již v prvním internetovém prohlížeči umožňujícím HTTP, který se jmenoval WorldWideWeb (později Nexus).

2.1 Současné open source editory

Dnešní situace je o něco složitější. Současné prohlížeče se staly postupem času mnohem rozsáhlejší, složitější a používanější. Z tohoto důvodu jsou i dnešní nároky na WYSIWYG editory větší.

Základní podmínkou u níže popsaných editorů je GPL nebo LGPL licence, tedy aby WYSIWYG editor byl volně šiřitelný a libovolně modifikovatelný.

Vzhledem k existenci několika používaných prohlížečů je jedním z důležitých faktorů, aby byl WYSIWYG editor kompatibilní a choval se v každém editoru stejně. To je vzhledem k faktu, že prohlížeče nejsou mezi sebou kompatibilní, náročný úkol.

Pro webmastery a webové vývojáře aplikací je důležité, aby se kód aplikace dal snadno modifikovat dle jejich přání, vylepšovat o specifické záležitosti a v neposlední řadě byl snadno integrovatelný.

Dnešní open source JavaScriptové WYSIWYG editory jsou velmi oblíbené díky své dobré funkcionalitě a nulovým pořizovacím nákladům. Ty nejlepší projekty dosahují kvalit komerčních projektů jako EditOnPro[1].

2.1.1 TinyMCE

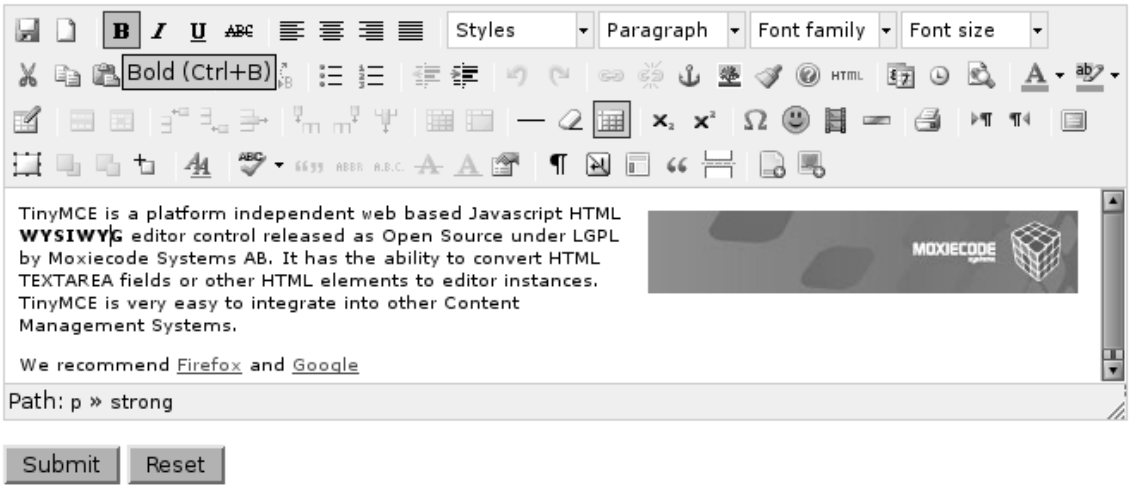
Jedním z nejpopulárnějších a nejpropracovanějších WYSIWYG XHTML editorů psaných v JavaScriptu je produkt TinyMCE vyvíjený společnosti Moxiecode systems.

Vzhledem k licenci LGPL je tento opensource projekt možno libovolně používat. Modifikace jsou řešeny pomocí pluginů a témat. K dispozici je i český překlad.

Je možné ho stáhnout na stránkách *http://tinymce.moxiecode.com/* v sekci download.

Integrace je velmi snadná. Po stažení souboru a rozbalení na místo, odkud bude aplikace volána stačí v cílovém dokumentu, kde má být editor použit, vložit externí JavaScriptový soubor s editorem a do hlavičky dokumentu vložit inicializační kód, ve kterém

[1] zdroj: *http://www.standards-schmandards.com/2007/WYSIWYG-editor-test-2/*



Obrázek 4: WYSIWYG editor TinyMCE

se provedou potřebná nastavení editoru. Nastavení inicializační části editoru již však tak snadné není, a pokud je potřeba vyřešit nějaké specifické nastavení, je třeba se seznámit s dokumentací a správně vše nakonfigurovat. Spolu s editorem se rozbalí i několik příkladů pro snadnější pochopení funkčnosti, a dobrá podpora je také na domovských stránkách.

Společnost Moxiecode kromě editoru nabízí i plugin pro práci se soubory MCFileManager a plugin pro práci s obrázky MCImageManager. Tyto pluginy však již nejsou dostupné zdarma a je nutné za ně zaplatit.

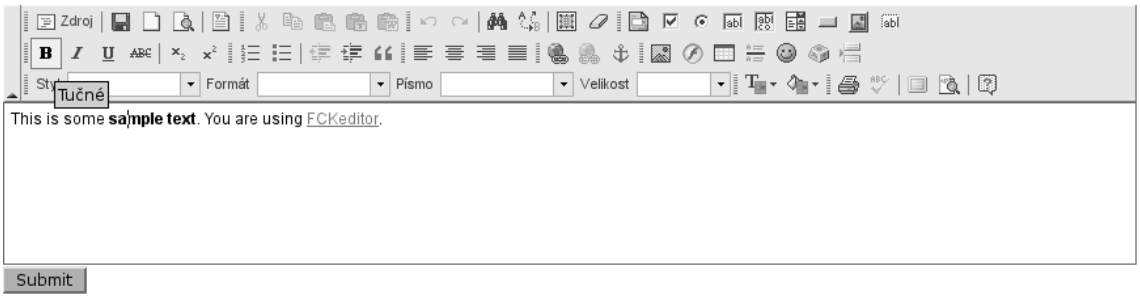
Mezi přednosti TinyMCE patří možnost uložení výsledných dokumentů pomocí technologie AJAX.

Nevýhodou je především placený souborový a obrázkový manager. Editor je velmi robustní, rozsáhlý a vnitřně komplikovaný z důvodu uspokojení uživatelů po co největším množství funkcí a možností, proto není vhodný pro méně náročné webové stránky nebo pro studium struktury, jak takový editor funguje.

2.1.2 FCKeditor

Jednou z velkých konkurencí populárního TinyMCE je editoru FCKeditor, který byl podle počítadla umístěném na stránce stáhnut více jak dva a půl milionkrát.

Tento editor nabízí rozsáhlejší řadu licencí a tak umožňuje vybrat si tu nejvhodnější. Editor je distribuován pod GPL, LGPL a MPL open source licencemi a navíc pod komerční licencí Closed Distribution License, která je určena pro společnosti a produkty, kterým



Obrázek 5: WYSIWYG editor FCKeditor

nevyhovují open source licence. Tato komerční licence je zpoplatněna.

Je možné ho stáhnout na stránkách *http://www.fckeditor.net/* v sekci download.

Integrace editoru do webových stránek je o něco obtížnější, než u editoru TinyMCE, což může mít souvislost s možností placené podpory. Po stažení a rozbalení souboru jsou v adresáři složky editor a samples s několika soubory. Pomocí příkladů však lze snadno odvodit, jak daný editor zakomponovat do stránek.

Na rozdíl od TinyMCE má FCKeditor pouze souborový manager, proto mnoho webových vývojářů, kteří potřebují do svého redakčního systému komplexnější řešení, zůstávají u TinyMCE i přes méně výhodné licenční podmínky a zpoplatněné pluginy.

Mezi přednosti tohoto editoru patří automatická detekce jazyka, nemusí se tedy speciálně stahovat a nastavovat jazyková mutace.

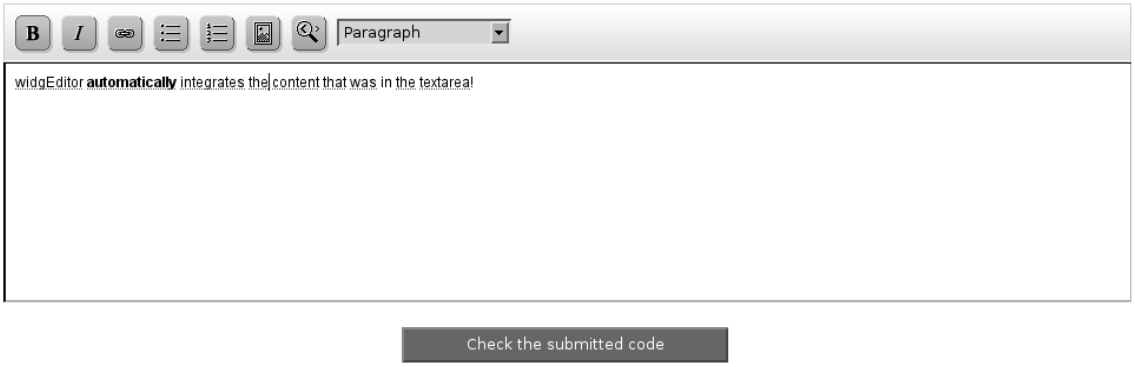
Specialitou je řešení nekompatibility prohlížečů Internet Explorer a Firefox. FCKeditor má pro každý prohlížeč zvláštní jádro.

2.1.3 widgEditor

Mezi robustními WYSIWYG editory se skrývá nenápadný, ale oblíbený editor widgEditor, který mnohé zaujal svou jednoduchostí, rychlostí a na rozdíl od TinyMCE nebo FCKeditoru svou datově mnohem menší velikostí.

Autor uvolnil editor pod licencí GPL a dále jej přestal vyvíjet. Díky licenci a přehlednosti kódu si jej oblíbilo mnoho webmasterů a vývojářů. Stoupající obliba widgEditoru donutila autora po třech letech k vydání nové verze s opravami chyb, které se v průběhu času objevili. Byl také přislíben vývoj nové verze editoru.

Základní vlastnosti editoru jsou oproti výše zmíněným editorům strohé. Editor umí používat tučný text, kurzívu, vkládat odkazy, obrázky, seznamy, odstavce, nadpisy a



Obrázek 6: WYSIWYG editor widgEditor

zobrazit zdrojový kód, což však mnoha lidem vyhovuje a díky jednoduchosti a přehlednosti je možné editor vcelku snadno modifikovat.

Je ho možné stáhnout na adrese *http://themaninblue.com/experiment/widgEditor/*.

Implementace do webových stránek je velmi jednoduchá. Po stažení a rozbalení souboru stačí v cílovém webovém dokumentu do hlavičky vložit externí JavaScriptový soubor s editorem a následně budou všechna textová pole <textarea> nahrazena WYSIWYG editorem. To sice vždy nemusí vyhovovat, ale na druhou stranu není potřeba žádná další nastavení.

Díky svojí přehlednosti a dobře čitelnému zdrojovému kódu je tento editor vhodný ke studijním účelům.

2.1.4 Bronziho editor

Poslední editor, o kterém bych se rád zmínil, je Bronziho editor, který je jeden z velmi nadějných a jeden z prvních českých projektů zaměřených na vytváření validního HTML a XHTML obsahu pro webové stránky pomocí JavaScriptového WYSIWYG editoru.

Záměrem autora je svobodné šíření a modifikace aplikace podle potřeb uživatele a proto je program uvolněn pod licencí GPL a LGPL.

Je možné ho stáhnout na stránkách *http://www.bronzi.cz/* spolu s dalšími projekty.

Podle autorova názoru se většina online WYSIWYG editorů používajících v redakčních systémech snaží o implementaci mnoha funkcí, které však nemají velkou využitelnost. Proto se snaží vyhnout situaci, kdy tvůrci webových stránek pracně omezují funkce editoru za účelem zabránění znehodnocení designu stránek nezkušenými uživateli, kteří by

Obrázek 7: WYSIWYG editor Bronzi

Obrázek 8: Vzhled Microsoft Word 2007

se pokoušeli vylepšit svůj příspěvek různými nevhodnými vylepšeními.

Aktivace editoru je velice snadná a dobře vysvětlená a znázorněná na webových stránkách autora. Pro stažení a rozbalení stačí do dokumentu, kde se má nacházet editor vložit odkaz na externí soubor se samotným skriptem, následně vložit jednoduchou konfiguraci a nakonec nastavit elementu <textarea>, který má být překonvertován na WYSIWYG editor, identifikátor „textarea“.

Grafické rozhraní bylo znatelně inspirováno nejnovějšími verzemi aplikace Microsoft Word[1] a proto bude pro mnoho uživatelů velkým přínosem jeho velmi líbivý a povědomý vzhled.

Autor dále také pracuje na vlastním souborovém a obrázkovém manažeru.

[1]Obrázek č. 8, zdroj: *http://myego.cz/item/microsoft-office-2007-o-deset-let-pred-konkurenci*

2.2 Princip JavaScriptových WYSIWYG editorů

Základním principem, na kterém jsou postaveny JavaScriptové WYSIWYG editory, je využití renderovacího jádra prohlížeče pro zobrazení editované nebo tvořené webové stránky a zároveň využití speciálních funkcí jádra pro možnost editace dokumentu.

Výsledkem je pak rozhraní, ve kterém je možno zároveň vidět náhled dokumentu a současně jej editovat. Díky tomu je náhled generován renderovacím jádrem prohlížeče, což zajišťuje pozdější shodné zobrazení stránky v normálním webovém režimu prohlížeče.

2.2.1 Začlenění editoru do stránek

Aby bylo možné editor použít pouze v určené oblasti vyhrazené editoru, je nutno editovanou oblast uzavřít do určitého elementu. Editaci pouze vybraného elementu je možné docílit s použítím atributu *contenteditable* vytvořeném společností Microsoft a zakomponovaném od prohlížeče Microsoft Internet Explorer 5.5. Tento atribut však není podporován jádrem Gecko. Neexistenci tohoto atributu však v tomto jádře nahrazuje vlastnost *designMode*, která nabízí obdobnou funkcionalitu jako atribut *contenteditable*.

Z tohoto důvodu musí všechny nejpoužívanější editory tento problém speciálně obcházet. Vytvoří se pomocná webová stránka s minimálním obsahem, u které se zapne editační mód. Následně v cílovém dokumentu s WYSIWYG editorem je použit element <frame>, <iframe> nebo <object> podporující vkládání externího dokumentu jako svůj obsah, čímž vznikne výsledný efekt vyhrazené editovatelné oblasti.

Možnost editování prostého textu však není nic výjimečného a tuto vlastnost umožňuje i tag <textarea> . Proto je nutno na klientské straně vytvořit určitou nadstavbu ovládacích prvků, které umožní modifikovat HTML kód.

2.2.2 Modifikace kódu dokumentu

Modifikace HTML kódu dokumentu se provádí pomocí funkce `execCommand`, která se volá na vzdáleném souboru se zapnutou možností editování. Jako příklad bych uvedl příkaz *bold*, který do dokumentu vloží značky pro tučný text, případně obklopí text označený.

Tato funkce má tři parametry, z toho první je povinný a předává se jím název příkazu, který se má provést.

Druhý nepovinný parametr nabývající booleovských hodnot „true“ nebo „false“ určuje možnost, zda má být uživatel vyzván k zadání dodatečných informací. Implicitní hodnota

tohoto parametru je „false“ a tato funkce zatím není prohlížeči podporována a nefunguje. S jejím využitím se počítá do budoucna.

Třetí a opět nepovinný parametr se používá spolu s některými příkazy, které vyžadují předání dodatečných informací ke správnému provedení daného příkazu. Například při vkládání odkazu je potřeba funkci informovat o adrese, kam má odkaz odkazovat, což by měl normálně zajišťovat druhý nepovinný parametr. Vzhledem k nefunkčnosti druhého parametru je nutné zjistit potřebné informace ručně. Možností je několik, od JavaScriptového příkazu `prompt`, který pomocí dialogového okna požádá uživatele o zadání určitých informací, po některý vstupní dialogový formulářový prvek. Takto získaná data se ještě musí ověřit. Pokud uživatel zadal korektní data, jsou předána jako třetí parametr funkce `execCommand`.

2.2.3 Validita kódu

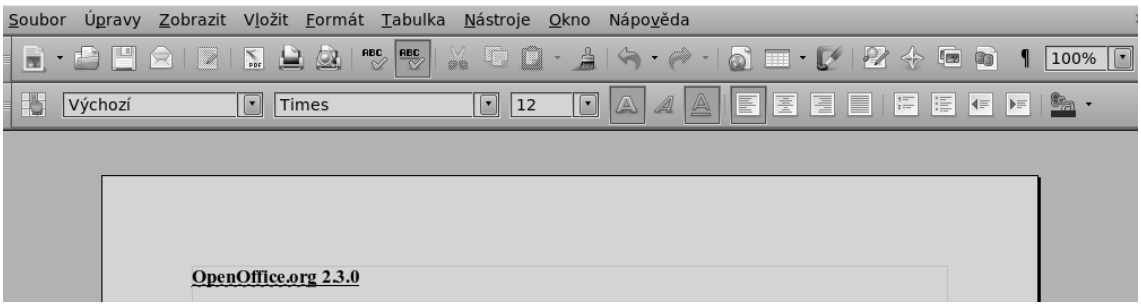
Bohužel každý webový prohlížeč má odlišnou implementaci funkce `execCommand`, kdy například Internet Explorer a Opera vkládá všechny značky velkými písmeny a Firefox používá řádkový tag span s vloženým stylem. Aby práce s funkcí `execCommand` nebyla až tak snadná, prohlížeče nepodporují stejné příkazy nebo jejich vykonávání neprovádějí úplně stejně. To znamená, že některé příkazy nelze použít, pokud chceme docílit jednotného rozhraní ve všech optimalizovaných prohlížečích. Přehledné porovnání kompatibility příkazů v jednotlivých prohlížečích poskytuje server *www.quirksmode.org* [2].

Funkce `execCommand` není jedinou překážkou při modifikaci zdrojového kódu dokumentu. Velkým problémem jsou samotné webové prohlížeče, které si při ručním vkládání kódu bez použití funkce `execCommand`, převádějí zdrojový kód do své vlastní vnitřní srozumitelné syntaxe. To je příčinou, proč se musí kód vytvářené webové stránky při přepnutí do textového módu nebo odesílání na server převalidovat. Důvod, proč webové prohlížeče přizpůsobují modifikovaný obsah, pramení ze snahy opravení chyb v kódu a snahou umožnit zobrazení i nevalidním a špatně napsaným dokumentům.

2.2.4 Grafické rozhraní

Z důvodů uživatelské přívětivosti a jednoduchosti ovládání je potřeba, aby webové WYSIWYG editory měly přívětivé grafické rozhraní. Vzhledem k prakticky monopolnímu

[2] *http://www.quirksmode.org/dom/execCommand.html*



Obrázek 9: Grafické ovládací rozhraní aplikace OpenOffice.org Writer 2.3.0

postavení komerčních produktů společnosti Microsoft se stalo jím používané grafické ovládání nepsanou podmínkou, pro veškeré grafické aplikace.

Příklad takového grafického rozhraní je možné vidět na obrázku č. 9 aplikace OpenOffice.org Writer ve verzi 2.3.0. Je zde patrné rozdělení na tři části.

První část je určena pro rozsáhlá menu a podmenu, která umožňují komplexní ovládání nebo přizpůsobení aplikace. Tato nabídka je však pro mnohé uživatele z počátku značně nepřehledná, a pokud uživatel nepotřebuje využít komplexnost celého programu, je pro něho zbytečné znát dopodrobna všechna její zákoutí. Z tohoto důvodu existuje panel nástrojů a hlavní menu uživatel prakticky nepotřebuje využívat. Obrátí se na něj jen pokud potřebuje přenastavit program nebo použít některou z pokročilejších funkcí, kterou mu nenabízí panel nástrojů.

Druhá část, které se říká panel nástrojů, se skládá z tématických podnabídek nástrojů pro rozličné činnosti. Každý nástroj má svou vlastní ikonku nebo nabídku a celý panel umožňuje základní práci s dokumentem, jeho načtení, uložení nebo tisk. Některá tlačítka jsou pouze dvoustavová a vztahují se k editovanému textu a jeho formátování. Ukazují, v jakém formátování se nachází označený text, případně text nacházející se pod aktivním kurzorem. Při použití těchto prvků se změní jejich stav i formátování textu. Díky své přívětivosti, názornosti a rychlé dosažitelnosti se nástroje hojně využívají, a proto je jejich použití ve WYSIWYG editorech nutností.

Ve třetí části, kterou zmíním, se nachází editační oblast, kde je možné sepisovat svůj dokument ve WYSIWYG módu bez nutnosti znalosti vnitřní kódové struktury dokumentu či syntaxe.

2.2.5 Ukládání výsledku

Kdyby nešlo v JavaScriptovém WYSIWYG editoru vytvořený dokument uložit, byl by takový editor zbytečný a nepoužitelný. Hlavním předpokladem pro uložení výsledku je podpora programovacího jazyka pro práci se soubory. JavaScript, který je používán u JavaScriptových WYSIWYG editorů však z praktických důvodů tuto funkcionalitu nemá vůbec integrovanou a nepodporuje ji.

Důvodem, proč JavaScript neumožňuje práci se soubory je především bezpečnost pro uživatele, jeho osobní data a systém. Jelikož se JavaScript provádí na klientově počítači, ale skripty se načítají ze vzdáleného serveru, někdy i nebezpečného a infikovaného viry, není možné zaručit nezávadnost načítaného skriptu. I kdyby tato funkcionalita byla ošetřena nějakými bezpečnostními prvky, stále by zde bylo velké riziko při odhalení chyb v zabezpečení a nabourání se do velkého množství počítačů přes závadný JavaScriptový kód. Z těchto důvodů se autoři JavaScriptu rozhodli vůbec neimplementovat podporu práce se soubory a tak plně eliminovat rizika spojené se zneužitím dané funkcionality.

Pro WYSIWYG editory to však znamená další problém a to jak vyřešit ukládání souborů. Jednou z možností je zobrazit náhled dokumentu v novém okně prohlížeče a využít možnosti uložení webové stránky prostřednictvím samotného webového prohlížeče. Tento způsob však není příliš vhodný například pro redakční systémy nebo pro upravování části obsahu webové stránky kdesi na Internetu, protože by uživatel musel pomocí prohlížeče pracně ukládat svůj výtvor na lokální disk a následně se připojovat přes FTP protokol a nahrávat modifikované soubory na vzdálený server.

Proto se pro ukládání nebo spíše odesílání dokumentů používá element <form> umožňující seskupit několik ovládacích polí do jednoho formuláře, který je možné najednou odeslat metodami `post` nebo `get` na libovolnou adresu, která s těmito údaji dále pracuje. Většinou se jedná o soubor umístěný na serveru obsahující PHP skript, který přijme poslaná data a může s nimi dále pracovat, například uložit do databáze nebo poslat klientovi vygenerovaný dokument se speciální hlavičkou, která vyzve uživatele k uložení jeho práce na lokální disk.

Dalším řešením je použít velmi populární technologii AJAX, pomocí níž je zdrojový kód, jako v případě formulářového elementu, odeslán na speciální soubor například s PHP skriptem. PHP soubor pro zpracování dat na serverové straně se pro použití formuláře a technologie AJAX shoduje. Výhodou AJAXu oproti formulářovému odesílání je pře-

devším v možnosti ponechání stávající stránky a rozepsaného textu aniž by se stránka musela znovu načítat nebo se musela pracně posílaná data na server odesílat zpět klientovi pro zobrazení na nově načítaných stránkách. V případě AJAXu tedy skript pouze odešle data na server a vyčkává na odpověď. Pokud server bude zrovna neaktivní, může WYSIWYG editor na tento stav upozornit.

Nevýhodou AJAXového a formulářového řešení je nutnost použití serveru.

2.2.6 Omezení JavaScriptových WYSIWYG editorů

Jeden z hlavních problémů a omezení JavaScriptových WYSIWYG editorů je specializace pouze na určitou skupinu jader umožňující použít vlastnosti pro editování dokumentu přímo ve webovém prohlížeči. Z tohoto důvodu jsou někteří uživatelé zaměření na exotičtější grafické nebo textové prohlížeče znevýhodněni a WYSIWYG editor jim pravděpodobně nebude fungovat nebo nebude fungovat optimálně.

Souvisejícím omezením je použitelnost na mobilních, PDA a jiných speciálních zařízeních. Tato vcelku malá a speciální zařízení jsou omezena svou velikostí a tudíž velikostí paměti, výkonem procesoru apod. Proto mají většinou značně omezené a upravené prohlížeče z důvodu snížení náročnosti a jejich jádro nemusí podporovat všechny speciální funkce vyžadované WYSIWYG editorem.

3 Vývoj aplikace

Základní principy WYSIWYG JavaScriptových editorů vypadají jednoduše, ale samotný vývoj aplikace je mnohem složitější a obtížnější, než se zprvu mohlo zdát.

Největším problémem při vývoji aplikace se stala nekompatibilita a rozdílné chování webových prohlížečů. Mnohdy určitá funkcionalita bezchybně fungovala v prohlížeči Firefox nebo Opera, avšak již nikoliv v prohlížeči Internet Explorer. Mnohé opravy chyb takovýchto nekompatibilit a rozdílů trvaly někdy pár hodin nebo dokonce i celý den, kdy byla přepsána polovička kódu.

Již před započetím programu jsem věděl, že se program musí dát snadno konfigurovat a přizpůsobit dle aktuálních potřeb webmastera či uživatele. Z tohoto důvodu jsem se rozhodl, aby bylo možno nakonfigurovat více editorů s různými menu zároveň na jedné stránce, cesty k souborům a obsahy jednotlivých menu.

Aby se navzájem různé a předem neznámé kopie editoru v jedné stránce neovlivňovaly nebo nebyly ovlivněny jiným kódem umístěným na stránce, bylo potřeba jej uzavřít do objektového modelu. Proto je editor rozdělen do čtyř základních částí.

3.1 Konfigurace

První ze čtyř základních částí je pole **editory**, které slouží pro základní konfiguraci jednotlivých editorů. Jednotlivé editory se do pole přidávací pomocí příkazu:

```
editory.push(["id", "menu", "umístění editoru", "typ"]);
```

Tímto příkazem se do pole editory přidávají záznamy s polem obsahující čtyři parametry, na základě kterých je pak generován WYSIWYG editor.

První parametr „id“ identifikuje element, místo kterého se má vložit WYSIWYG editor. Je nutné, aby byl vyplněn. Vkládání editoru není omezeno pouze na elementy <textarea> jako u mnoha jiných editorů, ale je možné jej použít i na jiné tagy. Například pro div obsahující provizorní jednodušší editor tvořený formulářem. Identifikátor musí být na stránce unikátní a v elementu se zapisuje jako atribut *id*. Pokud takovýto element s identifikátorem na stránce není, daný editor nebude vytvořen a pokračuje se dalším záznamem.

Druhým parametrem je „menu“ sloužící pro identifikaci nabídky, která má být pro daný editor použita. Tato nabídka se nastavuje v konfiguraci konstruktoru a je zde možné si

jednoduše vytvořit libovolné menu z jednotlivých funkcí editoru. Důvodem, proč se musí definovat menu, je variabilita aplikace. Takto je možné každému editoru na stránce přiřadit odlišné menu dle specifických požadavků.

Třetí parametr slouží pro informování generátoru editorů, kde se nachází soubor *temp.html*, který je používán jako hlavní součást ve WYSIWYG rozhraní a nad kterým se aktivuje editační mód jádra.

Čtvrtý a nepovinný parametr je určen především k testovacím účelům. Určuje, jaký element se má použít pro WYSIWYG rozhraní. V poslední verzi jsou implementovány dvě, ale je možné snadno do budoucna přidat i další. Jako první a základní je nastaven element <iframe>, jelikož druhá možnost použít element <object> nenabízí tak rozsáhlé možnosti kvůli špatné kompatibilitě prohlížečů a především velmi chabé podpoře ze strany Internet Exploreru. Zatímco <iframe> je generován pomocí funkce `document.createElement`, u objectu se musí obdobnou funkcí nejprve vytvořit obal z elementu a až poté se do něho pomocí funkce `innerHTML` vloží kód elementu <object>. Tento postup je nutný, jelikož pokud se vkládá daný kód před element <textarea>, což je nejčastější použití, Internet Explorer vypíše chybovou hlášku „Neznámá chyba při běhu programu“ a editor v tomto prohlížeči nebude fungovat. V ostatních prohlížečích problém nenastal. I po pracném řešení tohoto problému se později ukázaly další nevýhody a problémy, proč místo perspektivního elementu <object> použít již nemoderní <iframe>, který se používá ve všech ostatních JavaScriptových WYSIWYG editorech. Jednou z dalších možností je použití element se zapnutou vlastností *contenteditable*, avšak vzhledem k nemožnosti použití této vlastnosti v jádrech Gecko nebyla zatím tato možnost implementována a odzkoušena.

3.2 Inicializace editoru

Druhá ze základních částí WYSIWYG editoru je inicializace editoru. Inicializace programu se nachází v objectu `editor_ini`, která obsahuje dvě funkce, a to funkci `inicializuj` a `generuj`.

3.2.1 Funkce inicializuj

Funkce inicializuj má za cíl zajistit spuštění druhé inicializační části v okamžiku, kdy bude webová stránka plně načtena. Jedná se především o problém, kdy webové stránky nejsou

ještě plně načteny a skript by se již snažil vygenerovat WYSIWYG rozhraní. Generování WYSIWYG editoru by tak selhalo, protože v době implementace editoru by nemusely existovat elementy s identifikátorem předávaným v základní konfiguraci a tak by editor nefungoval.

V editoru widgEditor je tento problém řešen pomocí jednoduché funkce, která zkontroluje událost *onload* stránky, která se provádí po načtení webové stránky, a pokud zde není nahrána žádná funkce, načte druhou inicializační část programu. Pokud již nějaká funkce existuje, vloží novou funkci obsahující funkci původní i vkládanou. Původně jsem použil stejný postup, ale poté jsem zjistil, že dané řešení není úplně nejvhodnější. Z důvodu neustálých problémů s nekompatibilitou jsem se dokonce pokusil najít způsob, jakým lépe odladťovat problémy a použít metodu TDD. Použil jsem ji pro otestování této první části. Následně jsem však od TDD upustil, jelikož WYSIWYG editor je především grafickou aplikací a proto se zde výhody TDD nedají plně využít.

Hlavním problémem inicializační části widgEditoru bylo vyplnění základní události *onload* stránky v samotném kódu stránky. Autor zřejmě předpokládal, že se pro další funkce určené pro tuto událost bude používat obdobná inicializační funkce. Po provedení testů pomocí TDD a zkoumání problému jsem zjistil, že pokud se do základní události *onload* stránky v kódu vloží nějaká funkce, bude původní funkce vložená skriptem funkcí novou přepsána.

Z tohoto důvodu jsem se rozhodl najít lepší řešení. Tím se ukázal být standardní model událostí definovaný organizací W3C společně s nestandardním modelem nacházejícím se v prohlížeči Internet Explorer. Tyto modely jsou svou funkčností velmi podobné a události se u W3C kompatibilních prohlížečů přidávají pomocí funkce `addEventListener` a u Internet Exploreru pomocí `attachEvent`. Obě pro událost *onload* zajišťují spuštění funkce po načtení stránky nezávisle na základním nastavení událostí, což potvrdili i testy TDD.

3.2.2 Funkce generuj

Funkce `generuj` byla několikrát přepsána a původně měla sloužit pouze k vygenerování WYSIWYG editorů, avšak v pozdější fázi vývoje jsem narazil na chybu v prohlížeči Firefox, která způsobovala značné problémy a nefunkčnost editoru. Tato chyba byla zjevně zapříčiněna problémem spojeným s vytvořením a nastavením editačního módu webového rozhraní a využití funkce `execCommand`, která volala rozhraní, které v té

době ještě nebylo přepsáno.

Z tohoto důvodu se ve funkci generuj nachází několik ověřovacích podmínek konfigurace, generování základního rozhraní editoru a nakonec po určité časové prodlevě, kvůli zmiňované chybě Firefoxu, vytvoření vlastního editoru na základě definovaného konstruktoru.

Na začátku funkce se provádějí převážně kontroly konfigurace. Pokud není konfigurace vůbec nastavena, nebo jsou nastavené chybné údaje, generování aktuálního editoru se zruší a přejde se na generování následujícího editoru nakonfigurovaném v poli editory.

Po úspěšném projití kontrolní částí se získá umístění elementu, který má být nahrazen. Vygeneruje se hlavní element neboli „obal“, který obaluje celý editor a připojí se před nahrazovaný prvek, kterému bude v pozdější části nastaven style display na hodnotu „none“ a začne se chovat, jako by na stránce nebyl.

Následně je vygenerováno a do obalu editoru připojen prázdný obalový prvek menu, který je připojen pro další využití pomocí konstruktoru.

Po menu je potřeba vytvořit a připojit WYSIWYG rozhraní, které propojí aktuální stránku se stránkou temp, které se později zapne editační mód. Toto WYSIWYG okno se vytváří pomocí jednoho z elementů <iframe> či <object>.

Po dokončení přípravné části rozhraní WYSIWYG editoru se vytvoří instance objectu, pomocí konstruktoru editor a je deaktivován původní prvek.

3.3 Konstruktor editoru

Konstruktoru editoru se předávají dva parametry. Jako první je identifikátor původního nahrazovaného prvku, podle kterého se dále v kódu odvodí obal a další potřebné informace. Druhým parametrem je identifikace seznamu menu, které se má použít při generování hlavní nabídky.

Do proměnné temp se přiřadí celá cesta k temp souboru, pro jednodušší manipulaci a přehlednost. Pomocí *contentEditable* a *designMode* se posléze na soubor temp aplikuje editační mód prohlížeče.

V konstruktoru jsou také vytvořeny a přiřazeny další dva elementy.

Jedná se o pole <textarea> sloužící pro zobrazování čistého kódu. Přepínání mezi WYSIWYG módem a kódem je umožněno pomocí tlačítek v menu.

3.3.1 Vizuální prvky

Další součástí editoru je lišta, ve které se zobrazuje cesta k aktuálnímu prvku v kódu. Zjištění této cesty zajišťuje funkce `obnovaCesty`, která je volána každých 100 milisekund.

Součástí funkce je také kód zabezpečující vizuální znázornění formátování pomocí tlačítek menu, jak jej známe z textových WYSIWYG editorů.

3.3.2 Generování menu

Poslední částí konstruktoru je generování tlačítek menu do předem připraveného elementu . Zde se pomocí seznamu tlačítek menu volá funkce `typTlacitka`, která je připojena pomocí funkce `prototype`, což zajišťuje menší zatížení paměti u nižších verzí prohlížeče Internet Explorer, které se nešetrně chovají k operační paměti.

Volaná funkce přebírá dva parametry a to ukazatel na temp soubor a označení generovaného tlačítka. Vytváření tlačítek je založeno na příkazu s w i t c h, který vyhodnocuje proměnnou označující tlačítko a na základě jejího obsahu vykoná příkazy umístěné v návěštích, jejichž pojmenování je totožné. Tento způsob generování tlačítek umožňuje volitelnost a modifikovatelnost celého menu. V neposlední řadě tento způsob přináší snadnou rozšířitelnost o další funkce editoru dle aktuální potřeby.

3.3.3 Styly

Mezi tlačítky není možnost volby velikosti písmen, jejich barvy nebo velikosti. Tyto funkce není obtížné vytvořit, ale pro praktické vytváření webových dokumentů je dle mého hlediska vhodnější používat předem připravené externí styly a dodržet tak oddělení zdrojového a obsahového kódu od grafického popisu.

Editor proto umožňuje nastavování identifikátorů a tříd, které se můžou hned projevit nad daným dokumentem podle jednoho předem připraveného stylu, který se vybírá v konfiguraci.

3.3.4 Načítání a odesílání kódu pomocí AJAXu

Pro načítání a odesílání zdrojového kódu jsem se pro značné výhody rozhodl použít metodu AJAX. K tomuto účelu lze použít jedno z tlačítek, které slouží ke komunikaci se

Obrázek 10: Vytvořený WYSIWYG editor

serverem a umožňuje posílat či získávat všechen kód obsahu na předem konfigurovanou adresu serveru.

V souboru *server.php* je možné vidět jednoduchý skript, který zpracuje přijatá data a podle akce vygeneruje kód, který se odešle zpět editoru, kde si jej převezme AJAX. Je zde i ukázka, jak pomocí serveru a upravení hlavičky donutit prohlížeč k uložení souboru, který fyzicky na serveru neexistuje.

3.3.5 Speciální tlačítka

Vzhledem k zahrnutí AJAXu mezi funkce přístupné při generování tlačítek menu je možné tuto metodu využít i u jiných specializovaných funkcí.

Do menu je možné vkládat i speciální tlačítka pro grafické rozdělení nabídky. Pro vložení prázdného místa se používá tlačítko „nic“ a pro ukončení řádku tlačítko „radek“.

3.4 Parser

Poslední důležitou částí editoru je konstruktor `Parser`, podle kterého se v editoru vytváří instance objektu a slouží pro validaci kódu.

Tato část byla převzata z projektu Bronziho WYSIWYG editor z důvodu její komplexnosti, nezávislosti a tudíž i snadné nahraditelnosti nebo modifikovatelnosti. Nahrazuje tak dříve použitou funkci `validuj`, která nenabízela dostatečné množství oprav kódu a nebyla na takové úrovni.

Hlavním cílem je na vstupu získat nevalidní, nekompatibilní kód vytvořený různými jádry prohlížečů a převést jej do jednotné formy a validního stavu podle standardů W3C.

4 Testování

Vývoj i testování proběhlo na linuxové distribuci Ubuntu 7.10 Gutsy Gibbon s nainstalovaným Apache 2.0 serverem a PHP ve verzi 5.2.3-1ubuntu6.3.

Pro vývoj kódu aplikace byl použit editor Gvim verze 7.1.56 se zvýrazňováním syntaxe a pro tvorbu grafických prvků byl používán grafický editor Gimp 2.4.2.

Chování editoru bylo průběžně testováno v prohlížečích Firefox 2.0.0.13, Internet Explorer 6.0 a Opera 9.24. Dodatečně byl otestován v prohlížeči Microsoft Internet Explorer 7.0 spuštěném v systému Windows XP Home spuštěném pomocí nástroje Virtualbox.

4.1 Grafika a výstup

Po grafické stránce design editoru trpí stejnými rozmary prohlížečů jako každá jiná webová stránka. Z tohoto důvodu vzhled editoru závisí čistě na správném nadefinování kaskádových stylů. Vzhled editoru lze snadno přizpůsobit a vesměs přejímá grafiku stránky, na kterou je umístěn. Ve všech testovaných prohlížečích kromě nepodporovaného Konqueroru se editor zobrazuje korektně.

Vzhledem k použití komplexního a složitého skriptu pro úpravu zdrojového kódu se výstup editoru chová obdobně ve všech testovaných prohlížečích.

4.2 Konqueror a ostatní prohlížeče

Webový prohlížeč Konqueror s jádrem KHTML, používaný především v linuxovém prostředí KDE, nepodporuje standardně vnitřní užívané funkce WYSIWYG editorů jako mnoho dalších méně rozšířených či okrajových prohlížečů. Z tohoto důvodu je v konstruktoru implementována detekce zabraňující zobrazení nefunkčního editoru. Při zjištění problému jsou dosavadní připravené části WYSIWYG kódu, které ohraničující editor, odstraněny pomocí metody `removeChild` a na stránce tak nezůstane nefunkční torzo editoru, které by mátlo uživatele.

Místo WYSIWYG editoru se proto zobrazí původní nahrazovaný element, který by měl obsahovat náhradní řešení pro editaci kódu s případným předem připraveným upozorněním o nefunkčnosti původního editoru.

5 Závěr

I přes nesčetná úskalí a nezdary se mi podařilo vytvořit požadovaný WYSIWYG editor psaný v JavaScriptu a optimalizovat jej pro nejpoužívanější editory.

Avšak jako každý složitější a rozsáhlejší program má i tento řadu skrytých a nenápadných chybiček, které by bylo potřeba odladit. Také by editoru slušelo lepší a názornější rozhraní pro vkládání tabulek, rozhraní pro vkládání či úpravu obrázků a také prostředí pro práci s kaskádovými styly.

V poslední době se po dlouhé stagnaci okolo webových standardů a prohlížečů začínají dít velké věci. Nedávno vyšla nová verze Internet Explorer 7 a již se pracuje na verzi nové. Zanedlouho má vyjít Firefox 3 a také se znovu začalo pracovat na standardu HTML 5. Především větší aktivita při vydávání nových verzí prohlížeče Internet Explorer a ohlašovaná změna politiky společnosti Microsoft k lepšímu dodržování webových standardů může způsobit, že zanedlouho bude vytvoření WYSIWYG editoru velmi snadnou záležitostí a odpadne tak mnoho problémů.

Literatura

[1] DRUSKA, Peter: *CSS a XHTML*
ISBN 80-247-1382-9, Vydavatelství Grada Publishing, Praha 2006.

[2] ŠKULTÉTY, Rastislav: *JavaScript: Kapesní přehled*
ISBN 80-251-0884-8 , Vydavatelství Computer Press, a.s., Brno 2005.

[3] ASLESON, R. – SCHUTTA, N. T.: *Ajax: vytváříme vysoce interaktivní webové aplikace.* ISBN 80-251-1285-3, vydavatelství Computer Press, a.s., Brno 2006.

[4] Bronziho web [online]:
URL: <http://www.bronzi.cz>

[5] FCKeditor [online]:
URL: <http://www.fckeditor.net>

[6] Interval.cz – *Slabikář JavaScriptu* [online]:
URL: <http://interval.cz/serialy/slabikar-javascriptu>

[7] JavaScript and Object Oriented Programming [online]:
URL: <http://www.javascriptkit.com/javatutors/oopjs.shtml>

[8] Root.cz [online]:
URL: <http://www.root.cz>

[9] Root.cz – *TeX? Klidně i pro naprosté začátečníky!* [online]:
URL: <http://www.root.cz/clanky/
tex-klidne-i-pro-naproste-zacatecniky>

[10] tinyMCE [online]:
URL: <http://tinymce.moxiecode.com>

[11] W3 Consortium [online]:
URL: <http://www.w3.org>

[12] widgEditor [online]:
URL: <http://www.themaninblue.com/experiment/widgEditor>

Obsah CD

/bp/LaTeX/ – adresář s bakalářskou prací v latexu

/bp/bp.pdf – bakalářská práce ve formátu PDF

/editor/ – adresář s editorem

/cti.txt – popis obsahu CD a pokyny pro instalaci editoru