VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

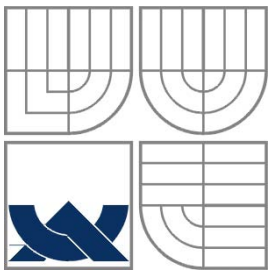
EDITOR ANOTACÍ PRO WYSIWYG TEXTOVÉ EDITORY NAPSANÉ V JAZYCE JAVASCRIPT

DIPLOMOVÁ PRÁCE
MASTER‘S THESIS

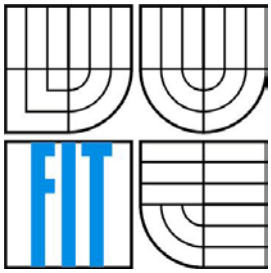
AUTOR PRÁCE
AUTHOR

BC. MARTIN KLEBAN

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

EDITOR ANOTACÍ PRO WYSIWYG TEXTOVÉ EDITORY NAPSANÉ V JAZYCE JAVASCRIPT

ANNOTATIONS EDITOR FOR WYSIWYG JAVASCRIPT-BASED TEXT EDITORS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BC. MARTIN KLEBAN

VEDOUCÍ PRÁCE
SUPERVISOR

ING. JAROSLAV DYTRYCH

BRNO 2011

Zadání diplomové práce/11068/2010/xkleba00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2010/2011

Zadání diplomové práce

Řešitel: **Kleban Martin, Bc.**
Obor: Informační systémy
Téma: **Editor anotací pro WYSIWYG textové editory napsané v jazyce JavaScript**
Annotations editor for WYSIWYG JavaScript-Based Text Editors
Kategorie: Web

Pokyny:

1. Seznamte se s různými WYSIWYG textovými editory vytvořenými v jazyce JavaScript (např. TinyMCE).
2. Srovnejte různé editory a prozkoumejte možnosti jejich rozšíření.
3. Navrhněte editor strukturovaných anotací, který bude rozšířením pro vybrané editory a bude obousměrně asynchronně komunikovat se serverem. Pro uživatelské rozhraní navrhněte vlastní řešení, přičemž se pokuste vyhnout využití složitých univerzálních knihoven.
4. Implementujte navržené rozšíření.
5. Zhodnoťte dosažené výsledky a srovnejte s alternativními přístupy.

Literatura:

- dle doporučení vedoucího

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1, 2 a 3.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Dytrych Jaroslav, Ing.**, UPGM FIT VUT
Datum zadání: 20. září 2010
Datum odevzdání: 25. května 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato práce se zabývá analýzou možností tvorby rozšíření pro WYSIWYG textové editory napsané v jazyce JavaScript (TinyMCE, CKEditor, NicEdit, jWYSIWYG) a popisuje návrh editoru anotací jako rozšíření pro zvolené editory. Editor strukturovaných anotací byl implementován pro WYSIWYG editor TinyMCE a obsahuje vlastní řešení uživatelského rozhraní bez použití složitých univerzálních knihoven. V závěru jsou shrnuty získané poznatky z analýzy WYSIWYG editorů a implementace editoru anotací.

Abstract

This master's thesis deals with analysis of plugins development possibilities for WYSIWYG JavaScript-based text editors (TinyMCE, CKEditor, NicEdit, jWYSIWYG) and it describes design of annotations editor as a plugin for chosen editors. Structured annotations editor was implemented for TinyMCE and it includes own user interface implementation without usage of any complex universal libraries. Gained pieces of knowledge from WYSIWYG editors analysis and annotations editor implementation are to be found in the enclosure.

Klíčová slova

JavaScript, WYSIWYG, textový, editor, TinyMCE, CKEditor, NicEdit, jWYSIWYG, rozšíření, editor anotací, anotace

Keywords

JavaScript, WYSIWYG, text, editor, TinyMCE, CKEditor, NicEdit, jWYSIWYG, extension, annotations editor, annotation

Citace

Kleban Martin: Editor anotací pro WYSIWYG textové editory napsané v jazyce JavaScript, diplomová práce, Brno, FIT VUT v Brně, 2011

Editor anotací pro WYSIWYG textové editory napsané v jazyce JavaScript

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně pod vedením Ing. Jaroslava Dytrycha. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Martin Kleban
24.5.2011

Poděkování

Děkuji svému vedoucímu Ing. Jaroslavu Dytrychovi za trpělivost při vedení této práce, za jeho konstruktivní návrhy a podnětné připomínky.

Obsah

1 Úvod

Cieľom tejto práce je výber štyroch WYSIWYG (anglicky What You See Is What You Get) textových editorov vytvorených v jazyku JavaScript a analýza možností ich rozšírenia. Súčasťou práce je aj návrh editora anotácií ako modulu pre zvolené editory a jeho implementácia pre WYSIWYG editor TinyMCE.

Kapitola 2 stručne popisuje zvolené technológie, pričom sa zameriava hlavne na ich krátku definíciu, históriu, aktuálne verzie, resp. na smerovanie ďalšieho vývoja. V tejto kapitole nájdeme popis služby World Wide Web, jazykov HTML (XHTML), CSS, JavaScript, XML, XPath, Java, metód pre asynchronnú komunikáciu AJAX a Comet a tiež knižnice jQuery pre JavaScript.

V kapitole 3 sa nachádza analýza WYSIWYG editorov, ktorá popisuje zvolené editory NicEdit, jWYSIWYG, TinyMCE a CKEditor. Pri každom editore je uvedená jeho štruktúra, rozsah, možnosti konfigurácie, opis tvorby rozšírení, licencia editora a jeho podpora webových prehliadačov.

Kapitola 4 obsahuje návrh editora anotácii, ktorý je vyvýjaný ako súčasť rozsiahleho a unikátneho systému pre kolaboratívne anotovanie v reálnom čase, ktorý je predmetom dizertačnej práce vytváranej na FIT VUT [47]. Osobitá pozornosť je venovaná inžinierskemu návrhu knižnice pre uživateľské rozhranie ako aj samotnému návrhu obrazoviek a popisu chovania.

V kapitole 5 sa rozoberá implementácia systému v jazyku JavaScript. Sú v nej spomenuté problémy, ktoré sa pri implementácii objavili spolu s popisom ich riešenia.

Kapitola 6 sa venuje testovaniu a popisuje spôsoby ladenia vyvýjanej aplikácie v prostredí webových prehliadačov.

V závere sú zhrnuté získané poznatky z analýzy WYSIWYG editorov a implementácie editora anotácií, ako aj návrh možných rozšírení do budúcna.

2 Technológie

Nasledujúca kapitola sa venuje využitým technológiám, ktoré boli zvolené vzhľadom k požiadavkám na výsledné riešenie a vzhľadom k tomu, že boli využité pri implementácii editora anotácií. Zameriava sa hlavne na ich stručný popis, históriu, aktuálne verzie, resp. na smerovanie ďalšieho vývoja.

K týmto technológiám radíme službu World Wide Web, ďalej jazyky používané pri tvorbe webových stránok a aplikácií ako HTML (XHTML), CSS a JavaScript. Ďalšími sú: metóda pre asynchronnú komunikáciu AJAX, Comet, knižnica jQuery, jazyk XML, XPath a jazyk Java.

2.1 World Wide Web

World Wide Web, známy ako WWW, Web alebo W3 je množina internetových protokolov a ďalších, s nimi súvisiacich technológií, ktorá slúži k prezentácii hypertextových informácií [1].

Autorom vízie je Timothy Bernes-Lee, ktorý ju vyvinul v roku 1989 počas pôsobenia v Európskom centre pre jadrový výskum CERN. V súčasnosti riadi vývoj štandardov súvisiacich s webovými technológiami konzorcium World Wide Web Consortium (označované tiež skratkou W3C). Špeciálne pre World Wide Web bol vyvinutý protokol HTTP, jazyk HTML, rozhranie CGI a adresy URL [1].

2.1.1 Hypertext

Hypertext je text, ktorý obsahuje odkazy na ďalšie texty [2]. Rodina protokolov World Wide Web bola jedna z prvých, ktorá umožňovala rýchle a efektívne prezeranie veľkého množstva nesekvečne uložených informácií. Informácie na Webe sú uložené vo forme orientovaného grafu [1].

2.1.2 Zdroje

Jednotlivé miesta v prostredí internetu nazývame zdroje. Zdroje poskytujú nejaké dáta, ktoré sa následne spracúvajú. Ide teda o zdroje dát [1].

Zdroje sú označované identifikátormi. Podľa [1] existujú dva typy identifikátorov:

- Identifikátory špecifikujúce miesto v sieti, a teda sú závislé na umiestnení v sieti (anglicky Uniform Resource Locator - URL).
- Identifikátory pomenuvávajúce zdroj, nezávislé na umiestnení v sieti (anglicky Uniform Resource Name - URN).

Spoločným názvom pre obe tieto skupiny je Universal Resource Identifier - URI [1].

2.1.3 HTTP

HTTP (anglicky Hypertext Transfer Protocol) je v súčasnosti najpoužívanejším protokolom pre prenos dát medzi klientom a serverom. Pracuje s adresami URL a bol vytvorený pre hypertextové prostredia [1].

Podľa W3C [3] je posledná špecifikácia HTTP/1.1 stabilná a konzorcium pozastavilo svoje aktivity v ďalšom vývoji. Posledná verzia splnila všetky požiadavky na vytvorenie úspešného štandardu, ktorý odstránil nedostatky predchádzajúcich verzií HTTP protokolu. Presnú špecifikáciu HTTP/1.1 popisuje dokument RFC 2616 [32].

2.1.4 CGI

CGI (anglicky Common Gateway Interface) je jednoduchým štandardom na prepájanie externých aplikácií s informačnými servermi. V súčasnosti sa ako informačné servery používajú HTTP servery. Rozhranie CGI sa v službe World Wide Web využíva od roku 1993 [4].

Poslednou verziou štandardu je CGI/1.1 a jeho presnú špecifikáciu popisuje dokument RFC 3875 [4], ktorý vytvorili v roku 2004 D. Robinson a K. Coar.

2.2 HTML a XHTML

HTML (anglicky HyperText Markup Language) je typom dokumentu SGML (anglicky Standard Generalized Markup Language - metajazyk určený na popis značkovacích jazykov [16]). Ide o značkovací jazyk, ktorý sa používa na popis dokumentov v prostredí World Wide Web. Pripravovaná verzia HTML 5 už ale závislosť na SGML obsahovať nebude. HTML poskytuje prostriedky na publikovanie online dokumentov s nádpismi, textom, tabuľkami, fotografiami, vytváranie formulárov, vkládanie videa, zvuku a iných aplikácií priamo do dokumentu [5, 6].

HTML je charakterizovaný množinou značiek a ich atribútov definovaných pre danú verziu. Atribúty sú doplňujúce informácie, ktoré upresňujú vlastnosti elementu [5].

XHTML je variantou HTML, ktorá využíva syntax jazyka XML (anglicky Extensible Markup Language). Keďže XHTML je aplikáciou XML, je pri týchto dokumentoch navyše možné využiť aj ďalšie XML nástroje (ako napr. XSLT, jazyk určený na transformácie XML dokumentov) [5].

Poslednými verziami špecifikácií sú HTML 4.01 z roku 1999 a XHTML 1.1 z roku 2010. Zároveň však pokračujú práce na štandarde HTML 5 a jeho XML variante XHTML 5 [7].

2.3 XML

XML (anglicky Extensible Markup Language) je jednoduchý textový formát určený na reprezentáciu štruktúrovaných informácií. Za vznikom štandardu stojí konzorcium W3C. XML je odvodený od staršieho štandardu SGML (ISO 8897). Poslednou verziou je Extensible Markup Language (XML) 1.0 (Piate vydanie) z novembra 2008 [9, 10].

Spolu s XML sa od jeho vzniku vyvinulo mnoho súvisiacích technológií, ako napr.:

- XSLT (anglicky eXtensible Stylesheet Language Transformations) - Jazyk určený na transformácie XML dokumentov na iné XML, textové alebo HTML dokumenty [17].
- XQuery - Štandardizovaný jazyk na dotazovanie širokého spektra informačných XML zdrojov (dokumenty, databázy, a pod.). Posledná verzia štandardu je z decembra 2010 a je dostupná na [18].
- XPath - Popísaný v kapitole 2.4.
- A ďalšie.

Pojem XML sa tak často používa na označenie jazyka XML spolu s týmito technológiami ako celku.

2.4 XPath

XPath (v najnovšej verzii XPath 2.0 z decembra 2010) je jazyk, ktorý umožňuje spracovanie hodnôt, ktoré zodpovedajú dátovému modelu definovanému v XDM (anglicky XQuery/XPath Data Model - dátový model, ktorý definuje informácie obsiahnuté vo vstupoch pre XSLT a XQuery procesor a tiež vymedzuje povolené hodnoty výrazov v jazykoch XSLT, XQuery a XPath [20]).

Výsledkom výrazu XPath môže byť výber uzlov zo vstupného dokumentu, atomická hodnota, alebo vo všobecnosti, ľubovoľná sekvencia povolená dátovým modelom. XPath 2.0 je nadmnožinou XPath 1.0 s pridanými rozšíreniami, napr. pre podporu bohatšej sady dátových typov a pod. XPath 2.0 je spätne kompatibilný s verziou XPath 1.0 a takmer všetky výrazy XPath 1.0 vracajú rovnaké výsledky aj v XPath 2.0 [19].

2.5 CSS

CSS (anglicky Cascading Style Sheets) je jazyk pre popis spôsobu zobrazovania obsahu dokumentov, vrátane farieb, rozloženia a písma. To umožňuje prispôsobiť prezentáciu na rôzne typy zariadení, ako sú veľké obrazovky, malé obrazovky, alebo tlačiarne. CSS je nezávislý na HTML a možno ho použiť s akýmkoľvek jazykom založeným na XML [5].

Hlavným zmyslom CSS je umožnenie oddelenia vzhľadu dokumentu od jeho štruktúry a obsahu. To značne uľahčuje spravovanie webových stránok a umožňuje zdieľanie štýlov medzi viacerými stránkami [5].

Jazyk CSS navrhol v roku 1994 Håkon Wium Lie [21] a v súčasnosti je udržovaný organizáciou W3C. Doposiaľ boli vydané dve úrovne špecifikácie na úrovni štandardu, poslednou je štandard Cascading Style Sheets, level 2 (skrátene CSS2) z roku 2008. Kandidátom na jeho nahradenie je revízia CSS 2.1, ktorá pridáva niekoľko rozšírení, ktoré sa v súčasnosti už často vyskytujú v najnovších verziách prehliadačov [22]. Vývoj ďalej pokračuje na pripravovanej verzii CSS3 [8].

2.6 JavaScript

JavaScript je interpretovaný programovací jazyk so základnou objektovo-orientovanou koncepciou. Jádro jazyka je syntakticky veľmi podobné C, C++ alebo Jave, no podobnosť však končí na úrovni syntaxe. JavaScript je jazykom, ktorý má potlačenú typovú kontrolu a mnohé myšlienky preberá z jazyka Perl [23].

JavaScript pôvodne vznikal pod názvom LiveScript a zmena názvu nastala tesne pred jeho uvedením na trh z marketingových dôvodov. Za vývojom stáli spoločnosti Netscape a Sun Microsystems. V produktoch Microsoft sa tento jazyk vyskytuje pod názvom JScript [23, 11].

V roku 1997 bol organizácii ECMA (European Computer Manufacturers Association - združenie európskych výrobcov počítačov) predložený JavaScript 1.1 (vyskytujúci sa v prehliadači Netscape Navigator 3) ako návrh pre štandardizáciu. Po určitých korekciách tak vznikol štandard ECMA-262 definujúci nový skriptovací jazyk ECMAScript [11].

Okrem jazyka JavaScript implementujú jazyk ECMAScript aj iné jazyky, ako napr. Adobe ActionScript a OpenView ScriptEase. Najnovšiou edíciou štandardu ECMA-262 je edícia 5 vydaná v roku 2009 [24].

Pojem *klientský JavaScript* vznikol integráciou JavaScriptu do webového prehliadača. Klientský JavaScript používa okrem univerzálnych rysov jazyka aj [23, 11]:

- Objektový model dokumentu (anglicky Document Object Model, resp. DOM), ktorý poskytuje rozhranie pre prácu s obsahom dokumentov.
- Objektový model prehliadača (anglicky Browser Object Model, resp. BOM), ktorý poskytuje rozhranie pre interakciu s prehliadačom.

Okrem klientského využitia je možné jadro jazyka obohatené o ďalšie knižnice používať aj mimo prehliadače, napr. v PDF dokumentoch alebo desktopových aplikáciách. JavaScript je možné použiť aj na strane serveru. Prvou implementáciou JavaScriptu na strane serveru bol LiveWire firmy Netscape uvedený v roku 1996 [25]. Dnes existuje niekoľko možností vrátane open-source implementácie Rhinola [26] založenej na Rhino (open-source implementácia jazyka JavaScript napísaná v jazyku Java [27]).

Porovnanie verzií jazyka JavaScript k jednotlivým edíciam štandardu ECMA-262 a ich podporu v najpoužívanejších prehliadačoch zobrazuje tabuľka č.1:

Verzia JavaScriptu	Ekvivalentné k	Netscape Navigator	Mozilla Firefox	Internet Explorer	Opera	Safari	Chrome
1.0		2.0		3.0			
1.1		3.0					
1.2		4.0 - 4.05					
1.3	ECMA-262 (1. edícia) / ECMA-262 (2. edícia)	4.06 - 4.7x		4.0			
1.4		Netscape Server					
1.5	ECMA-262 (3. edícia)	6.0	1.0	5.5 (JScript 5.5), 6 (JScript 5.6), 7 (JScript 5.7), 8 (JScript 5.8)	6.0 - 11.0	3.0 - 5	1.0 - 10.0.666
1.6	JavaScript 1.5 + vylepšenia		1.5				
1.7	JavaScript 1.6 + vylepšenia		2.0				
1.8	JavaScript 1.7 + vylepšenia		3.0				
1.8.1	JavaScript 1.8 + vylepšenia		3.5				
1.8.2	JavaScript 1.8.1 + vylepšenia		3.6				
1.8.5	JavaScript 1.8.1 + vylepšenia + niektoré časti z ECMA-262 (5. edícia)		4	9			

Tabuľka 1: Porovnanie verzií jazyka JavaScript s edíciami štandardu ECMA-262 a ich podpora v najpoužívanejších prehliadačoch [28, 29].

Častou otázkou ohľadom OOP v jazyku JavaScript býva, či tento jazyk obsahuje triedy (anglicky class). Podľa [48] JavaScript klasické triedy nemá, no medzi programátormi JavaScriptu platí konsenzus, že za triedu považujeme konštrukčnú funkciu, skrátene konštruktor, ktorý využíva vlastnosť *prototype*.

2.7 AJAX

AJAX (anglicky Asynchronous JavaScript And XML) nepredstavuje úplne novú technológiu, ide skôr o nový prístup, ako využiť existujúce technológie pre zabezpečenie asynchrónnej komunikácie medzi prehliadačom a HTTP serverom [12].

AJAX zjednocuje následujúce technológie [30]:

- HTML (XHTML), CSS.
- DOM - objektový model dokumentu.
- XML a XSLT pre výmenu a transformácie dát.
- Objekt XMLHttpRequest pre asynchrónnu komunikáciu so serverom.
- JavaScript na prepojenie predošlých častí.

Za autora pojmu AJAX sa považuje Jesse James Garrett, ktorý v roku 2005 publikoval článok s názvom Ajax: nový prístup k webovým aplikáciám (anglicky Ajax: A new approach to web applications) [13, 30].

2.8 Comet

Comet predstavuje ďalší prístup k tvorbe webových aplikácií. Tento koncept umožňuje posielať dáta zo servera ku klientovi v ľubovoľnom čase a to nie len ako odpoveď na predchádzajúci dotaz. Dáta sú prenášané cez jedno udržiavané spojenie. Tento prístup výrazne znižuje latenciu (dobu odozvy) pri poskytovaní dát zo servera [31].

Comet je podobne ako AJAX založený na asynchrónnej komunikácii medzi klientom a serverom. Aplikácie, ktoré implementujú koncept Comet, majú pri zmene stavu možnosť komunikovať s minimálnou latenciou, čo je obrovská výhoda oproti prístupu s využitím AJAX. Preto je tento prístup oveľa výhodnejší pri tvorbe viacužívateľských webových aplikácií určených pre spoluprácu medzi užívateľmi v reálnom čase [31].

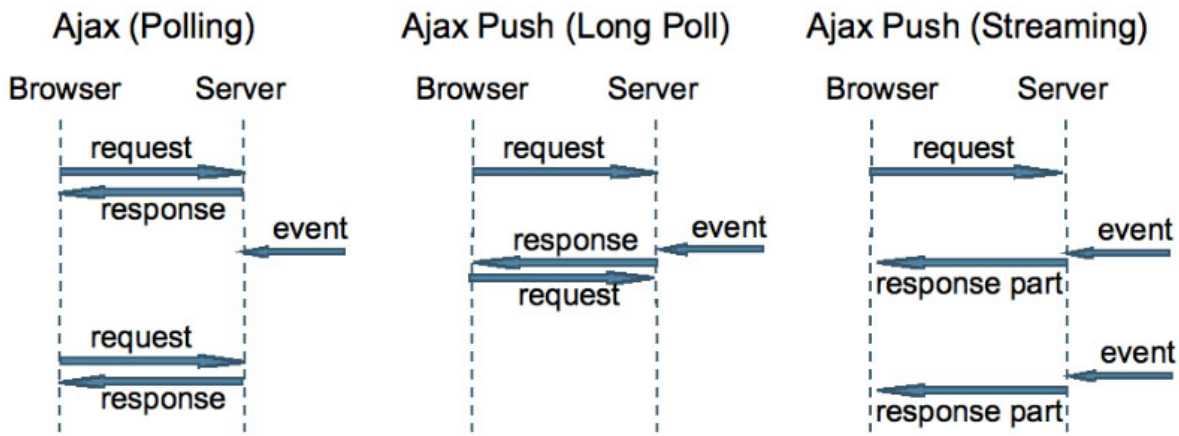
Autorom pojmu Comet je Alex Russel a názov je humornou narážkou na AJAX, pretože oba tieto názvy existujú aj ako názvy čistiacich prostriedkov [13].

Obrázok č.1 znázorňuje tri spôsoby asynchrónnej komunikácie medzi prehliadačom a serverom: Polling, Long Poll a Streaming. Prvý z nich (Polling) funguje na klasickom princípe AJAX, kedy klient zasiela požiadavky na server v pravidelných intervaloch a informácie o vzniknutej udalosti môže obdržať až po zaslaní požiadavku. Ak nenastala žiadna udalosť, klient obdrží prázdnu správu - bez užitočnej informácie. Polling preto nespadá medzi metódy označované pojmom Comet [49].

Ďalšími spôsobmi komunikácie sú Long Poll a HTTP Streaming. Pojem Comet môžeme spojiť s oboma z nich. Pri metóde Long Poll pošle klient na server požiadavok, no odpoveď dostane až pri výskyte udalosti na strane servera. Po jej prijatí posiela klient automaticky ďalší požiadavok. Tento spôsob umožňuje klientovi obdržať informácie o vzniknutej udalosti prakticky s minimálnou latenciou [49].

HTTP Streaming využíva princípy implementované v priehliadačoch, ktoré slúžia na urýchlenie vykresľovania stránok, napr. progressive rendering (anglicky postupné vykresľovanie).

Pri metóde progressive rendering dochádza k realizovaniu stránky v prehliadači hneď po prijatí prvku <*body*>. S príchodom nových dát sa pokračuje v ďalšom zobrazovaní stránky. Prehliadač ale netuší, ako dlho má čakať na nové dáta, ani koľko dát príjde - a to tvorí podstatu HTTP streamingu. Server tak posiela časti odpovede postupne, pri výskyte nových udalostí [13, 49].



Obrázok 1: Princíp metód Polling, Long Poll a HTTP Streaming [49].

2.8.1 Grizzly Framework

Grizzly Framework [52] je rozšírenie pre aplikačný server GlassFish [53], ktoré vzniklo za účelom zvýšenia jeho rozšíriteľnosti a rýchlosti. Súčasťou Grizzly Frameworku je okrem iného aj Grizzly Comet Framework, ktorý obsahuje sadu komponent umožňujúcich vytváranie aplikácií s vyžitím technológie Comet [49].

Poslednou oficiálnou verziou bola v dobe písania tejto práce Grizzly Framework 2.1 z 26. apríla 2011 [52]. Na serveri pre prácu s anotáciami sa použila verzia 1.9.18-o.

2.9 jQuery

JQuery patrí medzi najpoužívanejšie knižnice napísané v jazyku JavaScript. Zjednodušuje prístup k jednotlivým uzlom objektového modelu dokumentu, uľahčuje prácu s udalosťami, animáciou, AJAXom a celkovo tak umožňuje rýchlejší vývoj webových aplikácií [14].

Autorom prvej verzie bol v roku 2006 John Resig, ktorý na projekte aj naďalej pracuje ako hlavný vývojár. Poslednou verziou v dobe písania tejto práce bola jQuery 1.4.2 z februára 2010.

2.10 Java

Java je objektovo-orientovaný programovací jazyk, ktorého autorom je James Gosling, ktorý na ňom pracoval ako vedúci vývojového tímu v spoločnosti Sun Microsystems (dnes Oracle). Jazyk Java môžeme charakterizovať ako [50]:

- Jednoduchý - Zjednodušená verzia syntaxe jazyka C a C++.
- Distribuovaný - Umožňuje vytvárať distribuované aplikácie.
- Interpretovaný - Namiesto skutočného strojového kódu sa vytvára iba tzv. medzikód. Tento formát je nezávislý na architektúre počítača alebo zariadenia a je interpretovaný

virtuálnym strojom (označovaným ako JRE - Java Runtime Environment). JRE býva súčasťou balíka JDK (Java Development Kit), ktorého poslednou verziou v dobe písania tejto práce bola: Version 6 Update 20.

- Prenositeľný - Jazyk je okrem nezávislosti na architektúre nezávislý aj z hľadiska základných dátových typov.
- Robustný - Vhodný pre vytváranie robustých aplikácií.

Podľa [51] existujú v súčasnosti štyri edície Javy, ktoré sa líšia podľa cieľovej platformy. Konkrétne ide o Java Card (určená pre čipové karty), Java ME (Java Platform, Micro Edition - pre zariadenia s limitovanými prostriedkami), Java SE (Java Platform, Standard Edition - pre pracovné stanice) a Java EE (Java Platform, Enterprise Edition - vhodná pre distribuované prostredia a robustné aplikácie).

3 Analýza WYSIWYG editorov

Dôležitou časťou tejto práce je výber a analýza štyroch WYSIWYG textových editorov napísaných v jazyku JavaScript, z ktorých sa vyberie jeden, pre ktorý sa implementuje navrhovaný editor anotácií. Táto kapitola popisuje práve tieto štyri editory, stručne sa zaoberá ich históriou, vývojom, aktuálnymi verziami a rozsahom, no dôraz sa kladie hlavne na popis tvorby rozšírení a celkovú prácu s API.

Pri výbere editorov sa zohľadňovala najmä ich popularita a široké nasadenie v praxi. Cieľom bolo vybrať dva rozsiahlé a dva jednoduché editory, všetky s rozumnou a nekomerčnou licenciou. Výsledkom výberu sú rozsiahlé editory TinyMCE, CKEditor a jednoduché NicEdit a jWYSIWYG, ktorý je postavený na knižnici jQuery.

3.1 NicEdit

NicEdit [15] je jednoduchý WYSIWYG editor vytvorený v jazyku JavaScript. Autorom je Brian Kirchoff, ktorý na ňom pracuje od novembra 2007. Poslednou verziou v dobe písania tejto práce bola NicEdit 0.9 r23 z januára 2009.

Na obrázku č. 2 je ukážka vzhľadu editora NicEdit.

Obrázok 2: Ukážka vzhľadu editora NicEdit [15].

3.1.1 Štruktúra a rozsah

Editor NicEdit v štandardnej edícii pozostáva z týchto častí [15]:

- *nicCore* - Predstavuje jadro editora a je nevyhnutnou súčasťou pre všetky ďalšie rozšírenia.
- *nicPane* - Umožňuje vytváranie dialógových okien, okien s nápovedou a pod.
- *nicAdvancedButton* - Slúži na vytváranie tlačidiel do užívateľského rozhrania.
- *nicButtonTips* - Umožňuje zobrazovanie okien s nápovedov pri podržaní myši nad tlačidlom v užívateľskom rozhraní editora.
- *nicSelect* - Poskytuje možnosť označenia textu v editore a následné priradenie vlastností ako veľkosť, typ písma a formát.
- *nicLink* - Poskytuje funkcionalitu na vytváranie hypertextových odkazov.
- *nicColors* - Slúži na zmenu farby vybraného textu alebo jeho pozadia.
- *nicImage* - Umožňuje vkladanie obrázkov do textu.
- *nicSave* - Poskytuje možnosť ukládania obsahu prostredníctvom technológie AJAX.

K dispozícii sú aj ďalšie oficiálne vytvorené moduly [15]:

- *nicUpload* - Umožňuje nahrávanie (anglicky upload) obrázkov s využitím služby ImageShack.

- *nicXHTML* - Upravuje kód tak, aby odpovedal štandardu XHTML.
- *nicBBCode* - Umožňuje vytvoriť BBCode pre použitie v rôznych fórach a pod.
- *nicFloating* - Poskytuje možnosť vytvoriť plávajúci (anglicky floating) panel editora.
- *nicCode* - Slúži na editovanie HTML kódu obsahu editora.

Výhodou editora NicEdit je okrem jeho jednoduchosti aj veľkosť štandardnej edície, ktorá má menej ako 35kB (vrátane obrázkov uživateľského rozhrania) a menej ako 10kB v komprimovanej forme. Celý editor pozostáva iba z dvoch súborov, zdrojového kódu v jazyku JavaScript a zo súboru s ikonami.

Editor so svojou flexibilnou konfiguráciou umožňuje nahradzovať značky typu TEXTAREA a DIV, a nahradiť ich plnohodnotným prostredím pre úpravu obsahu. Jeden dokument môže obsahovať viacero editorov súčasne.

3.1.2 Konfigurácia

NicEdit je vysoko-konfigurovateľný nástroj. Konfiguračný objekt stačí predať konštruktoru pri vytváraní inštancie editorom, ako ukazuje následujúci zápis:

```
new NicEditor({KONFIG_OBJEKT}).panelInstance('TEXTAREA_ID')
```

Konfiguračný objekt má tvar:

```
{vlastnosť_01:hodnota, vlastnosť_02:hodnota, ...}
```

Medzi dostupne konfiguračné vlastnosti patria [15]:

- *fullPanel* - Ak sa táto hodnota nastaví na *true*, vytvorený editor bude poskytovať funkcionalitu všetkých dostupných rozšírení. Ak sa nastaví na *false* (čo je aj prednastavená hodnota), editor sa vytvorí s funkcionalitou definovanou vo vlastnosti *buttonList*.
- *buttonList* - Definuje zoznam (pole) tlačidiel, ktoré sa zobrazia v užívateľskom rozhraní editora. Napr.: *['bold','italic','underline','left','center']*.
- *iconsPath* - Definuje adresu k súboru, ktorý obsahuje ikony pre tlačidlá.
- *maxHeight* - Maximálna výška editora v pixeloch.
- *externalCSS* - Relatívna cesta k extérnemu súboru definujúcemu CSS štýly.
- *uploadURI* - Cesta k skriptu, ktorý zabezpečí nahrávanie (anglicky upload) obrázkov z editora.

NicEdit je možné inicializovať rôznymi spôsobmi. Použitím následujúceho kódu dôjde ku konvertovaniu všetkých prvkov TEXTAREA na editor NicEdit:

```
bkLib.onDomLoaded(function() { nicEditors.allTextAreas() });
```

Ďalší príklad zobrazuje vytvorenie troch editorov, kde každému sú predané iné konfiguračné parametre [15]:

```
bkLib.onDomLoaded(function() {
new nicEditor().panelInstance('area1');
```

```
new nicEditor({fullPanel : true}).panelInstance('area2');
new nicEditor({maxHeight : 100}).panelInstance('area3');
});
```

Údaje *area1, area2 a area3* predstavujú atribúty ID prvkov, ktoré budú konvertované na editor NicEdit.

3.1.3 Tvorba rozšírení

3.1.3.1 Príklad vytvorenia modulu

Vytváranie rozšírení pre NicEdit je vskutku jednoduché. Základný modul je možné vytvoriť podľa následujúceho postupu [15]:

1. Je potrebné obstarať si verziu editora NicEdit určenú pre vývoj. Táto verzia, narozdiel od štandardnej, obsahuje jednotlivé moduly separované do osobitných súborov a zatriedené do pevne danej adresárovej štruktúry.
2. V adresáry s názvom "src" vytvoríme nový adresár podľa názvu vytváraného rozšírenia, napr. "*nicExample*" a v ňom vytvoríme súbor "*nicExample.js*". Názov adresára a súboru pre zdrojový kód musia byť rovnaké.
3. Do zdrojového súboru skopírujeme kód uvedený v prílohe D.

Zdrojový kód z prílohy D je možné rozdeliť na niekoľko častí:

- Hlavička (riadky 1 až 7) - Špeciálny text uvedený v komentároch, ktorý sa však využíva systémom na sťahovanie editora NicEdit s navolenými rozšíreniami. V texte sú uvedené parametre ako názov rozšírenia, popis, závislosti na iných rozšíreniach, autor a verzia.
- Konfigurácia (riadky 9 až 15) - Nastavenie vlastností nového rozšírenia.
- Vlastná funkcionalita rozšírenia (riadky 17 až 20) - V ukážkovom príklade ide o vytvorenie funkcie, ktorá bude obsluhovať udalosti kliknutia myši.
- Registrácia (riadok 23) - Na záver sa vytvorené rozšírenie musí zaregistrovať do zoznamu aktívnych rozšírení.

3.1.3.2 NicEdit API

NicEdit poskytuje jednoduché API pre prácu s editormi. Základom API sú tri objekty:

- *nicEditors* - Poskytuje metódu pre konvertovanie všetkých prvkov TEXTAREA na editory, umožňuje prístup (získanie inštancie) editora podľa ID prvku v dokumente a udržuje zoznam všetkých aktívnych inštancií editora.
- *nicInstance* - Umožňuje prístup k obsahu danej inštancie editora.
- *nicEditor* - Objekt, ktorý umožňuje v dokumente vytvoriť viacero editovateľných oblastí, ktoré je možné ovládať jedným ovládacím panelom.

3.1.3.3 NicEdit udalosti

Pri tvorbe rozšírení pre NicEdit je možné pracovať aj s niekoľkými udalosťami, ktoré generuje jadro editora. Konkrétne ide o [15]:

- *blur* - Udalosť, ktorá sa posiela, ak editor stratí zameranie (anglicky focus).
- *focus* - Posiela sa pri získaní zamerania (napr. kliknutie myšou na editor).
- *key* - Udalosť vzniká pri stlačení ovládacích kláves (napr. Ctrl + B).

- *add* - Posiela sa pri pridaní novej inštancie editora.
- *panel* - Vzniká po vytvorení ovládacieho panelu.

Na prepojenie funkcií s udalosťami generovanými jadrom editora sa využíva metóda *addEvent* objektu *nicInstance*.

3.1.4 Licencia a podpora prehliadačov

NicEdit je možné bezplatne používať na ľubovoľné účely. Autor Brian Kirchoff ho vydal pod licenciou MIT.

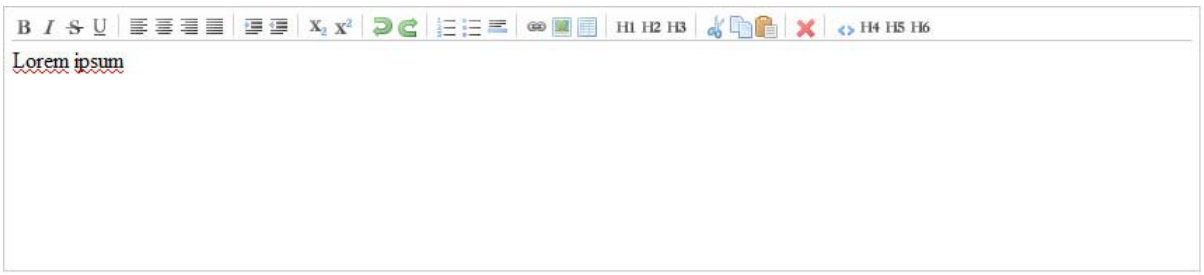
Editor by mal bez akýchkoľvek problémov fungovať v prehliadačoch:

- Internet Explorer 5.5+.
- Firefox 2+.
- Opera 9+.
- Safari 3+.

3.2 jWYSIWYG

Ďalším jednoduchým editorom je jWYSIWYG [33], ktorý je postavený na knižnici jQuery (kompatibilný s verziou jQuery 1.3.2 a vyššou). Aktuálnou verziou v dobe písania tejto práce bola jWYSIWYG 0.94 z 10. decembra 2010.

Na obrázku č. 3 je ukážka vzhľadu editora jWYSIWYG.



Obrázok 3: Ukážka vzhľadu editora jWYSIWYG [33].

3.2.1 Štruktúra a rozsah

Výhodou editora jWYSIWYG je jeho veľkosť, ktorá má pri poslednej verzii menej ako 26kB (vrátane obrázkov uživateľského rozhrania). Editor však pre svoju činnosť potrebuje knižnicu jQuery [14], ktorá má v minimalizovanej a komprimovanej verzii veľkosť 26kB.

Celý editor tvorí [33]:

- Zdrojový súbor jadra editora.
- Zdrojové súbory pre ovládacie prvky.
- Zdrojové súbory pre rozšírenia.
- Súbory s jazykovými prekladmi.
- Kaskádové štýly.
- Obrázky - ikony užívateľského rozhrania.

Editor umožňuje nahradzovať prvky TEXTAREA plnohodnotným prostredím pre úpravu obsahu. Jeden dokument môže obsahovať viacero editorov súčasne.

3.2.2 Konfigurácia

Editor jWYSIWYG je možné konfigurovať predaním konfiguračného objektu metóde *wysiwyg*, ako ukazuje následujúci zápis:

```
$('#TEXTAREA_ID').wysiwyg({KONFIG_OBJEKT});
```

Konfiguračný objekt má tvar:

```
{vlastnosť_01:hodnota, vlastnosť_02:hodnota, ...}
```

Medzi dostupné konfiguračné vlastnosti patria [33]:

- *html* - Reťazec obsahujúci kód v jazyku HTML, ktorý sa zobrazí v editore.
- *debug* - Nastavenie špecifikujúce ladenie chýb. Povolené hodnoty sú *true* alebo *false*.
- *css* - Nastavenie cesty k súboru definujúcemu štýly.
- *autoGrow* - Udáva, či sa editor bude svojou výškou prispôsobovať rozsahu textu, alebo sa zobrazí posuvník (anglicky scrollbar). Povolené hodnoty sú *true* alebo *false*.
- *autoSave* - Špecifikuje možnosť automatického kopírovania obsahu editora do pôvodného prvku TEXTAREA, ktorý editor nahradil. Povolené hodnoty sú *true* alebo *false*.
- *brIE* - Pri nastavení na hodnotu *true* sa v prehliadačoch Internet Explorer bude ako označenie nového riadku vkladať `<br/>`.
- *formHeight* - Číselná hodnota udávajúca výšku okna.
- *formWidth* - Číselná hodnota udávajúca šírku okna.
- *i18n* - Špecifikovanie využitia lokalizácie.
- *initialContent* - Reťazec špecifikujúci obsah editora po inicializácii.
- *maxHeight* - Udáva maximálnu vyšku, ktorú editor nesmie prekročiť pri aktivovaní vlastnosti *autoGrow*.
- *messages* - Objekt v tvare `{kľúč01:hodnota01, kľúč02:hodnota02}`, ktorý umožňuje špecifikovať vlastné hodnoty pre konkrétne udalosti (definované kľúčom). Momentálne je jediným dostupným kľúčom *nonSelection*, ktorý určuje text správy, ktorá sa zobrazí pri vkladaní odkazu do textu, ak užívateľ neoznačil do výberu žiadny text.
- *resizeOptions* - Nastavenie určujúce možnosť zmeny rozmerov editora užívateľmi.
- *rmUnusedControls* - Pri nastavení na hodnotu *true* sa v editore zobrazia iba tie ovládacie prvky, ktoré sú definované vo vlastnosti *controls*.
- *rmUnwantedBr* - Pri nastavení na hodnotu *true* nebude editor pridávať k obsahu nadbytočné značky `<br/>`.
- *tableFiller* - Textový reťazec, prednastavená hodnota je *Lorem ipsum*.
- *events* - Objekt špecifikujúci reakcie na udalosti. Udáva sa v tvare `{ udalosť01: funkcia01, udalosť02: funkcia02}`.
- *controls* - Objekt, ktorý špecifikuje ovládacie prvky, ktoré sa zobrazia v užívateľskom rozhraní editora. Momentálne je k dispozícii 33 vstavaných ovládacích prvkov,

ako napr. *bold*, *italic*, *cut*, *copy*, *paste*, a pod. Kompletný popis ovládacích prvkov nie je predmetom tejto práce a je možné ho nájsť na [33].

Následujúci príklad demonštruje možnosť vytvorenia inštancie editora jWYSIWYG s predaním konfiguračného objektu do metódy *wysiwyg*:

```
<script type="text/javascript">
$(function() {
$('#wysiwyg').wysiwyg({controls:{strikeThrough:{visibile: true}}});
});
</script>
<textarea id="wysiwyg"></textarea>
```

3.2.3 Tvorba rozšírení

3.2.3.1 Príklad vytvorenia modulu

Editor jWYSIWYG poskytuje možnosti vytvárania a integrácie nových rozšírení priamo do editora. Následujúci príklad demonštruje možnosť vytvorenia jednoduchého ovládacieho prvku [33]:

```
$('#wysiwyg').wysiwyg("addControl",
   "controlName",
   {
       icon: "/path/to/icon.png",
       exec:  function() { alert('Hello World'); }
   }
);
```

Uvedený príklad využíva možnosti jWYSIWYG API, konkrétne volanie *addControl*, ktoré slúži na pridanie nového ovládacieho prvku do užívateľského rozhrania. Následne je definovaný názov ovládacieho prvku (*controlName*), špecifikovaná cesta k ikone a funkcia, ktorá sa zavolá pri kliknutí na tento prvok.

3.2.3.2 jWYSIWYG API

Editor poskytuje aj jednoduché API pre prístup k obsahu, jeho modifikáciu, obsluhu udalostí a pod. Základ API tvorí objekt wysiwyg, ktorý poskytuje následujúce metódy [33]:

- *addControl* - Pridanie ovládacieho prvku do užívateľského rozhrania editora.
- *clear* - Nastavenie obsahu editora na prázdny reťazec.
- *createLink* - Vytvorenie hypertextového odkazu do textu.
- *destroy* - Zrušenie editora.
- *document* - Získanie odkazu na dokument.
- *getContent* - Získanie obsahu editora.
- *init* - Inicializácia editora.
- *insertHtml* - Vloženie kódu HTML na aktuálnu pozíciu kurzora.
- *insertImage* - Vloženie obrázka na aktuálnu pozíciu kurzora.
- *insertTable* - Vloženie tabuľky na aktuálnu pozíciu kurzora.
- *removeFormat* - Odstránenie formátovania označenému textu.

- *save* - Skopírovanie obsahu editora do pôvodného prvku TEXTAREA, ktorý editor nahradil.
- *setContent* - Nastavenie nového obsahu editora.

3.2.3.3 jWYSIWYG udalosti

Pri tvorbe rozšírení pre jWYSIWYG je možné pracovať aj s udalosťami. Nasledujúci príklad demonštruje vytvorenie funkcie, ktorá sa spustí vždy ako reakcia na udalosť kliknutia myši na obsah editora:

```
$('#wysiwyg').wysiwyg('document').click(function(e)
{
  e.preventDefault();
  alert('Klikli ste na obsah editora jWYSIWYG!');
});
```

Kompletný zoznam a popis udalostí nie je predmetom tejto práce a je dostupný na [34].

3.2.4 Licencia a podpora prehliadačov

Editor jWYSIWYG je licencovaný pod MIT a GPL. Editor by mal bez akýchkoľvek problémov fungovať v prehliadačoch:

- Internet Explorer 6+.
- Firefox 2+.
- Opera 9+.
- Safari 3+.
- Chrome 1+.

3.3 TinyMCE

TinyMCE [35] je jedným z najpoužívanejších WYSIWYG textových editorov napísaných v jazyku JavaScript. Ide o rozsiahly systém využívaný v mnohých webových aplikáciách. Poslednou stabilnou verziou bola počas písania tejto práce TinyMCE 3.3.9.3.

Na obrázku č. 4 je ukážka vzhľadu editora TinyMCE.

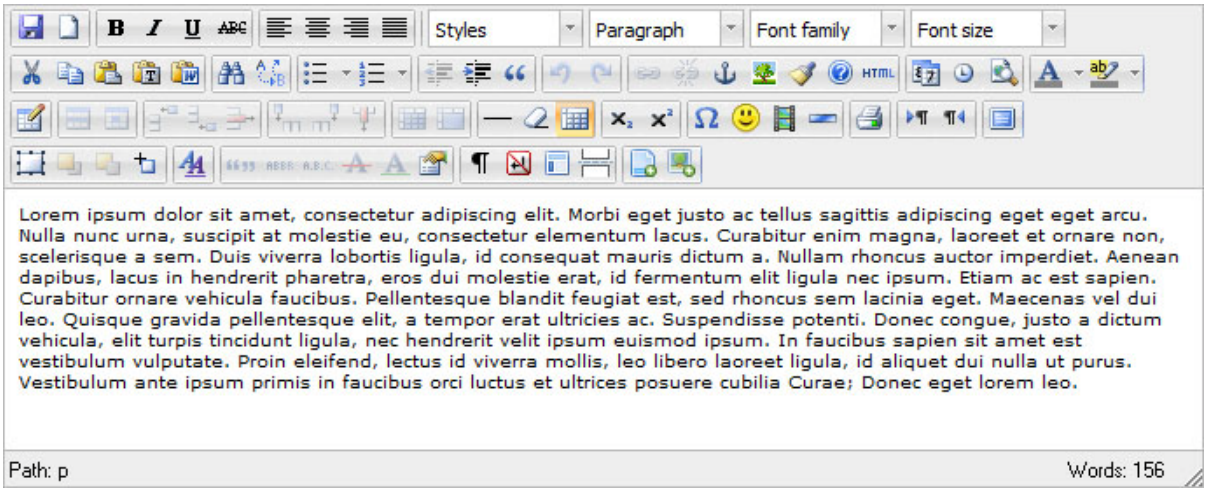
3.3.1 Štruktúra a rozsah

Veľkosť editora TinyMCE so všetkými zdrojovými súbormi, štýlmi, obrázkami a rozšíreniami v štandardnej edícii je 1508kB, no pri využití komprimovacích skriptov na strane serveru je možné túto veľkosť znížiť až o 75% [35].

Editor je tvorený [35]:

- Zdrojovými súbormi jadra editora.
- Rozšíreniami.
- Motívmi vzhľadu (anglicky Themes).
- Jazykovými lokalizáciami.
- Doplnkovými skriptami (anglicky Utilities).

TinyMCE umožňuje nahradzovať prvky TEXTAREA (a iné HTML elementy) plnohodnotným prostredím pre úpravu obsahu. Jeden dokument môže obsahovať viacero editorov súčasne.



Obrázok 4: Ukážka vzhľadu editora TinyMCE [35].

3.3.2 Konfigurácia

Konfigurácia editora TinyMCE je možná predaním konfiguračného objektu metóde *init* objektu *tinyMCE*, ako to ukazuje následujúci zápis:

```
tinyMCE.init({KONFIG_OBJEKT});
```

Konfiguračný objekt má tvar:

```
{vlastnosť_01:hodnota, vlastnosť_02:hodnota, ...}
```

TinyMCE ponúka viac ako 130 konfiguračných vlastností, ktoré sú kategorizované do skupín [36]:

- Všeobecné (anglicky General).
- Reakcie na udalosti (anglicky Callbacks).
- Výstup a úpravy (anglicky Cleanup / Output).
- URL.
- Štruktúra a vzhľad (anglicky Layout).
- Vizuálne pomôcky (anglicky Visual aids).
- Späť / Opakovať (angliocky Undo / Redo).
- Zoznamy súborov (anglicky File lists).
- Špúšťače a opravy (anglicky Triggers / Patches).
- Pokročilé nastavenia motívu vzhľadu (anglicky Advanced theme).

Podrobný popis všetkých konfiguračných nastavení je dostupný na [36].

Následujúci príklad demonštruje možnosť vytvorenia editora TinyMCE s predaním konfiguračného objektu metóde *init*:

```
<script type="text/javascript">
tinyMCE.init({
        mode : "textareas", theme : "simple"
});
</script>
```

Uvedený príklad vytvorí zo všetkých prvkov *TEXTAREA* editory TinyMCE s motívom vzhľadu *simple*.

3.3.3 Tvorba rozšírení

Vytváranie nových rozšírení pre TinyMCE je aj napriek jeho robustnosti veľmi jednoduché.
Pre začiatok je potrebné obstarať si verziu editora TinyMCE určenú pre vývoj.

Adresárovú štruktúru základného rozšírenia tvorí [37]:

- Adresár */css*, ktorý obsahuje špecifické súbory definujúce štýly.
- Adresár */img* s obrázkami a ikonami.
- Adresár */js* obsahujúci súbory so zdrojovým kódom v jazyku JavaScript.
- Adresár */langs*, ktorý obsahuje jazykové preklady.
- Súbor */editor_plugin.js* - Komprimovaný zdrojový kód rozšírenia.
- Súbor */editor_plugin_src.js* - Zdrojový kód rozšírenia.
- Súbor */dialog.htm* - HTML kód dialógového okna.

Súbor *editor_plugin_src.js* je jadrom editora TinyMCE ignorovaný. Obsahuje totiž prehľadne formátovaný a komentovaný zdrojový kód, ktorý je síce dobre čitateľný pre programátora, no kvôli nižším nárokom na prenos sa uprednostňuje jeho komprimovaná podoba (súbor *editor_plugin.js*). Tú je možné vytvoriť napr. pomocou obfuskátora s názvom JavaScript compressor [38].

3.3.3.1 Príklad vytvorenia modulu

V prílohe E je uvedená ukážka zdrojového kódu jednoduchého rozšírenia. Kód pozostáva z týchto častí:

- Načítanie jazykových prekladov modulu (riadok 3).
- Definovanie metódy *init*, ktorá sa zavolá ihneď po vytvorení rozšírenia (riadok 7). V tomto prípade sa zaregistruje vlastný príkaz rozšírenia (anglicky command) s názvom *mceExample* (riadok 8), ktorý je možné zavolať pomocou kódu:

  ```
  tinyMCE.activeEditor.execCommand('mceExample');
  ```

  Po zavolaní sa otvorí dialógové okno s obsahom definovaným v súbore *dialog.htm*. Metóda *init* ďalej vytvára a registruje ukážkové tlačídlo s názvom *example*, ktoré po kliknutí spúšťa príkaz *mceExample* (riadok 20). Na riadku 26 je definovaná reakcia na udalosť *onNodeChange*.
- Definícia metódy *getInfo* (riadok 30), ktorá vracia základné informácie o rozšírení, ako napr. celý názov rozšírenia, meno autora, verzia, URL adresa autora a pod.
- Registrácia modulu (riadok 40).

3.3.3.2 TinyMCE API

API editora TinyMCE je rozsiahlé a pozostáva z:

- Menných priestorov (anglicky namespaces):
  - *tinymce* - Koreňový menný priestor, ktorý pozostáva z tried vzťahujúcich sa priamo k editoru.
  - *tinymce.dom* - Obsahuje triedy pre prácu s objektovým modelom dokumentu (DOM)
  - *tinymce.html* - Triedy tvoriace logiku syntaktického analyzátora a serializéra HTML.
  - *tinymce.ui* - Obsahuje rôzne prvky užívateľského rozhrania, ako napr. tlačidlá, zoznamy a pod.
  - *tinymce.util* - Rôzne pomocné triedy.
  - *tinymce.plugins* - Triedy rozšírení.
- Tried (anglicky classes):
  - *tinymce* - Funkcionalita jadra editora TinyMCE.
  - *tinyMCEPopup* - Trieda pre dialógové okná.
- Vlastností (anglicky properties):
  - *tinyMCE* - alternatívne meno pre *tinymce*, ktoré slúži pre spätnú kompatibilitu s verziami 2.x.

Kompletný popis jednotlivých tried TinyMCE API je uvedený na [39].

3.3.3.3 TinyMCE udalosti

Nová verzia TinyMCE 3.x (v porovnaní s verziou 2.x) priniesla okrem iného aj nový spôsob práce s udalosťami, ktorý je založený na návrhovom vzore pozorovateľ (anglicky Observer). To umožňuje jednoduchší spôsob ošetrovania udalostí, ktorý funguje na princípe:

```
tinyMCE.aktivnyEditor.udalost.add
```

Konkrétnu ukážku zobrazuje následujúci príklad:

```
tinyMCE.init({
   ...
   setup : function(ed) {
      ed.onClick.add(function(ed, e) {
          console.debug('Editor was clicked: ' + e.target.nodeName);
      });
   }
});
```

V tomto prípade sa pri kliknutí myšou na editor vypíše uvedený text do konzoly.
Kompletný zoznam a popis udalostí nie je predmetom tejto práce a je dostupný na [40].

3.3.4 Licencia a podpora prehliadačov

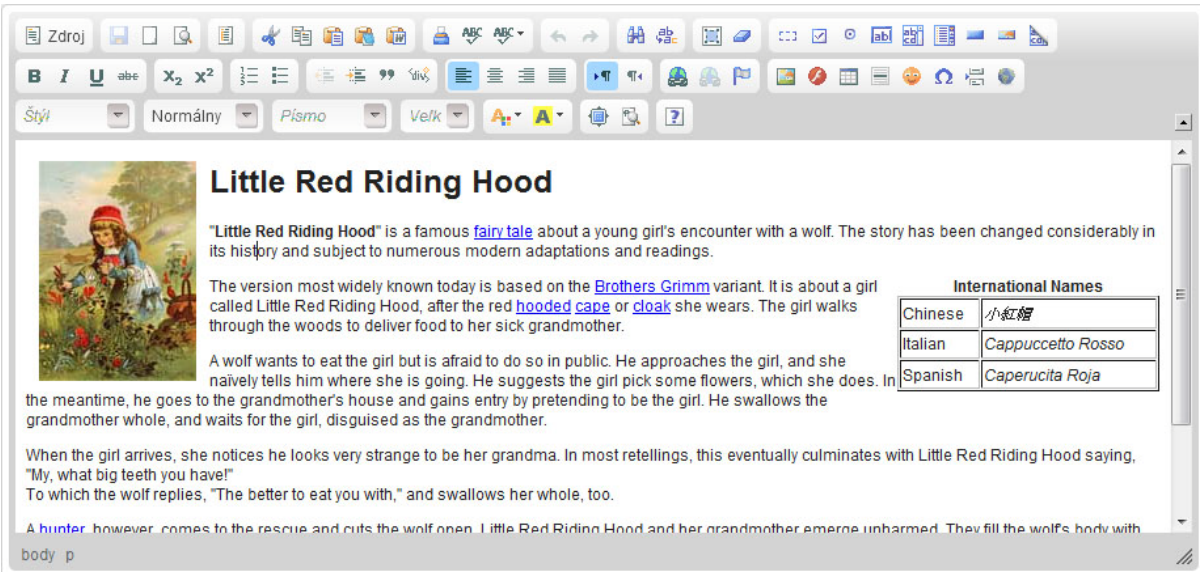
Editor TinyMCE je licencovaný pod LGPL a je plne funkčný v prehliadačoch:

- Internet Explorer 6+.
- Firefox 2+.
- Opera 10+.
- Safari 3+.
- Chrome 1+.

3.4 CKEditor

Ďalším rozsiahlým a často používaným editorom je CKEditor [41] s viac ako 3,5 miliónmi stiahnutí. Poslednou stabilnou verziou je CKEditor 3.5 z decembra 2010.

Na obrázku č. 5 je ukážka vzhľadu editora CKEditor.



Obrázok 5: Ukážka vzhľadu editora CKEditor [41].

3.4.1 Štruktúra a rozsah

CKEditor má so všetkými zdrojovými súbormi, štýlmi, obrázkami a rozšíreniami v štandardnej edícii viac ako 2MB.

Editor je tvorený [41]:

- Zdrojovými súbormi jadra editora.
- Rozšíreniami.
- Šablónami a Motívmi vzhľadu (anglicky Themes a Skins).
- Jazykovými lokalizáciami.
- Doplnkovými skriptami a súbormi.

CKEditor pracuje s prvkami *TEXTAREA* alebo *DIV* a v dokumente sa môže súčasne nachádzať viacero editorov súčasne.

3.4.2 Konfigurácia

Editor ponúka bohatú sadu konfiguračných nastavení, ktorými je možné meniť vzhľad, funkcie a celkové chovanie editora. Hlavným konfiguračným súborom je *config.js*, ktorý sa nachádza v koreňovom adresáry inštalácie editora. Jednou z možností konfigurácie je umiestnenie nastavení priamo do tohto súboru, ako ukazuje následujúci príklad [41]:

```
CKEDITOR.editorConfig = function( config ) {
     config.language = 'fr';
     config.uiColor = '#AADC6E';
};
```

V tomto príklade sa nastavil jazyk editora na francúzštinu a tiež sa pozmenila farba užívateľského rozhrania na uvedenú hodnotu.

Lepším spôsobom konfigurácie je umiestnenie nastavení priamo do metódy, ktorá vytvára nový editor, ide o tzv. nastavenie in-page. Pri tomto spôsobe nehrozia problémy pri migrácii na novšiu verziu editora.

Konfiguračný objekt sa pri in-page konfigurácii môže predať metódam:

```
CKEDITOR.replace('ID_or_ELEMENT_or_NAME', {KONFIG_OBJEKT});
```

alebo

```
CKEDITOR.appendTo('ID_or_ELEMENT', {KONFIG_OBJEKT}, DATA);
```

Obe tieto metódy vytvárajú novú inštanciu editora. Metóda *replace* príjma ako prvý argument ID, názov alebo odkaz na konkrétny prvok. Ako inicializačná hodnota sa pri prvku *TEXTAREA* použije hodnota atribútu *VALUE*, pri prvku *DIV* je to hodnota *innerHTML*. Na rozdiel od *replace*, metóda *appendTo* príjma ako prvý argument iba ID alebo odkaz na konkrétny prvok. Voliteľným argumentom je *DATA*, ktorý sa môže použiť na nastavenie inicializačnej hodnoty inštancie editora.

Konfigurácia in-page má vyššiu prioritu ako nastavenia v konfiguračnom súbore, a preto sa pri kolízii použije nastavenie z konfiguračného objektu predaného jednej zo spomínaných metód na vytváranie inštancií editora.

Konfiguračný objekt má tvar:

```
{vlastnosť_01:hodnota, vlastnosť_02:hodnota, ...}
```

CKEditor ponúka viac ako 130 konfiguračných vlastností. Podrobný popis všetkých nastavení je dostupný na [42].

Následujúci príklad demoštruje možnosť vytvorenia editora CKEditor s predaním konfiguračného objektu metóde *replace*:

```
CKEDITOR.replace( 'ELEMENT_ID',
    {
        language : 'fr',
        uiColor : '#AADC6E'
    });
```

Uvedený príklad zobrazuje spôsob vytvorenia novej inštancie editora CKEditor, ktorá nahradí prvok, ktorého id je *ELEMENT_ID*. Konfiguračné vlastnosti sú totožné s nastaveniami uvedenými v príklade konfigurácie pomocou súboru *config.js*.

3.4.3 Tvorba rozšírení

Tvorba rozšírení pre CKEditor taktiež dodržiava určité konvencie pri organizovaní adresárov a súborov. Každé rozšírenie má svoj vlastný adresár (podľa názvu rozšírenia) v adresáry *plugins*. V tomto adresáry sa nachádza súbor *plugin.js* (názov je pevne daný), ktorý obsahuje zdrojový kód rozšírenia. Rozšírenia často obsahujú aj adresár *dialogs*, v ktorom sa nachádzajú zdrojové kódy v HTML a jazyku JavaScirpt pre dialógové okná. Obrázky používané modulom sa zvyuknú ukladať do adresára *images*.

3.4.3.1 Príklad vytvorenia modulu

Pri vytváraní nového modulu je najprv potrebné tento modul zaregistrovať do konfiguračného súboru *config.js*, aby bol načítaný editorom CKEditor:

```
config.extraPlugins = nazovModulu;
```

Samotný kód pre vytvárané rozšírenie sa umiestní do súboru *plugin.js*. Základná štruktúra je jednoduchá, ako ukazuje následujúci príklad [43]:

```
CKEDITOR.plugins.add('nazovModulu',
{
    init: function(editor)
    {
        //miesto pre vlastný kód rozšírenia
    }
});
```

Po vytvorení modulu je posledným krokom kompresia zdrojového kódu. CKEditor využíva nástroj ckpackager.exe [44], ktorý slúži na kompresiu kódu a redukciu počtu zdrojových súborov (aby sa minimalizoval počet HTTP dotazov). Preto je potrebné pridať do súboru *ckeditor.pack* riadok

```
'_source/plugins/nazovModulu/plugin.js',
```

ktorý zabezpečí, aby sa do výsledku skomprimoval aj vytvorený modul. Komprimácia sa spúšťa príkazom:

ckpackager.exe ckeditor.pack

3.4.3.2 CKEditor API

API editora CKEditor je rozsiahlé a pozostáva z týchto menných priestorov (anglicky namespaces):

- *CKEDITOR* - Predstavuje jadro editora.
- *CKEDITOR.ajax* - Metódy pre prácu s AJAX.
- *CKEDITOR.config* - Udržiava všetky konfiguračné nastavenia.

- *CKEDITOR.dom* - Triedy pre prácu s objektovým modelom dokumentu.
- *CKEDITOR.dtd* - Udržiava objektovú reprezentáciu DTD.
- *CKEDITOR.env* - Informácie o prehliadači.
- *CKEDITOR.lang* - Funkcie súvisiace s jazykovými prekladmi.
- *CKEDITOR.loader* - Informácie o načítaných skriptoch.
- *CKEDITOR.plugins* - Práca s načítanými rozšíreniami.
- *CKEDITOR.resourceManager* - Udržiava informácie a metódy pre prácu so zdrojmi.
- *CKEDITOR.scriptLoader* - Metódy pre prácu s extérnymi skriptami.
- *CKEDITOR.skins* - Práca s motívmi vzhľadu.
- *CKEDITOR.themes* - Práca so šablónami.
- *CKEDITOR.tools* - Doplnkové nástroje.

Kompletný popis CKEditor API je uvedený na [45].

3.4.3.3 CKEditor udalosti

Práca s udalosťami v editore CKEditor funguje na princípe:

```
subscription = <object>.on( eventName, listenerFunction [,scopeObj]
[,listenerData] [,priority] ),
```

kde [46]:

- *eventName* - Predstavuje udalosť (anglicky event), na ktorú chceme reagovať.
- *listenerFunction* - Funkcia, ktorá sa zavolá na obsluhu udalosti.
- *scopeObj* (voliteľný) - Objekt, ktorý sa použije ako obor platnosti.
- *listenerData* (voliteľný) - Dáta, ktoré sa predajú funkcii na obsluhu udalosti.
- *priority* (voliteľný) - Slúži na nastavenie priority spúšťania funkcie na obsluhu udalosti v porovnaní s ďalšími funkciami, ktoré reagujú na danú udalosť.
- *subscription* - Hodnota, ktorá slúží na identifikáciu udalosti a reagujúcej funkcie a môže byť použitá na ich odstránenie, resp. odstránenie naslúchania na udalosť, ako ukazuje následujúci príklad:

```
Event.removeListener(subscription)
```

Kompletný zoznam a popis udalostí nie je predmetom tejto práce a je dostupný na [45].

3.4.4 Licencia a podpora prehliadačov

CKEditor je distribuovaný pod licenciami GPL, LGPL a MPL a je plne funkčný v prehliadačoch [41]:

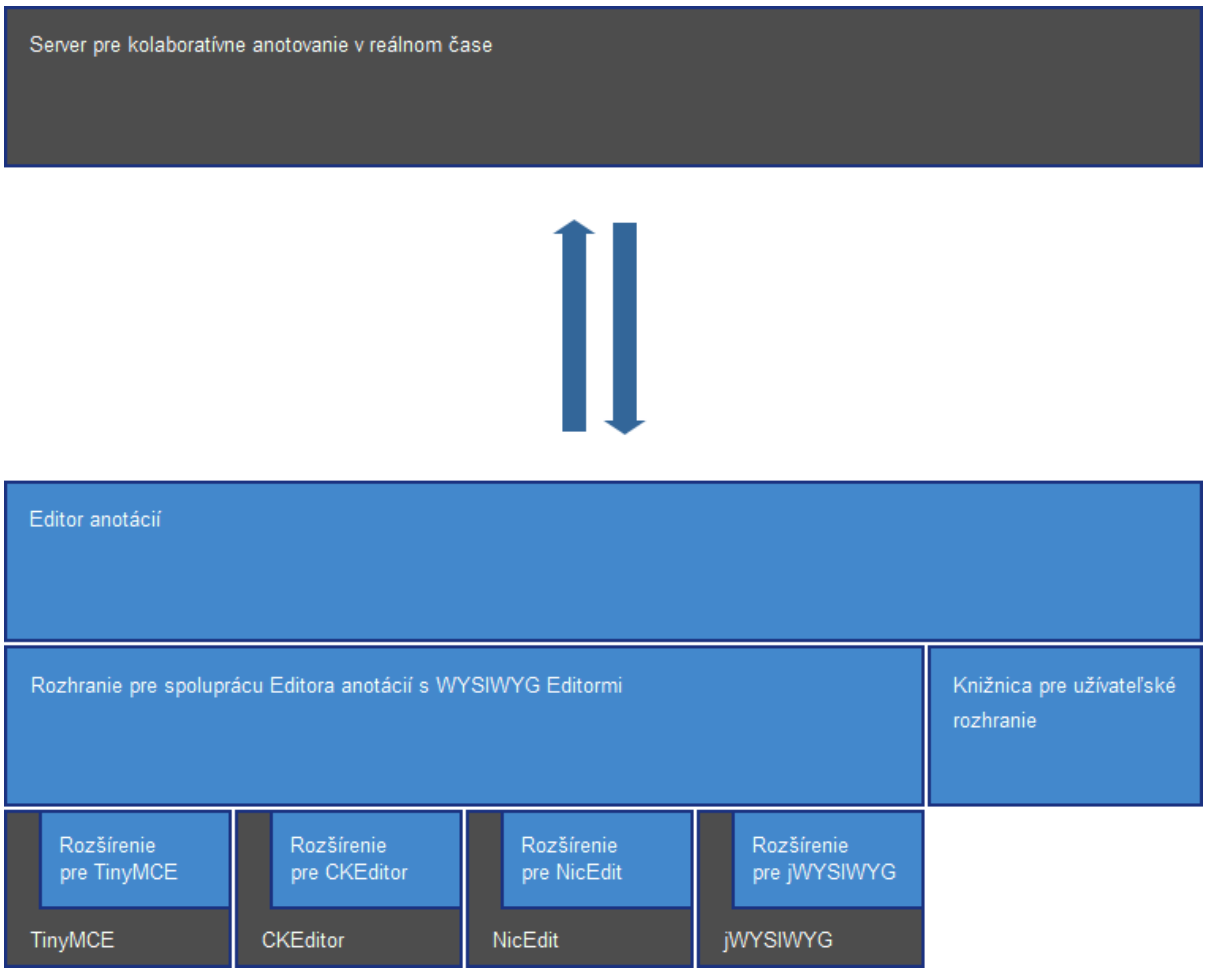
- Internet Explorer 6+.
- Firefox 3+.
- Opera.
- Safari 4+.
- Chrome.
- Camino 1+.

4 Návrh

Anotácia je doplňujúca informácia pridaná k dokumentu. Anotácie sa najčastejšie vyskytujú vo forme jednoduchých poznámok pridávaných k textu, no môžu doplňovať aj štrukturované informácie [47].

Editor anotácií je vyvýjaný ako súčasť rozsiahleho systému pre kolaboratívne anotovanie v reálnom čase, ktorý je predmetom dizertačnej práce vytváranej na FIT VUT [47]. V rámci rozsahu diplomovej práce bude Editor anotácií navrhnutý pre zvolené WYSIWYG editory TinyMCE, CKEditor, NicEdit a jWYSIWYG, pri návrhu sa však bude myslieť aj na jednoduchú rozšíriteľnosť pre iné typy editorov.

Editor anotácií bude dostupný v češtine, slovenčine a angličtine. Zároveň ho bude možné jednoduchým spôsobom lokalizovať aj do iných jazykov.

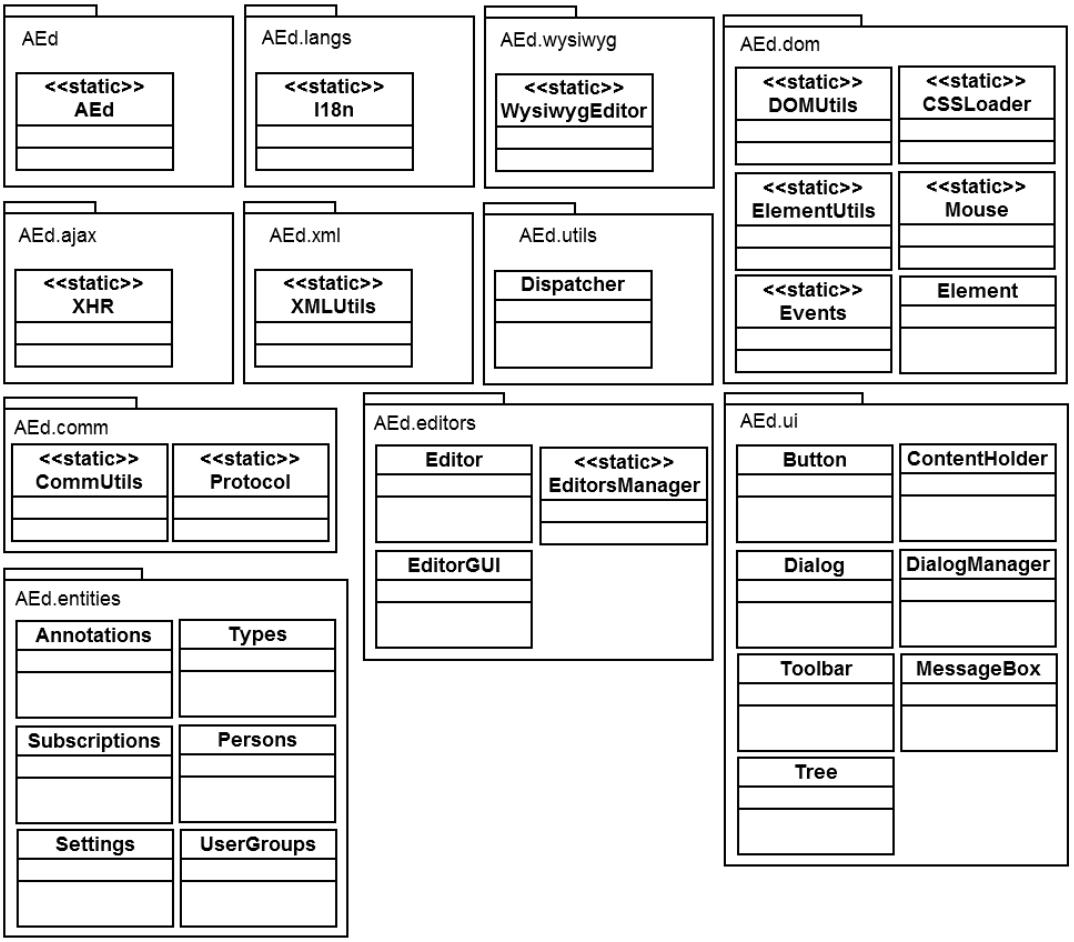


Obrázok 6: Štruktúra vytváraného systému.

Štruktúra vytváraného systému je zobrazená na obrázku č. 6, predmetom tejto práce je návrh a implementácia častí vyobrazených modrou farbou:

- *Editor anotácií* - Aplikácia, ktorá umožní anotovanie dokumentov v textových WYSIWYG editoroch.
- *Knižnica pre užívateľské rozhranie* - Vlastné riešenie bez použitia zložitých univerzálnych knižníc.

- *Rozhranie pre spoluprácu Editora anotácií s WYSIWYG editormi* - Umožní jednotný spôsob práce s rôznymi typmi WYSIWYG editorov.
- *Rozšírenia pre zvolené WYSIWYG editory* - Umožnia spúšťanie Editora anotácií z panela nástrojov každého zo zvolených WYSIWYG editorov.



Obrázok 7: Diagram balíčkov.

Obrázok č.7 zobrazuje diagram balíčkov editora anotácií, ktorý je pre väčšiu prehľadnosť vyobrazený bez závislostí. Podrobný popis jednotlivých tried sa nachádza v podkapitolách 4.1, 4.3 a 4.4.

4.1 Návrh editora anotácií

Táto podkapitola popisuje návrh samotnej aplikácie editora anotácií. Návrh sa sústreďuje hlavne na triedy a menné priestory a ich popis nemusí byť nutne úplný. Pri implementácií sa môžu rozšíriť alebo pozmeniť podľa problémov, s ktorými sa pri návrhu nerátalo.

4.1.1 Menné priestory

Menné priestory (anglicky namespaces) sú dôležitým prvkom pri vývoji aplikácií v jazyku JavaScript. Je dobrým programátorským zvykom použiť minimálne jeden menný priestor, ktorý poskytne kontext pre všetky triedy, funkcie a premenné využívané aplikáciou. V opačnom prípade by ľahko mohlo dôjsť ku kolízii identifikátorov s nejakými ďalšími skriptami, čo by mohlo mať kritické následky na funkčnosť vyvýjanej aplikácie.

Editor anotácií bude obsahovať hneď niekoľko menných priestorov, ktoré jednak poskytnú kontext a zároveň zvýšia prehľadnosť zdrojového kódu:

- *AEd* - Koreňový menný priestor, ktorého najdôležitejšou úlohou je poskytnutie kontextu pre všetky ostatné menné priestory, triedy, funkcie a premenné, ktoré budú využívané aplikáciou editora anotácií. Štýl programovania v jazyku JavaScript umožňuje, aby menný priestor *AEd* (ktorý bude implementovaný ako typ *Object* jazyka JavaScript) bol zároveň aj triedou. V našom prípade pôjde o singleton, ktorého návrh je v podkapitole 4.1.2.1.
- *AEd.dom* - Bude obsahovať triedy pre prácu objektovým modelom dokumentu (DOM), konkrétne: *CSSLoader*, *DOMUtils*, *Element*, *ElementUtils*, *Events* a *Mouse*. Podrobný návrh jednotlivých tried je v podkapitole 4.1.2.2.
- *AEd.editors* - Triedy súvisiace s funkcionalitou editora anotácií, správa viacerých inštancií editora a pod. (triedy *Editor*, *EditorsManager* a *EditorGUI* popísané v podkapitole 4.1.2.3.).
- *AEd.langs* - Bude združovať triedu potrebnú pre lokalizáciu editora anotácií: *I18n*, spolu s jazykovými prekladmi. Podrobný popis sa nachádza v podkapitole 4.1.2.4.
- *AEd.ui* - Menný priestor, ktorý bude obsahovať triedy knižnice pre užívateľské rozhranie. Návrhu knižnice sa detailne venuje podkapitola 4.3.
- *AEd.utils* - Bude obsahovať všetky triedy, ktoré poskytnú užitočnú funckionalitu pre editor anotácií a logicky ich nebude možné zaradiť do iného menného priestoru. V rámci rozsahu diplomovej práce bude *AEd.utils* obsahovať iba jednu triedu: *Dispatcher*, no pri ďalšom vývoji editora môže poskytnúť kontext aj pre ďalšie užitočné triedy. Návrh triedy *Dispatcher* sa nachádza v podkapitole 4.1.2.5.
- *AEd.ajax* - Bude obsahovať všetky triedy, ktoré budú súvisieť s asynchronnou komunikáciou, či už spôsobom AJAX alebo Comet. Detailnému popisu sa venuje podkapitola 4.1.2.6.
- *AEd.xml* - Združuje triedy, ktoré poskytnú funkcie pre prácu s XML a XPath. Podrobný popis sa nachádza v podkapitole 4.1.2.7.
- *AEd.wysiwyg* - Menný priestor, ktorý bude obsahovať triedy rozhrania pre spoluprácu WYSIWYG editorov s editorom anotácií. Návrh rozhrania je možné nájsť v podkapitole 4.4.
- *AEd.entities* - Bude združovať triedy, ktoré poskytnú metódy pre prácu so základnými entitami, s ktorými pracuje server pre kolaboratívne anotovanie. Konkrétne ide o triedy: *Annotations*, *Types*, *Subscriptions*, *Persons*, *UserGroups* a *Settings*. Ich návrh sa nachádza v podkapitole 4.1.2.8.
- *AEd.comm* - Menný priestor, ktorý bude obsahovať triedy umožňujúce komunikáciu so serverovou časťou. Podrobný popis sa nachádza v podkapitole 4.1.2.9.

4.1.2 Triedy

Táto podkapitola detailnejšie popisuje návrh jednotlivých tried a ich umiestnenie do menných priestorov.

4.1.2.1 Trieda AEd

Ako bolo uvedené v časti 4.1.1, *AEd* bude predstavovať ako koreňový menný priestor, tak i statickú triedu. Jej najdôležitejšími metódami budú:

- *init()* - Inicializácia aplikácie, jej argumentom je konfiguračný objekt.
- *createNamespace()* - Vytvorenie nového menného priestoru.
- *extendClass()* - Umožňuje vytvorenie triedy dedením vlastností a metód z inej triedy.
- *$()* - Funkcia, ktorá vytvorí inštanciu triedy *AEd.dom.Element*, príklad použitia sa nachádza v podkapitole 4.1.2.2.

Jedinou udalosťou, ktorú poskytne trieda *AEd*, bude *onDomReady*, ktorá spustí priradené funkcie na obsluhu v prípade, že štruktúra DOM bola pripravená.

4.1.2.2 Triedy v mennom priestore AEd.dom

Menný priestor *AEd.dom* bude obsahovať triedy pre prácu s objektovým modelom dokumentu (DOM), konkrétne: *CSSLoader*, *DOMUtils*, *ElementUtils*, *Element*, *Events* a *Mouse*.

Trieda *CSSLoader* bude slúžiť na dynamické načítanie extérneho súboru definujúceho CSS štýly. Najdôležitejšou metódou bude *load()*, ktorá načíta súbor zadaný ako parameter funkcie.

Trieda *DOMUtils* poskytne užitočné metódy, ktoré sa budú počas vývoja využívať častejšie. Navrhovanými metódami sú:

- *getClientWidth()* - Získanie šírky klientskej časti okna prehliadača.
- *getClientHeight()* - Získanie výšky klientskej časti okna prehliadača.
- *getScrollX()* - Získanie offsetu, ktorý určuje o koľko obrazových bodov je zobrazený dokument posunutý pomocou posuvníka (anglicky scrollbar) na x-ovej osi.
- *getScrollY()* - Podobne ako pri *getScrollX()*, ale vráti hodnotu offsetu na y-ovej osi.

Trieda *ElementUtils* bude obsahovať užitočné funkcie pre prácu s elementami. Medzi základné metódy budú patriť:

- *getCssValue()* - Získanie hodnoty zadaného štýlu CSS.
- *setCssValue()* - Nastavenie hodnoty štýlu CSS.
- *moveByX()* - Posunutie elementu o hodnotu na x-ovej osi.
- *moveByY()* - Posunutie elementu o hodnotu na y-ovej osi.
- *moveBy()* - Posunutie elementu o hodnotu na osiach x a y.
- *moveTo()* - Nastavenie pozície elementu na zadané hodnoty x a y.
- *getWidth()* - Získanie šírky elementu.
- *getHeight()* - Získanie výšky elementu.
- *hide()* - Ukryje element.
- *show()*- Zobrazí skrytý element.
- *setOpacity()* - Nastaví priehľadnosť elementu.
- *getAllClasses()* - Vráti pole všetkých názvov tried priradených elementu.
- *addClass()* - Pridá elementu novú triedu.
- *removeClass()* - Odoberie elementu zadanú triedu.

- *hasClass()* - Vráti hodnotu *true* v prípade, že element obsahuje triedu so zadaným názvom.

Trieda *Element* bude plniť rovnakú úlohu ako trieda *ElementUtils*, rozdiel však bude v spôsobe zápisu. Pre ilustráciu uvažujme, že chceme skryť element, ktorého id je: *id_elementu*. Pomocou triedy *ElementUtils* by to bolo možné týmto spôsobom:

```
var element = document.getElementById("id_elementu");
var eu = AEd.dom.ElementUtils;
eu.hide(element);
```

Využitím triedy *Element* bude skrývanie elementu vyzerať takto:

```
var element = new AEd.dom.Element("id_elementu");
element.hide();
```

Triedu *Element* bude využívať metóda *AEd.$()*, ktorá bude slúžiť na skrátenie kódu pri práci s elementami DOM. Predchádzajúci príklad by pomocou metódy *AEd.$()* vyzeral takto:

```
AEd.$("id_elementu").hide();
```

Ďalšou triedou v mennom priesotre *AEd.dom* bude trieda *Events*, ktorej hlavnou úlohou bude poskytnutie metód pre prácu s udalosťami, ktoré budú fungovať vo väčšine prehliadačov:

- *addHandler() / removeHandler()* - Priradí / Odoberie elementu obsluhu udalosti.
- *getEvent()* - Získanie objektu udalosti.
- *getTarget()* - Vráti element, na ktorom sa udalosť vyskytla.
- *preventDefault()* - Zabraní implicitnej akcii vo vykonaní.
- *stopPropagation()* - Zastaví prebublávanie udalosti cez štruktúru DOM.

Posledná trieda v mennom priestore *AEd.dom* bude *Mouse*, ktorá bude obsahovať iba dve metódy:

- *getAbsMouseX() / getAbsMouseY()* - Vráti pozíciu myši na osi x / y.

4.1.2.3 Triedy v mennom priestore AEd.editors

V mennom priestore *AEd.editors* budú implementované tri triedy: *Editor*, *EditorManager* a *EditorGUI*. Inštancie triedy *Editor* budú predstavovať samotný editor anotácií. Medzi ich základné vlastnosti budú patriť:

- *gui* - Každá inštancia editora anotácií má vlastné grafické užívateľské rozhranie (inštancia triedy *EditorGUI*).
- *wysiwygEditor* - Každá inštancia editora anotácií je tiež spojená s konkrétnym wysiwyg textovým editorom.

Medzi navrhované metódy triedy *Editor* patria:

- *render() / destroy()* - Vykreslenie / odstránenie editora.
- *show() / hide()* - Zobrazenie / skrytie editora.

Trieda *EditorsManager* bude slúžiť na správu viacerých inštancií editora anotácií (trieda *Editor*). Bude obsahovať vlastnosť *editors* - kolekciu všetkých inštancií editora a tiež vlastnosť *activeEditor* - ukazateľ na aktívnu inštanciu. Základnými metódami budú *add()* a *remove()* slúžiace na pridávanie / odoberanie inštancií editorov.

Úlohou triedy *EditorGUI* bude vyvorenie grafického užívateľského rozhrania editora anotacií. Bude obsahovať dve metódy: *render()* a *destroy()* na vykreslenie / odstránenie všetkých prvkov UI.

4.1.2.4 Triedy v mennom priestore AEd.langs

Menný priestor *AEd.langs* bude obsahovať triedu *I18n* zaisťujúcu preklad jazykových reťazcov. Trieda *I18n* bude obsahovať dve metódy:

- *getLang()* - Vráti aktuálne nastavenie jazyka.
- *translate()* - Preloží zadaný reťazec podľa aktuálne nastaveného jazyka.

4.1.2.5 Triedy v mennom priestore AEd.utils

Menný priestor *AEd.utils* bude obsahovať iba jednu triedu: *Dispatcher*. Táto trieda bude slúžiť na vytváranie a obsluhu vlastných udalostí pre editor anotácií. Navrhnutými metódami sú:

- *addHandler() / removeHandler()* - Pridanie / odobranie funkcie na obsluhu udalosti.
- *fire()* - Vyvolanie udalosti.

4.1.2.6 Triedy v mennom priestore AEd.ajax

Menný priestor *AEd.ajax* bude obsahovať triedu *XHR*. Jedinou verejnou metódou tejto triedy bude *sendRequest()*, ktorej úlohou bude odosielanie dotazov na server. Metóde sa predá objekt definujúci parametre dotazu, ktorý bude obsahovať:

- *method* - Typ dotazu (GET alebo POST).
- *url* - URL serveru.
- *data* - Dáta, ktoré sa majú odoslať.
- *requestHeaders* - Nastavenie hlaviček.
- *onSuccess* - Funkcia obsluhy v prípade úspešného príchodu odpovede so stavovým kódom z intervalu <200, 300).
- *onNotModified* - Funkcia obsluhy v prípade úspešného príchodu odpovede so stavovým kódom 304.
- *onFailure* - Funkcia obsluhy v prípade neúspechu.
- *scope* - Kontext, v ktorom sa funkcie obsluhy majú spustiť.

4.1.2.7 Triedy v mennom priestore AEd.xml

V mennom priestore *AEd.xml* bude implementovaná trieda *XMLUtils*, ktorá bude obsahovať metódy:

- *xmlParser()* - Vytvorí analýzator kódu jazyka XML.
- *getElementXPath()* - Vráti XPath zadaného elementu.
- *xmlToString()* - Serializuje kód XML a vráti ho ako reťazec.

4.1.2.8 Triedy v mennom priestore AEd.entities

Menný priestor *AEd.entities* združuje triedy, ktoré poskytnú metódy pre prácu s entitami, s ktorými pracuje server pre kolaboratívne anotovanie. Konkrétne ide o triedy: *Annotations*, *Types*, *Subscriptions*, *Persons*, *UserGroups* a *Settings*. Každá z týchto tried bude obsahovať vlastné metódy pre pridanie a odstránenie objektu danej entity do kolekcie všetkých objektov toho typu.

4.1.2.9 Triedy v mennom priestore AEd.comm

V mennom priestore *AEd.comm* budú implementované dve triedy: *CommUtils* a *Protocol*. Trieda *CommUtils* poskytne metódy *sendPostRequest()* a *sendGetRequest()* na posielanie asynchrónnych dotazov. Táto trieda bude taktiež spracovávať serverové správy a to vyvolaním príslušnej udalosti podľa typu správy ktorá dorazila. Podľa protokolu pre komunikáciu so serverom (popísaným v podkapitole 4.2) budú vytvorené tieto typy udalostí:

- *onServerMsgConnected.*
- *onServerMsgDisconnect.*
- *onServerMsgLogged.*
- *onServerMsgPersons.*
- *onServerMsgUserGroups.*
- *onServerMsgResynchronize.*
- *onServerMsgSynchronized.*
- *onServerMsgTextModification.*
- *onServerMsgTypes.*
- *onServerMsgAnnotations.*
- *onServerMsgSuggestions.*
- *onServerMsgSettings.*
- *onServerMsgError.*
- *onServerMsgWarning.*
- *onServerMsgOk.*
- *onServerMsgUnknown.*

Typy správ, pri ktorých sa jednotlivé udalosti vyvolajú, sú zrejmé z ich názvu. Ďalšou triedou v mennom priestore *AEd.comm* bude *Protocol*. Táto trieda poskytne metódy na vytvorenie alebo analýzu správ protokolu pre komunikáciu so serverom. Konkrétne pôjde o metódy s názvami *createMsg()*, *parseServerMsgFromString()* a *parseServerMsgFromXml()*.

4.2 Protokol pre komunikáciu so serverom

Editor anotácií bude obojstranne asýnchronne komunikovať so serverom. Princípy tohoto spôsobu komunikácie, označované ako Comet, sú popísané v podkapitole 2.8.

Protokol pre komunikáciu so serverom je popísaný v prílohe B a je prevzatý z [47]. Jednotlivé správy budú posielané vo formáte XML a môžu byť podľa potreby združované (je možné zaslať viacero správ v jednom XML). Správy môžeme kategorizovať do týchto skupín [47]:

- Správa sedenia.
- Užívatelia a skupiny.
- Riadenie odberu anotácií.
- Synchronizácia dokumentu.
- Prenos typov anotácií.
- Prenos anotácií.
- Ponúkanie anotácií.
- Prenos nastavení.
- Chyby a upozornenia.
- Prázdna odpoveď.

Formát anotácie je možné nájsť v prílohe A. Detailný popis jednotlivých správ sa nachádza v prílohe B. Príloha C obsahuje zjednodušený príklad komunikácie medzi klientom a serverom.

4.3 Návrh knižnice pre užívateľské rozhranie

Knižnica bude obsahovať základné komponenty užívateľského rozhrania, ktoré bude vyžadovať editor anotácií. Rozhranie bude implementované bez použitia zložitých univerzálnych knižníc, čo je jedna z požiadavok zo zadania tejto práce.

Medzi navrhované prvky patria:

- *Dialog* - Trieda, ktorá umožní vytváranie dialógových okien.
- *DialogManager* - Zodpovedný za správu okien, bude evidovať všetky otvorené dialógové okná, zaistí prenos aktívneho okna do popredia a pod.
- *ContentHolder* - Komponenta, ktorá predstavuje abstraktný kontainer, do ktorého je možné vložiť ľubovoľný obsah.
- *Toolbar* - Predstavuje hlavný panel nástrojov editora anotácií. *Toolbar* bude obsahovať tlačítka pre ovládanie všetkých funkcií editora.
- *Button* - Trieda, ktorá umožní vytvoriť tlačítka a priradzovať im akcie pri určitých udalostiach, ako napr. kliknutie myšou.
- *MessageBox* - Trieda, ktorá zjednoduchší vytváranie správ, ktoré sa zobrazia užívateľovi.
- *Tree* - Komponenta vytvárajúca rozbaľovací strom prvkov.

Všetky prvky užívateľského rozhrania budú vytvorené tak, aby bolo možné ich vzhľad upraviť jednoduchou zmenou motívu vzhľadu v konfiguračnom objekte editora anotácií. Motívom vzhľadu sa podrobne venuje podkapitola č. 4.3.8.

4.3.1 Dialog

Dialog je trieda slúžiaca na vytváranie dialógových okien. Pri vytváraní inštancie sa konštruktoru predá konfiguračný objekt, pomocou ktorého bude možné prispôsobiť vzhľad a chovanie vytváraného okna.

Medzi základné konfiguračné nastavenia budú patriť:

- *title* - Text, ktorý sa zobrazí v záhlaví okna.
- *width* - Nastavenie šírky okna.
- *height* - Nastavenie výšky okna.
- *corners* - Nastavenie umožnujúce špecifikovať, ktoré okraje dialógového okna budú zaoblené.
- *draggable* - Určuje, či dané dialógové okno bude možné presúvať pomocou myši na inú polohu.
- *resizeable* - Určuje, či danému dialógovému oknu bude možné meniť rozmery pomocou myši.
- *render* - Nastavenie, ktoré špecifikuje, či sa má okno po vytvorení inštancie ihneď vykresliť.
- *center* - Udáva, či sa má okno po vykreslení vycentrovať na stred obrazovky.
- *showOverlay* - Nastavenie, ktoré určuje, či sa pod dialógovým oknom má vykresliť vrstva na celé rozmery klientského okna, ktorá zakryje všetky elementy pod dialógovým oknom.

- *targetElement* - DOM Element, do ktorého sa dialógové okno vykreslí.
- *contentElement* - DOM Element, ktorý sa použije ako hlavný obsah dialógového okna.
- *allwaysOnTop* - Určuje, či bude dialógové okno vždy nad ostatnými oknami.
- *trackMouse* - Určuje, či sa bude poloha okna meniť spolu so zmenou polohy kurzora myši.
- *autoHide* - Nastavenie, ktoré umožňuje automatické odstránenie dialógového okna po uplynutí zadaného časového intervalu. Ide o objekt, ktorý pozostáva z:
    - *autoHide.delay* - dlžka zobrazenia dialógového okna v milisekundách.
    - *autoHide.callback* - Funkcia, ktorá sa vykoná po odstránení okna.
    - *autoHide.callbackScope* - Obor platnosti, v ktorom sa callback funkcia spustí.

Všetky uvedené konfiguračné nastavenia sú voliteľné, pri ich neuvedení sa použijú implicitné hodnoty.

Pri práci s dialógovými oknami bude možné využiť aj udalosti, ktoré generujú. Pôjde konkrétne o:

- *onClick* - Pri kliknutí myšou na dialógové okno.
- *onRendered* - Pri pridaní okna do stromu DOM.
- *onBeforeRemove* - Pred odstránením okna zo stromu DOM.
- *onRemove* - Po odstránení okna zo stromu DOM
- *onDragStart* - Pri začatí presúvania okna myšou.
- *onDragMove* - Počas presúvania okna myšou.
- *onDragStop* - Pri ukončení presúvania okna myšou.
- *onResizeStart* - Pri začatí upravovania rozmerov okna myšou.
- *onResizeMove* - Počas upravovania rozmerov okna myšou.
- *onResizeStop* - Pri ukončení upravovania rozmerov okna myšou.
- *onFocus* - Pri aktivovaní okna.

Navrhované metódy dialógového okna sú:

- *render()* - Vykreslí dialógové okno na koniec (ako posledného potomka) elementu *targetElement* z konfiguračného objektu.
- *renderAt(index)* - Vykreslí dialógové okno medzi potomkov (na pozíciu udanú parametrom *index*) elementu *targetElement* z konfiguračného objektu.
- *renderTo(element, index)* - Vykreslí dialógové okno medzi potomkov (na pozíciu udanú parametrom *index*) elementu špecifikovaného parametrom *element*.
- *remove()* - Odstráni dialógove okno zo stromu DOM.
- *hide() / show()* - Skryje / zobrazí dialógové okno.
- *moveBy(dx, dy)* - Presunie okno o hodnoty *dx* a *dy*.
- *moveTo(x, y)* - Zmení pozíciu okna na súradnice *x* a *y*.
- *setPosX(x) / getPosX()* - Nastavenie / získanie pozície na x-ovej osi.
- *setPosY(y) / getPosY()* - Nastavenie / získanie pozície na y-ovej osi.
- *setZIndex(z) / getZIndex()* - Nastavenie / získanie z-ovej súradnice.
- *setTitle(title) / getTitle()* - Nastavenie / získanie textu v záhlaví okna.
- *setWidth(w) / getWidth()* - Nastavenie / získanie šírky okna.
- *setHeight(h) / getHeight()* - Nastavenie / získanie výšky okna.
- *setDraggable(value)* - Hodnota parametru *value* (*true* alebo *false*) udáva, či bude okno presúvateľné pomocou myši.

- *setResizeable(value)* - Hodnota parametru *value* (*true* alebo *false*) udáva, či bude možné pomocou myši meniť rozmery okna.

4.3.2 DialogManager

DialogManager je trieda zodpovedná za správu otvorených dialógových okien. Bude obsahovať štyri hlavné vlastnosti:

- *openDialogs* - Kolekcia otvorených dialógových okien.
- *openAnnotations* - Kolekcia otvorených anotácií - ide o špeciálny typ dialógového okna, ktorý sa bude evidovať osobitne od ostatných okien.
- *frontDialog* - Ukazateľ na dialógové okno v popredí.
- *frontAnnotation* - Ukazateľ na anotáciu v popredí.

Medzi metódy triedy *DialogManager* budú patriť:

- *registerDialog(dlg) / unregisterDialog(dlg)* - Pridanie / odobranie dialógového okna.
- *registerAnnotation(dlg) / unregisterAnnotation(dlg)* - Pridanie / odobranie dialógového okna určeného pre zobrazovanie anotácie.
- *bringDialogToFront(dlg)/ bringAnnotationToFront(dlg)* - Prenos dialógového okna / okna s anotáciou do popredia.

4.3.3 ContentHolder

ContentHolder predstavuje abstraktný kontainer, do ktorého je možné vložiť ľubovoľný obsah. Pri vytváraní inštancie sa konštruktoru predá konfiguračný objekt, ktorým je možné prispôsobiť vlastnosti a chovanie kontainera. Medzi konfiguračné nastavenia budú patriť:

- *id* - Unikátne id pre element, ktorý bude predstavať kontainer v strome DOM.
- *classNames* - Reťazec názvov tried oddelených medzerou, ktoré sa priradia elementu.
- *render* - Nastavenie, ktoré špecifikuje, či sa má kontainer po vytvorení inštancie ihneď vykresliť.
- *targetElement* - DOM Element, do ktorého sa kontainer vykreslí.
- *contentElement* - DOM Element, ktorý sa použije ako hlavný obsah kontainera.

Navrhované metódy pre *ContentHolder*:

- *render()* - Vykreslí kontainer na koniec (ako posledného potomka) elementu *targetElement* z konfiguračného objektu.
- *renderAt(index)* - Vykreslí kontainer medzi potomkov (na pozíciu udanú parametrom *index*) elementu *targetElement* z konfiguračného objektu.
- *renderTo(element, index)* - Vykreslí kontainer medzi potomkov (na pozíciu udanú parametrom *index*) elementu špecifikovaného parametrom *element*.
- *remove()* - Odstráni kontainer zo stromu DOM.
- *hide() / show()* - Skryje / zobrazí kontainer.
- *add(element) / addAt(element, index) / remove(element)* - Pridá / odstráni DOM element do / z kontainera.
- *addButton(btn) / addButtonAt(btn, index) / removeButton(btn)* - Pridá / odstráni inštanciu triedy Button do / z kontainera.
- *addMessageBox(msg) / addMessageBoxAt(msg) / removeMessageBox(msg)* - Pridá / odstráni inštanciu triedy MessageBox do / z kontainera.

4.3.4 Toolbar

Toolbar predstavuje hlavný panel nástrojov editora anotácií. Bude obsahovať tlačítka pre ovládanie všetkých funkcií editora. Trieda *Toolbar* rozširuje triedu *Dialog*, takže obsahuje všetky jej vlastnosti, metódy a udalosti.

Okrem toho pridáva metódy:

- *addButton(btn) / addButtonAt(btn, index)* - Pridanie inštancie triedy Button do panela nástrojov.
- *removeButton(btn)* - Odstránenie inštancie triedy Button z panela nástrojov.
- *getButtonAt(index)* - Získanie inštancie triedy Button na zadanej pozícii.

4.3.5 Button

Button je trieda, ktorá umožní vytvárať tlačítka a priradzovať im akcie pri určitých udalostiach. Medzi základné konfiguračné nastavenia budú patriť:

- *title* - Text, ktorý sa zobrazí na tlačítku.
- *width* - Nastavenie šírky tlačítka.
- *height* - Nastavenie výšky tlačítka.
- *align* - Zarovnanie tlačítka.
- *icon* - Špecifikácia ikonky na tlačítku.
- *render* - Nastavenie, ktoré špecifikuje, či sa má tlačítko po vytvorení inštancie ihneď vykresliť.
- *disabled* - Nastavenie, ktoré špecifikuje, či má tlačítko po vytvorení inštancie zmeniť stav na *disabled*.
- *selected* - Nastavenie, ktoré špecifikuje, či má tlačítko po vytvorení inštancie zmeniť stav na *selected*.
- *targetElement* - DOM Element, do ktorého sa tlačítko vykreslí.
- *srcElement* - DOM Element, ktorý sa použije ako element pre tlačítko, tzn. že je možné tlačítko vytvoriť aj z už existujúceho elementu v dokumente.

Pri práci s tlačítkami bude možné využiť aj udalosti, ktoré generujú. Pôjde konkrétne o:

- *onClick* - Pri kliknutí myšou na tlačítko.
- *onRendered* - Pri pridaní tlačítka do stromu DOM.
- *onBeforeRemove* - Pred odstránením tlačítka zo stromu DOM.
- *onRemove* - Po odstránení tlačítka zo stromu DOM.
- *onMouseOver* - Ak sa myš nachádza nad tlačítkom.
- *onMouseOut* - Ak myš opustí tlačítko.

Navrhované metódy triedy *Button*:

- *render()* - Vykreslí tlačítko na koniec (ako posledného potomka) elementu *targetElement* z konfiguračného objektu.
- *renderAt(index)* - Vykreslí tlačítko medzi potomkov (na pozíciu udanú parametrom *index*) elementu *targetElement* z konfiguračného objektu.
- *renderTo(element, index)* - Vykreslí tlačítko medzi potomkov (na pozíciu udanú parametrom *index*) elementu špecifikovaného parametrom *element*.
- *remove()* - Odstráni tlačítko zo stromu DOM.

- *hide() / show()* - Skryje / zobrazí tlačítko.
- *setZIndex(z) / getZIndex()* - Nastavenie / získanie z-ovej súradnice.
- *setTitle(title) / getTitle()* - Nastavenie / získanie textu na tlačítku.
- *setWidth(w) / getWidth()* - Nastavenie / získanie šírky tlačítka.
- *setHeight(h) / getHeight()* - Nastavenie / získanie výšky tlačítka.
- *setDisabled(value)* - Hodnota parametru *value* (*true* alebo *false*) udáva, či bude tlačítko v stave *disabled* alebo nie.
- *setSelected(value)* - Hodnota parametru *value* (*true* alebo *false*) udáva, či bude tlačítko v stave *selected* alebo nie.
- *setHover(value)* - Hodnota parametru *value* (*true* alebo *false*) udáva, či bude tlačítko v stave *hover* alebo nie.
- *setIcon(icon)* - Nastavenie ikonky pre tlačítko.

4.3.6 MessageBox

MessageBox bude slúžiť na jednoduchšie vytváranie správ, ktoré sa zobrazujú užívateľovi. Konštruktoru stačí predať iba názov, text a typ správy, z ktorých sa automaticky vygeneruje kód v HTML. Typ správy sa rozlišuje graficky pomocou ikonky a vrámci diplomovej práce budú vytvorené tieto preddefinované typy:

- Chybová správa.
- Upozornenie.
- Oznam.
- Správa OK.
- Čakanie na určitú udalosť.
- Informácie o anotácii.

4.3.7 Tree

Tree je trieda, pomocou ktorej bude možné vytvoriť rozbaľovací strom prvkov. Medzi jej vlastnosti budú patriť:

- *childNodes* - Pole všetkých priamych potomkov.
- *assignedObject* - Ku každému uzlu stromu bude možné priradiť ľubovoľný objekt, čo umožní organizáciu objektov do hierarchickej stromovej štruktúry.
- *parentNode* - Odkaz na rodičovský uzol.
- *text* - Textový popis uzlu.

Navrhované metódy triedy *Tree*:

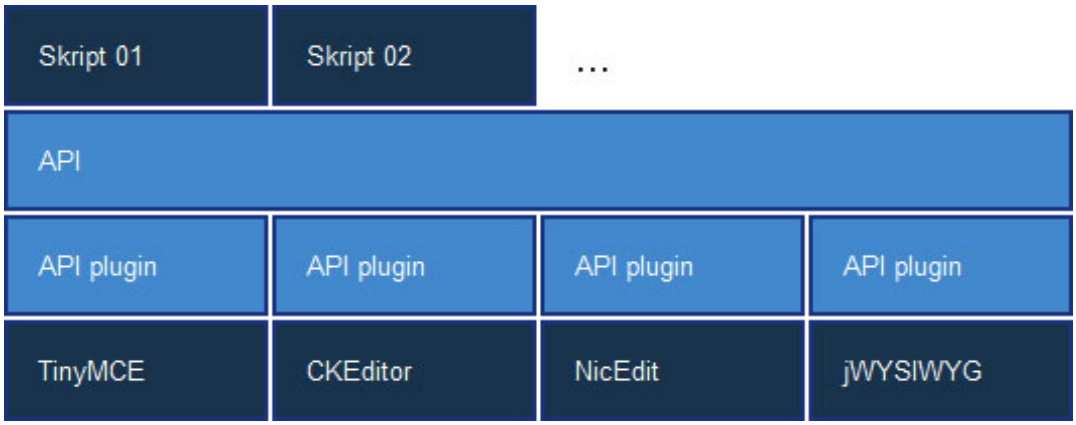
- *addChild(treeNode) / addChildAt(treeNode, index) / removeChild(treeNode)* - Pridá / odstráni uzol stromu.
- *expand() / collapse()* - Rozbalenie / zbalenie podstromu.
- *setSelected(value)* - Hodnota parametru *value* (*true* alebo *false*) udáva, či bude uzol označený.

4.3.8 Motívy vzhľadu

Motívy vzhľadu (anglicky themes) umožnia prispôsobovanie vizuálnej stránky komponent užívateľského rozhrania. Budú zložené z obrázkov a súborov definujúcich kaskádové štýly (CSS). Všetky motívy vzhľadu budú organizované v jednom adresári s názvom *themes*, pričom každý motív bude v samostanej zložke.

4.4 Návrh rozhrania pre spoluprácu WYSIWYG editorov s editorom anotácií

Rozhranie pre spoluprácu WYSIWYG editorov s editorom anotácií umožní jednotný spôsob práce s rôznymi typmi editorov. Ide o tzv. API (Application Programming Interface), ktorého princíp ilustruje obrázok č. 8.



Obrázok 8: Koncept rozhrania pre spoluprácu WYSIWYG editorov s extérnymi skriptami.

Rozšírenie je tvorené aplikačným rozhraním a špecifickými implementáciami metód tohto rozhrania pre jednotlivé editory. Extérnym skriptom budú skryté ich špecifické vlastnosti a budú tak môcť pracovať rovnakým spôsobom s rôznymi typmi WYSIWYG editorov.

Štruktúru API tvoria:

- *WysiwygEditor* - Rozhranie reprezentujúce WYSIWYG editor. Všetky inštancie editorov implementujú rozhranie *WysiwygEditor*. Medzi základné vlastnosti patria:
  - *id* - Predstavuje ID inštancie editora.
  - *type* - Typ editora.
  - *instance* - Predstavuje samotnú inštanciu WYSIWYG editora.

  Metódy:
  - *getContent* - Získanie obsahu editora.
  - *setContent* - Nastavenie obsahu editora.
  - *getElement* - Získanie elementu, ktorý bol nahradený editorom.
  - *getIFrameElement* - Získanie elementu IFRAME, ktorý bol vytvorený editorom.
  - *getIFrameDocumentElement* - Získanie elementu DOCUMENT elementu IFRAME, ktorý bol vytvorený editorom.

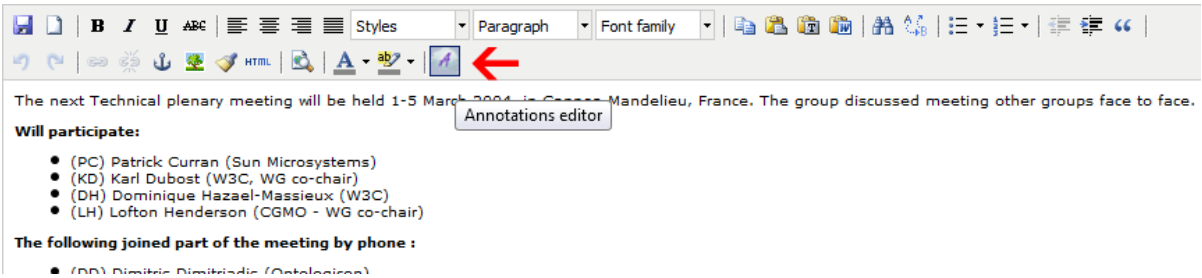
- *activate* - Aktivovanie editora.
- *deactivate* - Deaktivovanie editora.
- *remove* - Odstránenie inštancie editora.
- *getSelectionContent* - Získanie obsahu výberu.
- *setSelectionContent* - Nastavenie obsahu výberu.
- *setSelectionRange* - Označenie zadaného rozsahu. Argumentom tejto metódy je objekt Range, definovaný v [54].
- *getSelectionRange* - Vráti objekt Range, definovaný v [54], ktorý popisuje výber.
- *addSelectionRange* - Pridanie nového výberu k už existujúcemu. Ak sa v dokumente žiadny výber nenachádza, metóda sa správa ako *setSelectionRange*. Argumentom tejto metódy je tiež objekt Range.
- *getSelectionObject* - Vráti natívny objekt Selection.
- *getSelectionNode* - Vráti DOM element, ktorý je rodičom pre elementy v aktuálnom výbere.

Udalosti:

- *onBeforeInit* - Vzniká pred inicializáciou editora.
- *onInit* - Vzniká po inicializácii editora.
- *onBeforeActivate* - Vzniká pred aktivovaním editora.
- *onBeforeDeactivate* - Vzniká pred deaktivovaním editora.
- *onActivate* - Vzniká po aktivovaní editora.
- *onDeactivate* - Vzniká po deaktivovaní editora.
- *onRemove* - Vzniká pri odstránení editora.
- *onClick* - Udalosť kliknutia myšou.
- *onDblClick* - Dvojité kliknutie myšou.
- *onMouseUp* - Udalosť uvolnenia tlačidla myši.
- *onMouseDown* - Udalosť stlačenia tlačidla myši.
- *onKeyDown* - Stlačenie klávesy.
- *onKeyUp* - Uvolnenie klávesy.
- *onKeyPress* - Stlačenie a následne uvolnenie klávesy.
- *onSubmit* - Odoslanie formulára.
- *onReset* - Resetovanie formulára.
- *onBeforeGetContent* - Pred získaním obsahu editora.
- *onBeforeSetContent* - Pred nastavením obsahu editora.
- *onGetContent* - Po získaní obsahu editora.
- *onSetContent* - Po nastavení obsahu editora.
- *onContentChange* - Pri zmene obsahu editora.
- *onUndo* - Pri stlačení na Undo.
- *onRedo* - Pri stlačení na Redo.
- *onBeforeSetSelectionContent* - Vzniká pred nastavením obsahu výberu.
- *onBeforeGetSelectionContent* - Vzniká pred získaním obsahu výberu.
- *onSetSelectionContent* - Vzniká po nastavení obsahu výberu.
- *onGetSelectionContent* - Vzniká po získaní obsahu výberu.

4.5 Návrh rozšírení pre zvolené WYSIWYG editory

Hlavnou úlohou rozšírenia pre jednotlivé editory bude spúšťanie a vypínanie užívateľského rozhrania editora anotácií z panela nástrojov. Na obrázku č. 9 je ukážka panela nástrojov editora TinyMCE, v ktorom sa nachádza tlačítko editora anotácií.



Obrázok 9: Panel nástrojov editora TinyMCE spolu s tlačítkom pre editor anotácií.

Po kliknutí na tlačítko sa spustí užívateľské rozhranie editora anotácií a tlačítko zostane v aktívnom (stlačenom) stave. Po opätovnom kliknutí sa editor anotácií vypne a tlačítko zmení stav na neaktívne.

Pluginy sa budú opierať o princípy vytvárania rozšírení pre jednotlivé editory, ktoré sú podrobne diskutované v kapitole 3. Všetky rozšírenia sa vytvoria spolu s jazykovými predkladmi v českom, anglickom a slovenskom jazyku.

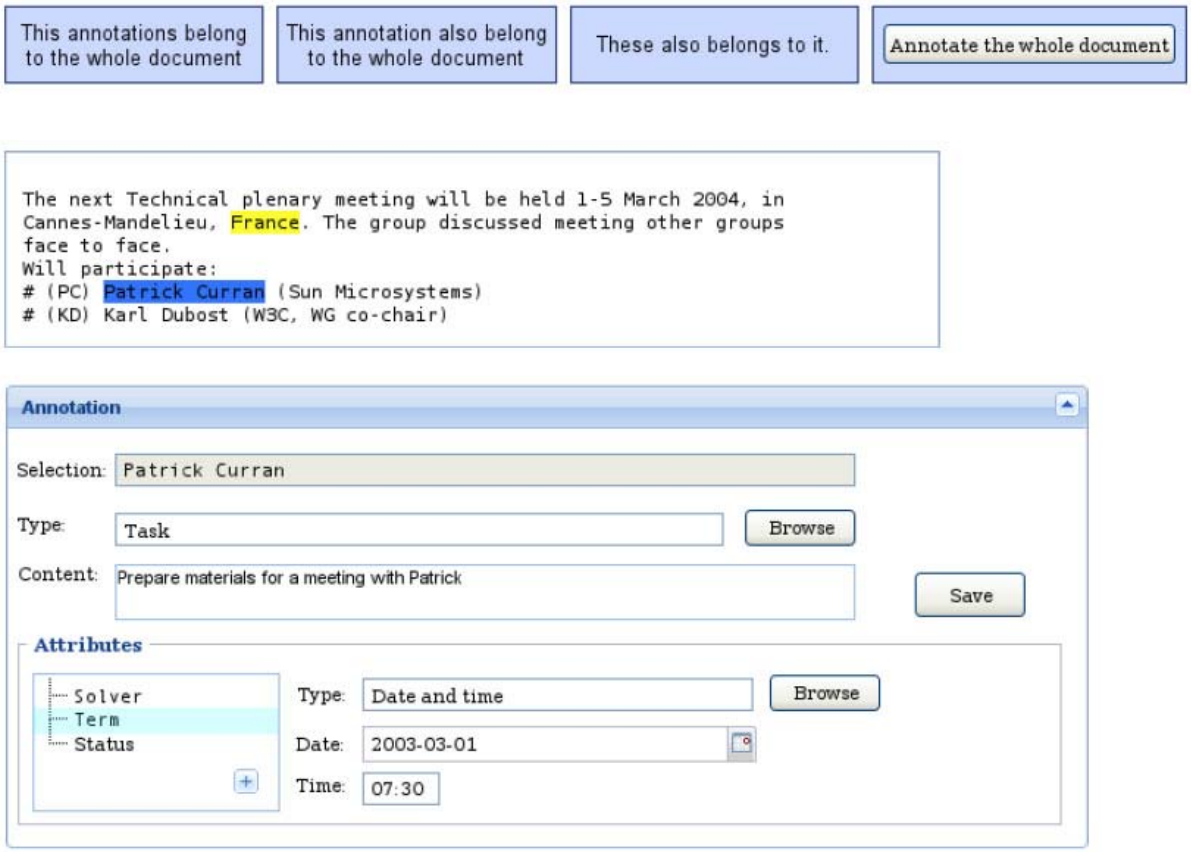
4.6 Návrh obrazoviek s popisom chovania

Na obrázku 10 je návrh základného editora, ktorý obsahuje postranný panel s anotáciami priradenými celému dokumentu, ďalej samotný text dokumentu a dialógové okno pre vytvorenie anotácie.

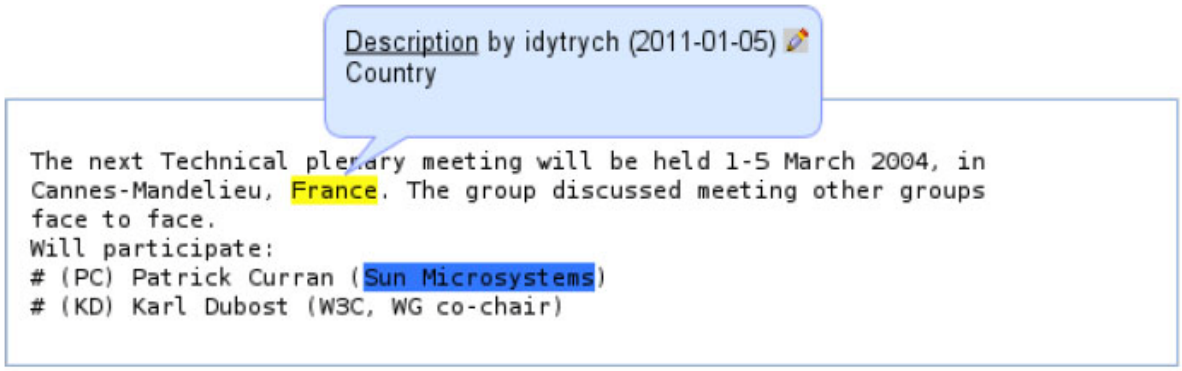
Pri anotovaní užívateľ najprv označí anotovaný fragment, následne zadá informácie o anotácii a uloží. Anotáciu je možné priradiť aj celému dokumentu pomocou tlačidla v postrannom panely. Existujúce anotácie sa zobrazujú farebným odlíšením anotovaného fragmentu textu a v prípade, že užívateľ prejde nad tento fragment myšou, zobrazí sa bublina s informáciami o anotácii. Pokiaľ užívateľ na tento fragment klikne, zostane bublina s informáciami zobrazená až do kliknutia mimo anotovaný text. Bublina tiež obsahuje ikonu pre editáciu a po kliknutí na ňu sa anotácia vypíše do formulára, kde je možné požadované informácie upraviť [47].

Základnými prvkami dialógového okna pre vytvorenie anotácie sú:

- Pole s vybraným textom.
- Pole pre určenie typu anotácie.
- Pole pre zadanie textového obsahu anotácie.
- Výpis priradených atribútov s možnosťou pridávania nových.
- Tlačidlo pre uloženie anotácie.



Obrázok 10: Návrh užívateľského rozhrania editora anotácií. Hore je postranný panel s anotáciami priradenými celému dokumentu, v strede je samotný text dokumentu a dole sa nachádza dialógové okno pre vyvorenie anotácie [47].

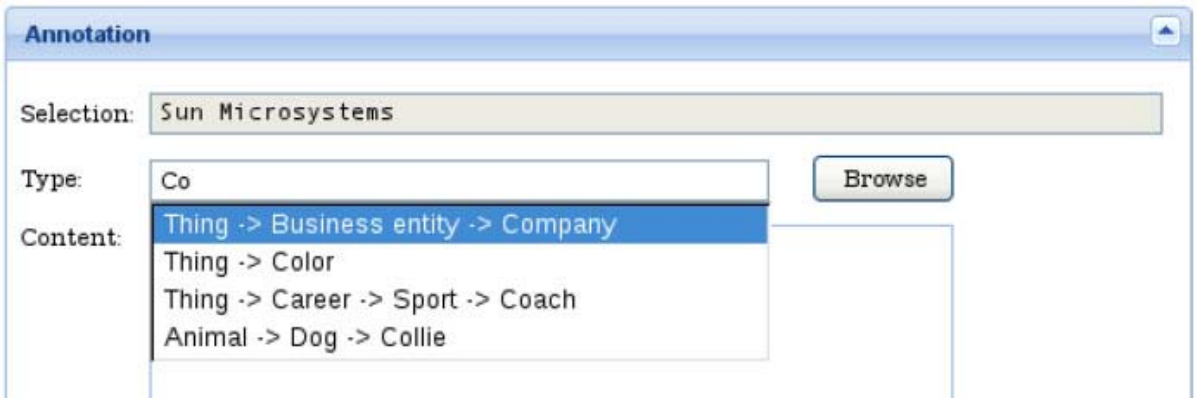


Obrázok 11: Zobrazenie bubliny s informáciami o anotácii [47].

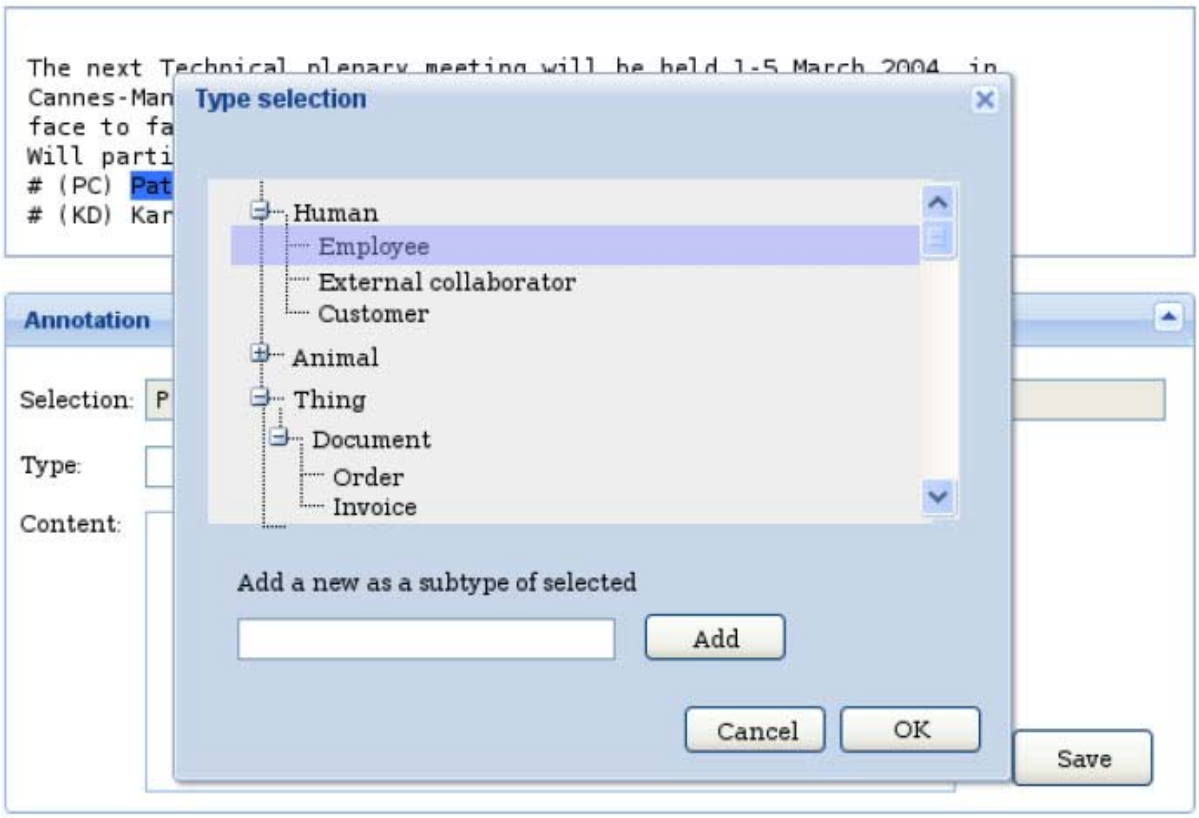
Určovanie typu anotácie je možné vytvorením nového typu alebo výberom z už existujúcich typov. Pre vytvorenie nového typu stačí napísať jeho názov do príslušného vstupného poľa. Výber z existujúcich typov je možný dvoma spôsobmi [47]:

- Zobrazením zoznamu typov, ktorých linearizovaný názov obsahuje text vložený do vstupného poľa (obrázok č. 12).

- Výberom typu zo stromu typov. Užívateľ klikne na tlačidlo vedľa poľa pre zadanie typu a otvorí sa mu dialóg (obrázok č. 13), v ktorom môže strom typov upravovať a rozširovať.



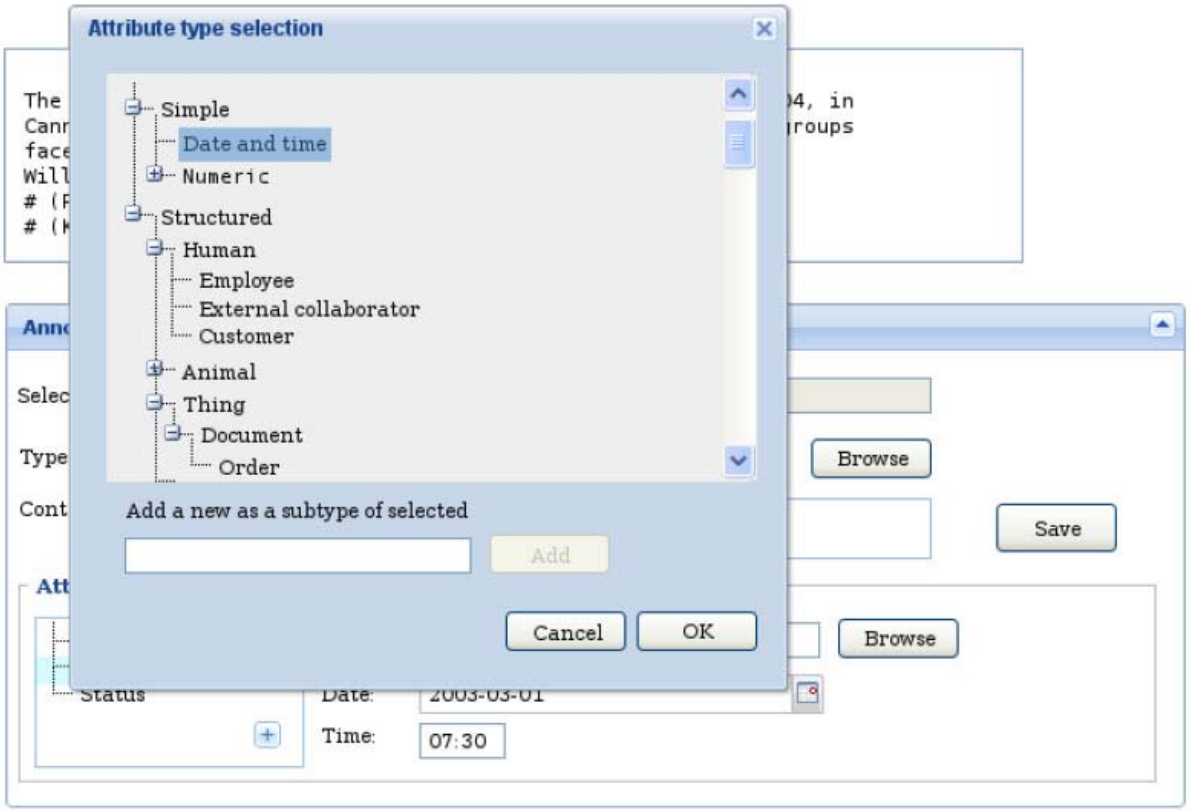
Obrázok 12: Zobrazenie zoznamu typov pri zadávaní názvu do vstupného poľa [47].



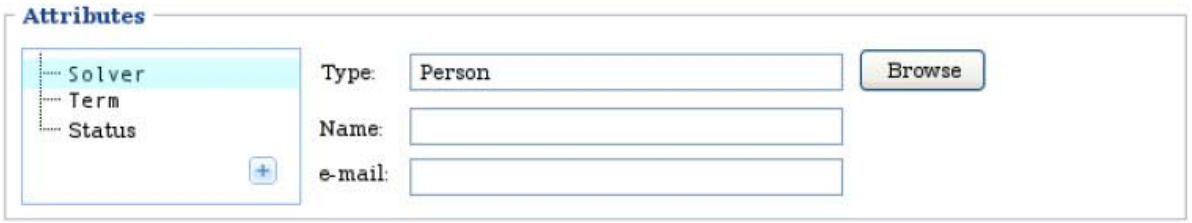
Obrázok 13: Výber typu zo stromu typov [47].

K anotáciám je možné priradiť ľubovoľný počet atribútov. Každý atribút má názov, typ a hodnotu. Hodnotou atribútu môže byť aj vnorená anotácia. Atribúty tejto vnorenej anotácie budú tvoriť vetvu stromu vychádzajúcu z daného atribútu. Týmto spôsobom je možné prehľadne zobraziť ľubovoľný počet úrovní zanorenia [47].

Výber typu atribútu (obrázok č. 14) je možný rovnakými spôsobmi ako výber typu anotácie. Pri výbere zo stromu typov sa zobrazujú rovnaké typy ako pri výbere typu anotácie, doplnené o jednoduché a rozšírujúce dátové typy. Do podstromu s jednoduchými typmi nemôže užívateľ pridávať nové typy.



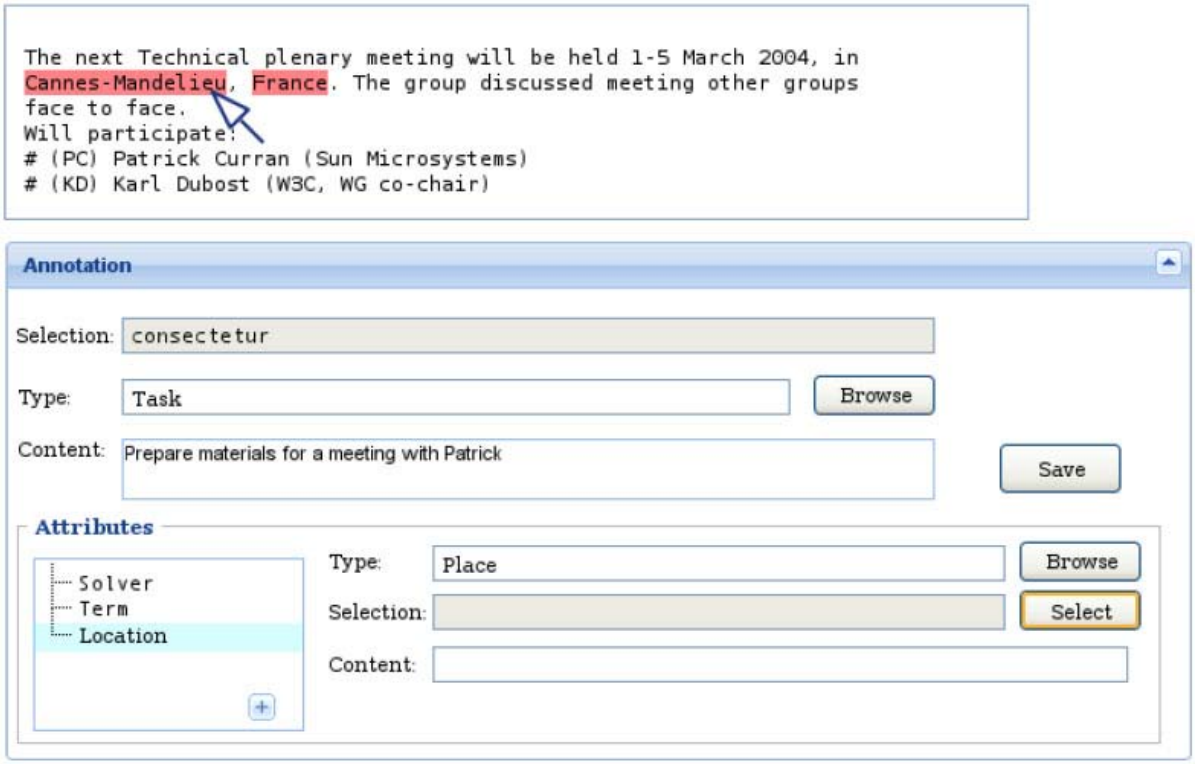
Obrázok 14: Výber typu atribútu zo stromu typov [47].



Obrázok 15: Ukážka polí pre zadávanie hodnoty dátového typu osoba [47].

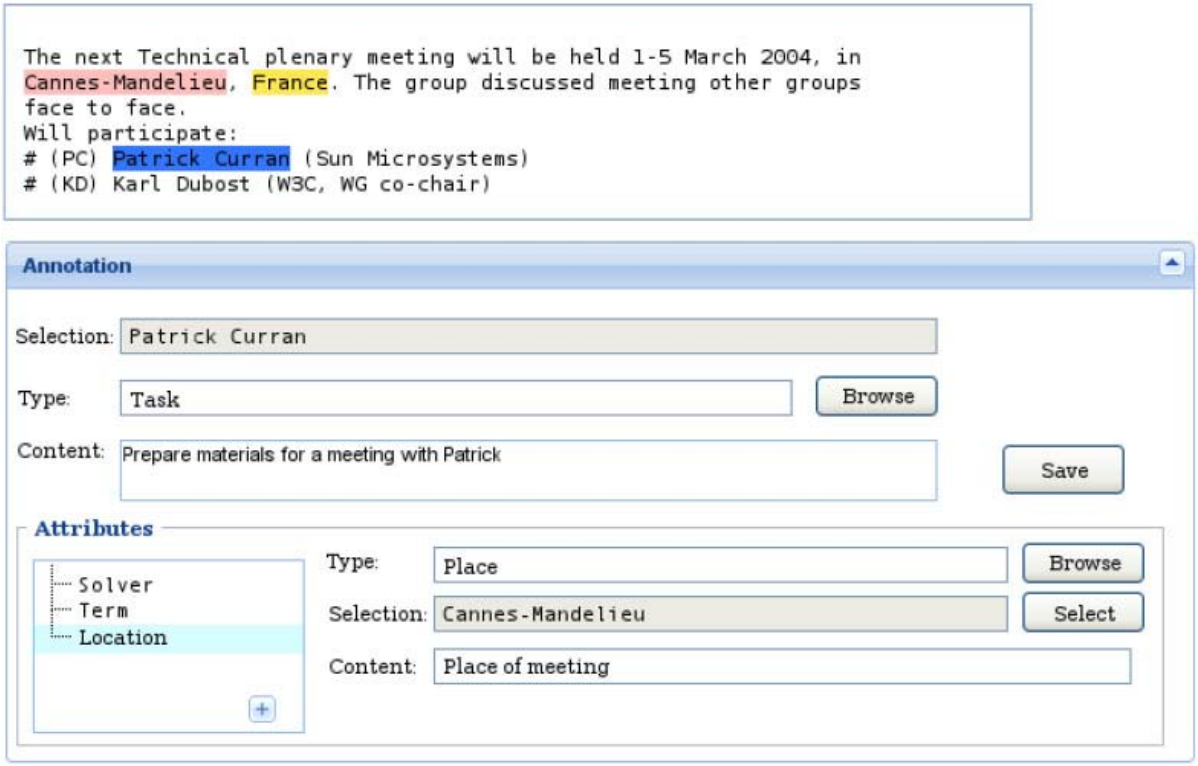
Na obrázku č. 15 je ukážka polí pre zadávanie hodnoty dátového typu osoba. Pri zadávaní konkrétnej osoby sa zobrazí ponuka mien užívateľov (obrázok č. 16). Keď bude výber mena jednoznačný, email sa doplní automaticky. Rovnako to platí aj opačne, pri jednoznačnom výbere emailu, vtedy sa automaticky doplní položka meno [47].

Obrázok 16: Zadávanie konkrétnej osoby [47].

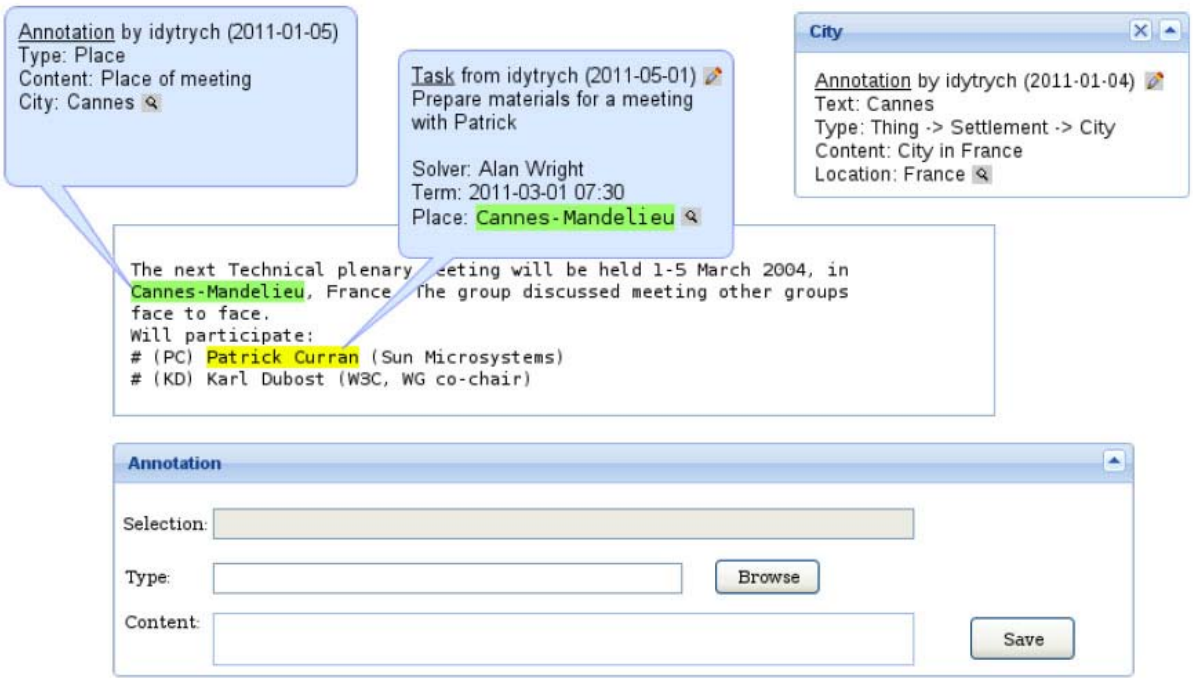


Obrázok 17: Zadávanie odkazu na anotáciu [47].

Na obrázku č.17 je ukážka formulára pre zadávanie vnorenej anotácie. Pri výbere exsitujúceho typu atribútu sa v texte vyznačia všetky anotácie daného typu. Ak užívateľ klikne na niektorý z nich, namiesto vnorenej anotácie sa do atribútu uloží odkaz na zvolenú anotáciu. Pokiaľ užívateľ nechce vložiť odkaz, ale vnorenú anotáciu (obrázok č.18), musí kliknúť na tlačidlo pre výber fragmentu a následne ťahom myši označiť zvolený text. Po dokončení výberu sa znova zvýrazní fragment anotácie najvyššej úrovne [47].



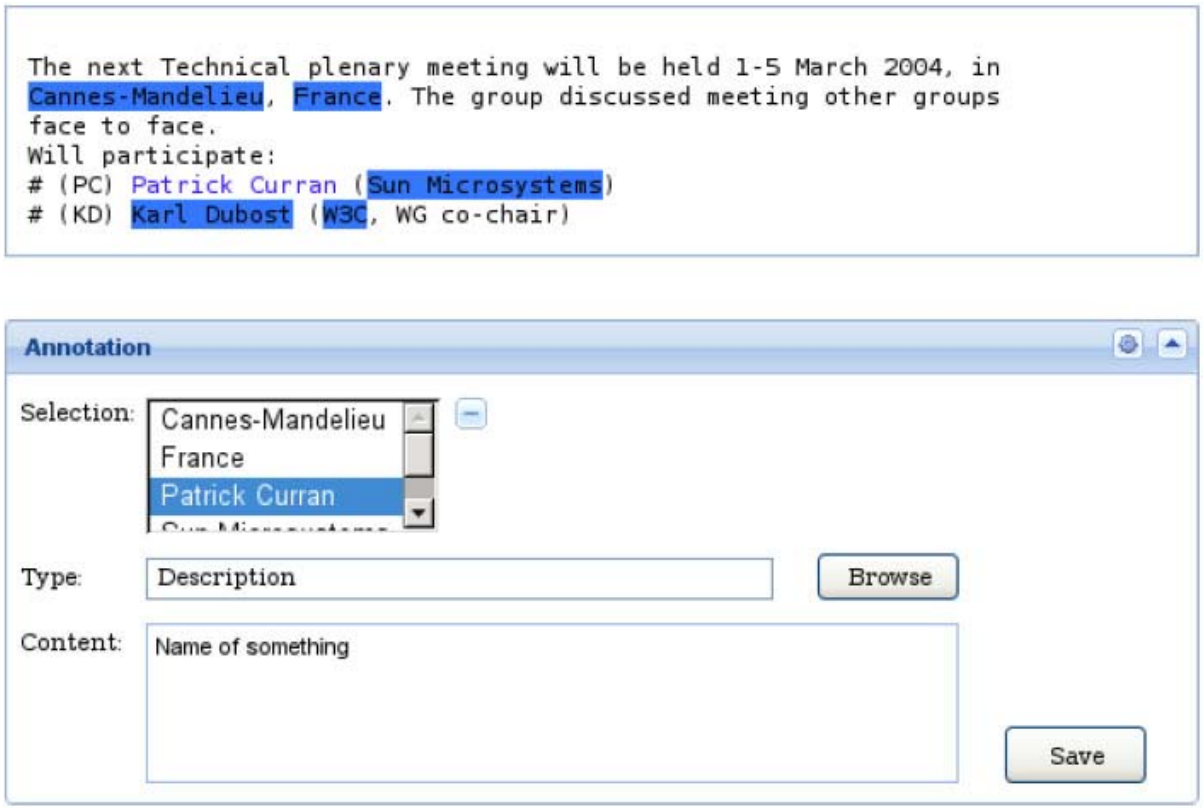
Obrázok 18: Zadávanie vnorenej anotácie [47].



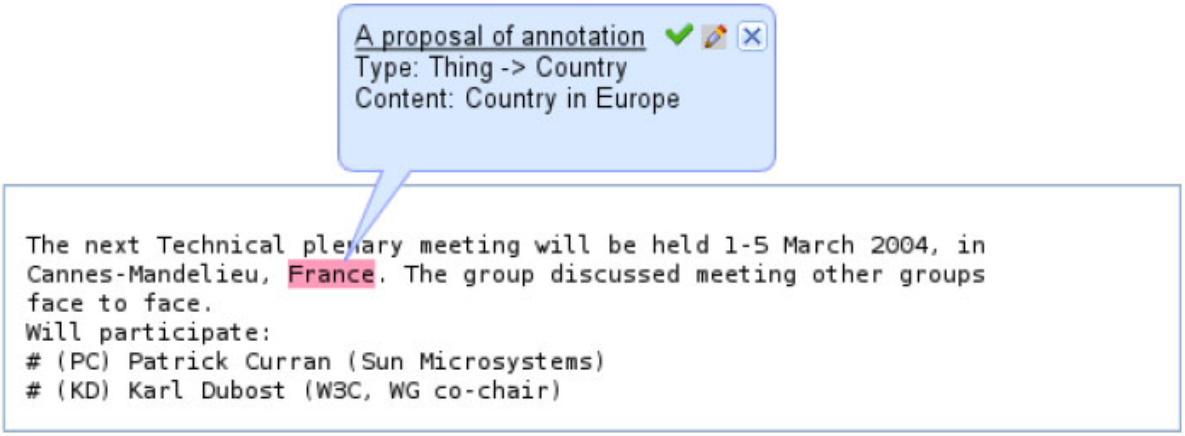
Obrázok 19: Zobrazovanie anotácií druhej a vyššej úrovne zanorenia [47].

Zobrazovanie anotácií druhej a vyššej úrovne zanorenia demonštruje obrázok č.19. Pokiaľ užívateľ nájde myšou nad anotovaný fragment, zobrazí sa mu nie len anotácia k tomuto fragmentu, ale tiež anotácie, ktoré sú na prvej úrovni zanorenia. Anotácie druhej a vyššej úrovne sa kvôli prehľadnosti nebudú zobrazovať v bublinách ale v oknách, ako to zobrazuje obrázok č.19 [47].

Na stránke je možné vybrať aj viacero fragmentov textu a anotovať ich súčasne. Pri uložení sa pre každý fragment vygeneruje osobitná anotácia, ako keby sa zadávala samostatne. Tento spôsob je znázornený na obrázku č. 20. Výber viacerých fragmentov súčasne je možné využiť aj pri vytváraní atribútov. Vytvorí sa tak viacero atribútov s rovnakým názvom, typom a obsahom.

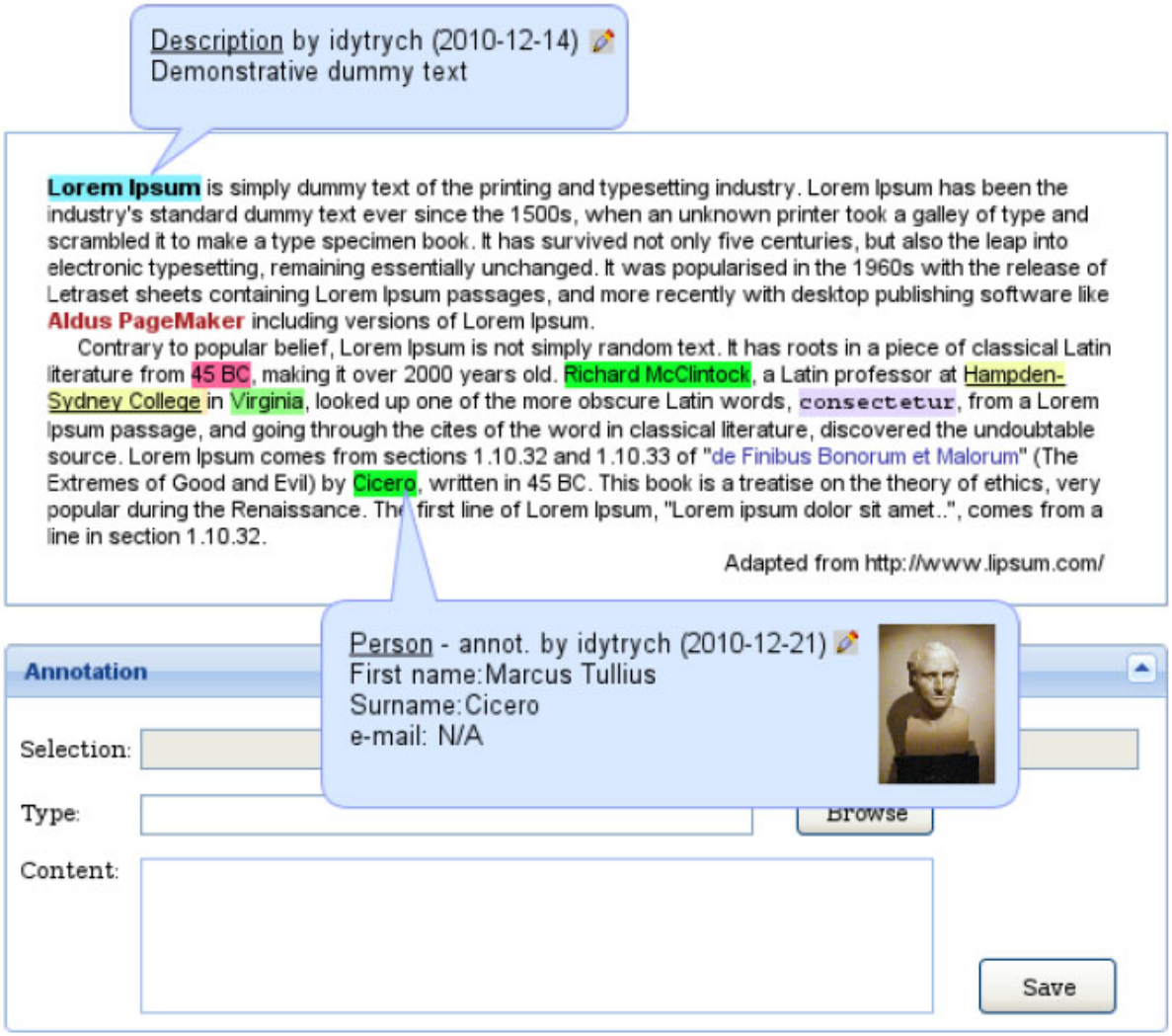


Obrázok 20: Režim výberu a súčasného anotovania viacerých fragmentov [47].



Obrázok 21: Ponúkanie anotácií [47].

Obrázok č.21 ukazuje príklad ponúkania anotácií. Keď si užívateľ zapne režim ponúkania anotácií, editor anotácií požiada server o anotácie a následne ich zobrazí užívateľovi.



Obrázok 22: Zobrazenie viacerých typov anotácií [47].

Ak sa bude v texte nachádzať viacero typov anotácií, editor ich zobrazí rôznym spôsobom. Užívateľ bude mať možnosť voľby zobrazenia (farby, fonty, a pod.).

5 Implementácia

Táto kapitola popisuje implementáciu editora anotácií spolu s výberom najzávažnejších problémov, ktoré sa pri vývoji vyskytli.

5.1 IDE

Za vývojové prostredie pri programovaní editora anotácií bol zvolený PSPad[1]. Ide o český editor, ktorý je dostupný zdarma a poskytuje všetky potrebné funkcie pre vývoj aplikácie v jazyku JavaScript.

5.2 Implementácia editora anotácií

Implementácia editora anotácií sa opiera o návrh, ktorý je podrobne popísaný v podkapitole 4.1. Vytvorili sa všetky triedy s navrhovanými vlastnosťami a metódami, pričom sa dodržovala následujúca konvencia organizácie zdrojového kódu:

- *Jedna trieda = jeden súbor* - Vrámci zvýšenia prehľadnosti kódu sú všetky triedy umiestnené v osobitnom súbore.
- *Pomenovanie súborov* - Každý súbor je pomenovaný podľa názvu triedy, ktorú obsahuje, napríklad trieda *AEd* sa nachádza v súbore s názvom *AEd.js*.
- *Umiestnenie v zložkách* - Pre každý menný priestor sa vytvoril adresár s identickým názvom a všetky triedy patriace do daného priestoru sú umiestnené v tejto zložke, napríklad trieda *Dispatcher*, ktorá sa hierarchicky nachádza v mennom priestore *utils*, je umiestnená v súbore *Dispatcher.js*, ktorý sa nachádza v adresáry *utils/*.

Najčastejšie problémy pri vývoji editora anotacií sa týkali nekompatibility implementácií interprétov jazyka JavaScript a úrovne modelu DOM medzi rôznymi typmi a verziami prehliadačov. Najhoršie sú na tom prehliadače Internet Explorer pred verziou 8. Odlišnosti a chyby ich inteprétov jazyka JavaScript a modelu DOM sú natoľko rozsiahle, že ich vypísanie by mohlo byť témou osobitnej diplomovej práce a ani implementácia editora anotácií nie je odladená na používanie v týchto prehliadačoch. Preto v tejto podkapitole uvediem iba spôsoby riešenia týchto problémov a nezameriam sa na ich konkrétny výpis v jednotlivých prehliadačoch.

Jedným z často používaných riešení na problémy nekompatibilných implementácií JavaScriptu a úrovne modelu DOM je detekcia prehliadača. Ako vyplýva z názvu, v implementovanej aplikácií sa detekuje typ a verzia prehliadača a samotný kód sa potom rozdelí do vetví podľa prehliadača, pre ktorý je určený. Tento spôsob je síce veľmi rozšírený, no pri implementácií editora anotácií nebol použitý. Jeho nevýhodou je to, že nezaručuje bezproblémový chod aplikácie v nových typoch a verziách prehliadačov.

Namiesto detekcie prehliadača sa pri implementácii editora anotácií použila detekcia podpory požadovanej funkcionality. To znamená, že sa v aplikácií nedetekuje typ a verzia prehliadača, ale namiesto toho sa zistí dostupnosť konkrétnej funkcie alebo objektu pred samotným použitím.

Pre ilustráciu uvediem príklad priradenia obsluhy udalosti kliknutia myšou na konkrétny element. Pri tradičnom spôsobe priradzovania obsluhy udalostí v JavaScripte (tiež uvádzanom ako DOM Level 0) by zdrojový kód mohol vyzerať takto:

[1] *http:// www.pspad.com*

```
var element = document.getElementById("id_elementu");
element.onclick = function() {
    alert("ok");
};
```

Z dokumentu získame element podľa jeho ID a priradíme mu obsluhu udalosti *onclick*. Obsluha udalostí na úrovni DOM Level 2 využíva dve metódy, ktoré riešia priradzovanie a odstráňovanie obslúh udalostí: *addEventListener()* a *removeEventListener()*. Tieto metódy existujú na všetkých uzloch modelu DOM a prijímajú tri argumenty [11]:

- Názov udalosti, ktorá sa má obslúžiť.
- Funkciu obsluhy události.
- Logickú hodnotu, ktorá signalizuje, či sa má zadaná obsluha udalosti zavolať vo fáze zachytávania (hodnota *true*) alebo počas fázy prebublávania (hodnota *false*).

Kód z predchádzajúceho príkladu by pri použití nástrojov DOM Level 2 mohol vyzerať:

```
var element = document.getElementById("id_elementu");
element.addEventListener("click", function() {
    alert("ok");
}, false);
```

Obsluha udalostí v prehliadačoch Internet Explorer je odlišná oproti tomu, ako to bolo uvedené v predchádzajúcich príkladoch. Internet Explorer implementuje podobné metódy ako *addEventListener()* a *removeEventListener()*, ktoré sa volajú *attachEvent()* a *detachEvent()* a prijímajú dva argumenty:

- Názov udalosti, ktorá sa má obslúžiť.
- Funkciu obsluhy události.

Kód z predchádzajúcich príkladov vyzerá pre Internet Explorer takto:

```
var element = document.getElementById("id_elementu");
element.attachEvent("onclick", function() {
    alert("ok");
});
```

Pri implementácii editora anotácií sa vytvorila trieda *AEd.dom.Events* (súbor *Events.js*), ktorej úlohou je poskytnutie metód pre prácu s udalosťami, ktoré budú fungovať vo väčšine prehliadačov. Ďalšia ukážka kódu zobrazuje riešenie jednej z metód: *addHandler()*, ktorá priradí elemetu obsluhu špecifikovanej udalosti. Argumenty funkcie *addHandler()* sú:

- *element* - DOM element, ktorému chceme priradiť obsluhu udalosti.
- *type* - Typ udalosti, ktorú chceme obsluhovať.
- *handler* - Funkcia obsluhy udalosti.
- *scope* - Kontext, v ktorom sa funkcia obsluhy udalosti spustí.

Zdrojový kód metódy *addHandler()*:

```
t.addHandler = function(element, type, handler, scope) {
   var F = function(e) { handler.call(scope, e); }
   if (element.addEventListener) {
      element.addEventListener(type, F, false);
   }
   else if (element.attachEvent) {
      element.attachEvent("on"+type, F);
   }
   else {
      element["on"+type] = F;
   }
}
```

Na začiatku tejto funkcie sa overuje existencia metódy *addEventListener()*, pretože v súčasnej dobe prevažuje počet prehliadačov, ktorý ju podporujú a pravepodobnosť, že sa podmienka vyhodnotí kladne (hodnotou *true*), je celkom vysoká. To znamená aj optimalizáciu výkonu, pretože interprét nemusí vyhodnocovať výrazy vo zvyšných vetvách príkazu *if-else* (ktoré nadobúdajú hodnotu *true* štatisticky v menšom počte prípadov).

Ako ďalšia v poradí sa testuje existencia metódy *attachEvent()*, ktorá existuje v prehliadačoch Internet Explorer. Úplne poslednou možnosťou, pokiaľ zlyhali predošlé dve, je tradičný spôsob priradzovania obsluhy udalostí, ktorý je charakteristický pre DOM Level 0.

Zdrojový kód metódy *addHandler()* ilustruje, akým spôsobom sa pri vývoji editora anotácií riešil problém nekompatibility implementácií interprétov jazyka JavaScript a modelu DOM medzi rôznymi typmi a verziami prehliadačov - detekciou podpory požadovanej funkcionality namiesto detekcie typu a verzie prehliadača. Rovnakým spôsobom sa riadil vývoj aj ostatných metód, no vrámci rozsahu tejto práce nebolo možné ošetriť všetky prehliadače v rôznych verziách. Ide totiž o náročnú úlohu, ktorá si vyžaduje bohaté skúsenosti programátora s touto problematikou.

5.3 Implementácia knižnice pre užívateľské rozhranie

Implementácia knižnice pre užívateľské rozhranie sa opierala o návrh, ktorý je podrobne popísaný v podkapitole 4.3. Konvencia organizácie zdrojového kódu bola totožná s tou, ktorá je uvedená v podkapitole 5.2.

Knižnica obsahuje základné komponenty užívateľského rozhrania, ktoré boli potrebné pre editor anotácií. Všetky komponenty sa implementovali bez použitia zložitých univerzálnych knižníc. Vrámci vývoja knižnice pre užívateľské rozhranie boli implementované tieto triedy:

- *Dialog* (súbor *Dialog.js*) - Trieda, ktorá umožňuje vytváranie dialógových okien.
- *DialogManager* (súbor *DialogManager.js*) - Zodpovedný za správu okien, eviduje všetky otvorené dialógové okná, zaisťuje prenos aktívneho okna do popredia a pod.
- *ContentHolder* (súbor *ContentHolder.js*) - Komponenta, ktorá predstavuje abstraktný kontainer, do ktorého je možné vložiť ľubovoľný obsah.
- *Toolbar* (súbor *Toolbar.js*) - Predstavuje hlavný panel nástrojov editora anotácií. *Toolbar* obsahuje tlačítka pre ovládanie všetkých funkcií editora.

- *Button* (súbor *Button.js*) - Trieda, ktorá umožní vytvorenie tlačítka a priradenie akcií pri určitých udalostiach, ako napr. kliknutie myšou.
- *MessageBox* (súbor *MessageBox.js*) - Trieda, ktorá zjednodušuje vytváranie správ, ktoré sa zobrazia užívateľovi.
- *Tree* (súbor *Tree.js*) - Komponenta vytvárajúca rozbaľovací strom prvkov.

Najčastejšie problémy pri vývoji knižnice pre užívateľské rozhranie boli rovnaké ako pri implementácii editora anotacií. Týkali sa nekompatibility implementácií interprétov jazyka JavaScript a podpory modelu DOM medzi rôznymi typmi a verziami prehliadačov. Popis riešenia problémov tohoto typu je podrobne opísaný v kapitole 5.2.

Dôležitým bodom pri vývoji knižnice pre užívateľské rozhranie bol návrh štruktúry DOM elementov, ktoré budú jednotlivé komponenty tvoriť. Kvalita HTML kódu ovplyvní jednoduchosť prispôsobovania grafického vzhľadu pomocou motívov vzhľadu (anglicky themes). Podrobný popis všetkých prvkov je nad rozsah textovej časti diplómovej práce, preto v tejto kapitole popíšem iba jeden z nich - dialógové okno. Štruktúru ostatných prvkov je možné dohľadať v zdrojových kódoch vytvorenej aplikácie.

Následujúca ukážka zobrazuje kód HTML, ktorý tvorí základ pre dialógové okno:

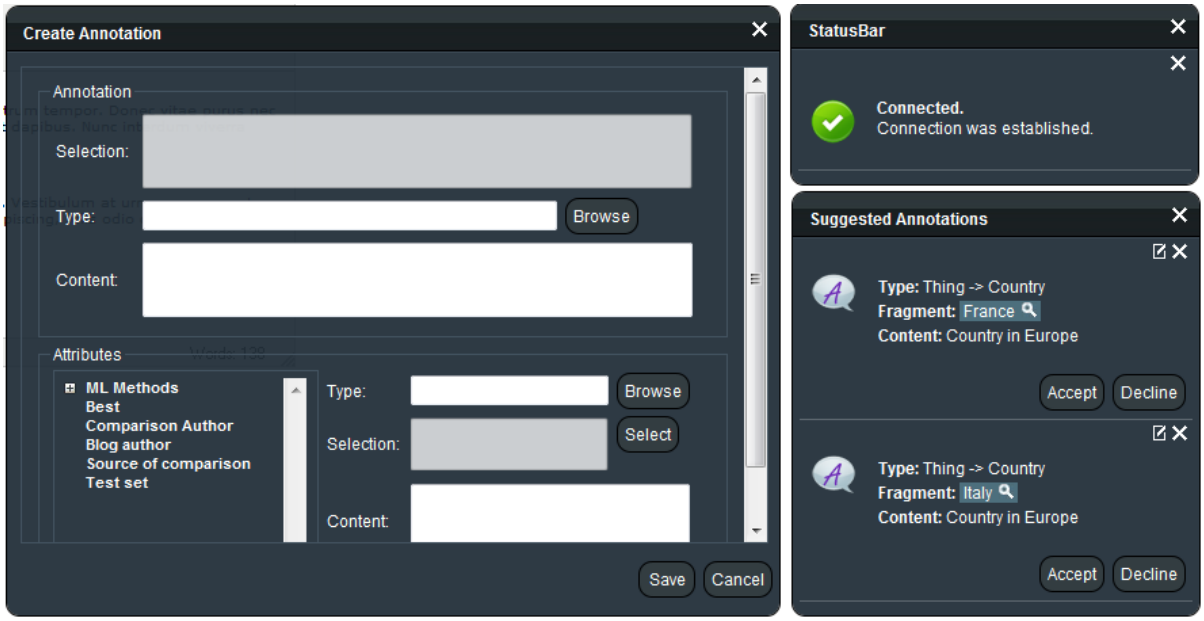
```
<div class="aed-ui-dialog aed-ui-corner-all
            aed-ui-draggable aed-ui-resizable"
     style="width: 296px; height: 201px;
            left: 671px; top: 261px; z-index: 1000;">

   <div class="aed-ui-dialog-titlebar aed-ui-corner-all">
      <span class="aed-ui-dialog-title">Title</span>
   </div>

   <div class="aed-ui-dialog-content"></div>
   <div class="aed-ui-resizable-handle
               aed-ui-resizable-handle-n"
        style="z-index: 1011;">
   </div>
...
<!-- Analogicky pre ostatné tri strany -->
...
   <div class="aed-ui-resizable-handle
               aed-ui-resizable-handle-ne"
        style="z-index: 1015;">
   </div>
...
<!-- Analogicky pre ostatné tri diagonály -->
...
</div>
```

Ako je vidieť v uvedenom výpise, HTML kód dialógového okna obsahuje iba minimum elementov, ktoré sú nevyhnutné pre dosiahnutie požadovanej funkcionality. Rovnako je to aj s vloženými (anglicky inline) CSS štýlmi. HTML kód ich obsahuje iba nevyhnutné množstvo, všetky ostatné definície su uvedené v extérnom súbore, ktorý je súčasťou motívu vzhľadu. Vizuálna

úprava vzhľadu prvkov užívateľského rozhrania je tak veľmi jednoduchá a nevyžaduje znalosti jazyka JavaScript, ani orientáciu v zdrojových kódoch.



Obrázok 23: Grafické spracovanie dialógových okien.

Na obrázku č.22 je znázornené grafické spracovanie dialógových okien podľa základného motívu vzľadu.

5.4 Implementácia rozhrania pre spoluprácu WYSIWYG editorov s editorom anotácií

Rozhranie pre spoluprácu WYSIWYG editorov s editorom anotácií umožňuje jednotný spôsob práce s rôznymi typmi editorov. Ide o tzv. API (Application Programming Interface). Implementácia sa operia o návrh, ktorý je rozpísaný v kapitole 4.4.

Podpora WYSIWYG editorov bola vrámci rozsahu tejto práce zvolená iba na editor TinyMCE, no rozhranie je navrhnuté tak, aby bolo možné jednoduchým spôsobom pridávať nové typy editorov.

Implementácia si navyše vyžiadala znalosti TinyMCE API, ktoré je opísané v podkapitole 3.3.3.2. Vzhľadom na predošlú podrobnú analýzu sa pri implementácii nevyskytli žiadne vážnejšie problémy.

5.5 Implementácia rozšírenia pre editor TinyMCE

Hlavnou úlohou rozšírenia pre editor TinyMCE je spúšťanie a vypínanie užívateľského rozhrania editora anotácií z panela nástrojov editora TinyMCE. Návrh rozšírenia je popísaný v podkapitole 4.5.

Implementácia rozšírenia bola jednoduchá a rýchla, kedže nešlo o plugin s obšírnou funkcionalitou. Kostru programu tvorila osnova, ktorá je zobrazená v prílohe E. Vzhľadom na predošlú podrobnú analýzu tvorby rozšírení pre editor TinyMCE sa pri implementácii nevyskytli žiadne vážnejšie problémy.

6 Testovanie a ladenie

JavaScript bol odjakživa považovaný za jeden z najnáročnejších jazykov na ladenie, čo je dané jeho dynamickou povahou a rokmi bez kvalitných vývojových nástrojov. Tretia edícia ECMAScriptu zaviedla príkazy *try-catch* a *throw* spolu s najrôznejšími typmi chýb, čo vývojárom umožnilo reagovať na prípadné chyby už v okamžiku ich vzniku. Neskôr vzniklo aj mnoho ladiacich nástrojov učených pre webové prehliadače [11].

6.1 Typy chýb

Počas vykonávania kódu môže dôjsť k niekoľkým rôzným typom chýb. Každý typ chyby má odpovedajúci typ objektu, ktorý sa pri jej vzniku vyvolá. Štandard ECMA-262 definuje následujúce typy chýb [11, 24]:

- *Error* - Je základným typom, od ktorého sú odvodené všetky typy chýb. Chyba typu *Error* je v prehliadači vyvolávaná iba zriedka, slúži predovšetkým vývojárom pre vyvolávanie vlastných typov chýb.
- *EvalError* - Vyvoláva sa v okamžiku, kedy pri použití funkcie *eval()* dôjde k výnimke.
- *RangeError* - K chybe dôjde, ak je nejaké číslo mimo hranicu svojho rozsahu.
- *ReferenceError* - K tomuto typu chyby dochádza najčastejšie pri pokusoch o prístup k premennej, ktorá neexistuje.
- *SyntaxError* - Vyvoláva sa najčastejšie v okamžiku, kedy sa v reťazci JavaScriptu predanom funkcii *eval()* nachádza syntaktická chyba.
- *TypeError* - Ide o typ s najčastejším výskytom. K chybe dochádza, ak je nejaká premenná neočakávaného typu alebo ak dôjde k pokusu o prístup k neexistujúcej metóde.
- *URIError* - K výskytu chyby typu *URIError* môže dôjsť iba pri použití funkcie *encodeURI()* alebo *decodeURI()* pri zle vytvorenom URI.

6.2 Hlásenie chýb v prehliadačoch

Všetky známe webové prehliadače (Internet Explorer, Firefox, Safari, Chrome, Opera) hlásia nejakým spôsobom chyby JavaScriptu užívateľovi. Okrem prehliadača Internet Explorer, všetky ostatné prehliadače túto informáciu v štandardnom nastavení zakrývajú [11].

Internet Explorer pri vzniku chyby zobrazí v ľavom dolnom rohu prehliadača malú žltú ikonku. Pokiaľ jej zobrazenie nečakáte, ľahko ju možete prehliadnúť. Po kliknutí na ňu sa zobrazí dialógové okno s chybovou správou a ďalšími súvisiacími informáciami, ako je číslo riadku, číslo znaku, kód chyby a názov súboru.

Prehliadač Firefox vo svojom štandardnom nastavení nemení svoje užívateľské rozhranie pri vzniku chyby. Namiesto toho chybu spolu sa adresou URL a číslom riadku zaprotokoluje do chybovej konzoly.

Safari na systémoch Windows i Mac OS v štandardnom nastavení taktiež nezobrazuje chybové informácie užívateľovi. Je však možné aktivovať chybovú konzolu, ktorá ukladá informácie o chybách.

Opera podobne ako Firefox a Safari chybové správy v štandardnom nastavení skrýva. Chybová konzola zaznamenáva nielen informácie o chybách JavaScriptu, ale tiež (podobne ako Firefox) chyby a varovania pre kód v jazyku HTML, CSS, XML, XSLT a pod.

Prehliadač Chrome chyby štandardne nezobrazuje. Obsahuje však konzolu JavaScriptu (anglicky JavaScript console), do ktorej zaznamenáva chybovú správu, adresu URL a číslo riadku, na ktorom k chybe došlo.

6.3 Ladiace nástroje

V provopočiatkoch JavaScriptu sa za účelom ladenia aplikácie často využivala funckia *alert()*. V dnešnej dobe už však existujú plne vybavené ladiace nástroje s možnosťou krokovania kódu a kontroly stavu premenných [11].

Prehliadač Internet Explorer sa od verzie 8 dodáva spolu s ladiacim nástrojom JavaScriptu (Internet Explorer Debugger), ktorý je súčasťou vývojových nástrojov prehliadača. Pomocou neho je možné okrem iného zobrazovať hodnoty sledovaných premmených, sledovať premmenné v aktuálnom obore platnosti, pridávať do kódu zarážky (anglicky breakpoints) a ladiť kód krokovaním. Taktiež je možné využiť konzolu pre vyhodnocovanie kódu jazyka JavaScript v rámci kontextu stránky.

Firebug[2] patrí medzi najobľúbenejšie nástroje na ladenie JavaScriptu. Je určený pre prehliadač Firefox a ponúká kompletnú funkcionalitu pre ladenie aplikácií napísaných v jazyku JavaScript. Podobne ako Internet Explorer Debugger, aj Firebug umožňuje vytváranie záražiek, krokovanie kódu a sledovanie hodnôt zvolených premenných.

Ďalším ladiacim nástrojom je Drosera [55], ktorý je určený pre WebKit, čo je vykresľovací subsystém používaný v prehliadači Safari. Drosera sa síce s prehliadačom Safari nedodáva, no jej stiahnutie je umožnené bezplatne. Z hľadiska základnej funkcionality sa Drosera vyrovná predchádzajúcim nástrojom.

Aj prehliadač Opera má svoj vlastný ladiaci nástroj, ktorý sa dodáva od verzie 9.5 [11]. Aktivovať ho je možné v ponuke Nástroje (anglicky Tools) > Pokročilé (anglicky Advanced) > Vyvojárske nástroje (anglicky Developer Tools). Jeho funkcionalita je podobná ako v ostatných spomenutých nástrojoch.

6.4 Testovanie

Vývoj aplikácie editora anotácií prebiehal spôsobom, ktorý sa označuje ako vývoj riadený testami. Pri vytváraní každej funkcie sa najprv definovali jej vstupy a výstupy, následne sa vytvorila sada testovacích prípadov, ktoré predvídajú výsledky pre príslušnú sadu vstupov. Až po vytvorení testovacích prípadov sa dokončí telo vytváranej funkcie, ktoré úspešne splní testovacie prípady.

Na testovanie aplikácií existujú rôzne testovacie frameworky. Pri vývoji editora anotácií bol použitý framework YUI Test[3] od spoločnosti Yahoo!. Framework YUI Test je súčasťou knižnice Yahoo! UI, ale chová sa úplne rovnako aj pri použití inej knižnice JavaScriptu. Umožňuje vytvárať a spúšťať testovacie prípady a detekovať a zaznamenávať chyby. Testovacie prípady je možné zhlukovať do veľkých sád testov, ktoré umožňujú jednotným spôsobom spúšťať viacero testov v jednom behu. Obrovskou výhodou tohoto frameworku je podpora testovania asynchronného volania metód, vďaka ktorému je možné testovať spätné volania metód Ajaxu. Vo väčšine prehliadačov je tiež možné simulovať udalosti modelu DOM [11].

Následujúca ukážka zobrazuje vytvorenie testovacieho prípadu pre funkciu *AEd.createNamespace*, ktorá umožňuje vytvorenie menného priestoru podľa zadaného parametra.

[2] *http://www.getfirebug.com*

[3] *http://developer.yahoo.com/yui/yuitest/*

V tomto konkrétnom prípade sa pokúšame vytvoriť menný priestor s názvom *AEd.newNS*, pričom výsledok následne testujeme na jeho existenciu.

```
YAHOO.AEd.test.AEdCreateNamespace = new YAHOO.tool.TestCase({
name: "AEdCreateNamespace",
testObject: function() {
    var assert = YAHOO.util.Assert;
    var ns = "newNS";
    var result = AEd.createNamespace(ns);
    assert.isObject(result);
}
});
```

Podobným spôsobom boli vytvorené aj ďalšie testovacie prípady pre funkciu *AEd.createNamespace* i pre ostatné vytvorené funkcie.

7 Záver

Táto diplomová práca rozširuje semestrálny projekt, ktorého obsahom bol výber štyroch WYSIWYG textových editorov vytvorených v jazyku JavaScript a analýza možností ich rozšírenia. Súčasťou práce bol aj návrh editora anotácií ako modulu pre zvolené editory. Vrámci diplomovej práce bol editor anotácií implementovaný pre WYSIWYG editor TinyMCE.

Na výber WYSIWYG editorov boli kladené tieto kritériá:

- Popularita a široké nasadenie editora v praxi.
- Dostupnosť editora s otvorenou a nekomerčnou licenciou.
- Výber dvoch robustných a dvoch jednoduchých editorov.

Výsledkom výberu sú editory TinyMCE, CKEditor, NicEdit a jWYSIWYG, ktorých detailný popis je v kapitole 3. Analýza editorov bola z väčšej časti zameraná na princípy tvorby rozšírení. Editory TinyMCE a CKEditor disponujú robustným API, preto je ich popis na abstraktnejšej úrovni. Pri jednoduchých editoroch NicEdit a jWYSIWYG bolo možné popísať ich API takmer úplne.

Ďalšou časťou práce bol návrh a implementácia editora anotácií vyvýjaného ako súčasť systému pre kolaboratívne anotovanie v reálnom čase, ktorý je predmetom dizertačnej práce vytváranej na FIT VUT [47]. Kapitola č. 4 pozostávala z návrhu:

- Aplikácie editora anotácií.
- Knižnice pre užívateľské rozhranie.
- Rozhrania pre spoluprácu WYSIWYG editorov s editorom anotácií.
- Rozšírení pre zvolené WYSIWYG editory.
- Obrazoviek s popisom chovania.

V tejto práci boli diskutované aj vybrané problémy a riešenia z implementačnej časti a tiež popis ladenia a testovania aplikácie.

Medzi možné rozšírenia editora anotácií by mohli patriť:

- Podpora nových typov WYSIWYG textových editorov.
- Rozšírenie funkcionality editora o možnosti, ktoré ponúku nové verzie serverovej časti pre kolaboratívne anotovanie v reálnom čase.
- Rozšírenie knižnice pre užívateľské rozhranie o nové komponenty.
- Rozšírenie rozhrania pre spoluprácu WYSIWYG editorov s editorom anotácií o plnú podporu metód a udalostí, ktoré ponúkajú súčasné implementácie WYSIWYG editorov.
- Vytvorenie nových jazykových lokalizácií.

Vytvorený editor anotácií je prvým existujúcim klientom, ktorý dokáže spolupracovať so systémom spomínaným v [56]. Z toho dôvodu nebolo možné porovnať existujúce riešenie s alternatívnymi klientmi. Na vývoji editora anotácií plánujem v rámci výskumu v skupine NLP[4] pokračovať.

[4] *http://www.fit.vutbr.cz/research/groups/nlp/*

Literatúra

[1] Hruška, T. *Internetové aplikace (WAP) I. část Internet a WWW (Studijní opora)*. Brno, 2007.

[2] W3C. *What is Hypertext [online]*. [cit. 2010-12-26]. URL: <http://www.w3.org/WhatIs.html>.

[3] W3C. *HTTP - Hypertext Transfer Protocol [online]*. Posledná modifikácia: 8. marca 2010. [cit. 2010-12-26]. URL: <http://www.w3.org/Protocols/>.

[4] Robinson, D., Coar, K. *The Common Gateway Interface (CGI) Version 1.1 (RFC3875)* [online]. Posledná modifikácia: október 2004. [cit. 2010-12-27]. URL: <http://tools.ietf.org/html/rfc3875>.

[5] W3C. *HTML & CSS [online]*. [cit. 2010-12-27]. URL: <http://www.w3.org/standards/webdesign/htmlcss>.

[6] Hruška, T., Burget, R. *Internetové aplikace (WAP) II. část SGML, HTML, CSS, DOM (Studijní opora)*. Brno, 2007.

[7] W3C. *HTML Current Status [online]*. [cit. 2010-12-27]. URL: <http://www.w3.org/standards/techs/html#w3c_all>.

[8] W3C. *CSS Current Status [online]*. [cit. 2010-12-28]. URL: <http://www.w3.org/standards/techs/css#w3c_all>.

[9] W3C. *XML Essentials [online]*. [cit. 2010-12-28]. URL: <http://www.w3.org/standards/xml/core>.

[10] W3C. *XML Current Status [online]*. [cit. 2010-12-28]. URL: <http://www.w3.org/standards/techs/xml#w3c_all>.

[11] Zakas, N. *JavaScript pro webové vývojáře*. Computer Press, Brno, 2009. ISBN 987-80-251-2509-0.

[12] Hruška, T., Máčel, L., Kužela, A. *Internetové aplikace (WAP) V. část AJAX (Studijní opora)*. Brno, 2007.

[13] Zakas, N., McPeak, J., Fawcett, J. *Ajax profesionálne*. Zoner Press, Brno, 2007. ISBN 978-80-86815-77-0.

[14] The jQuery Project. *JQuery is a new kind of JavaScript Library. [online]*. [cit. 2010-12-29]. URL: <http://jquery.com/>.

[15] Kirchoff, B. *NicEdit. [online]*. [cit. 2011-01-03]. URL: <http://nicedit.com/>.

[16] W3C. *On SGML and HTML [online]*. [cit. 2011-01-05]. URL: <http://www.w3.org/TR/html4/intro/sgmltut.html>.

[17] W3C. *Transformation [online]*. [cit. 2011-01-05]. URL: <http://www.w3.org/standards/xml/transformation#xslt>.

[18] W3C. *XQuery 1.0: An XML Query Language (Second Edition) [online]*. [cit. 2011-01-05]. URL: <http://www.w3.org/TR/xquery/>.

[19] W3C. *XML Path Language (XPath) 2.0 (Second Edition) [online]*. Posledná modifikácia: 3. januára 2011. [cit. 2011-01-05]. URL: <http://www.w3.org/TR/xpath20/>.

[20] W3C. *XQuery 1.0 and XPath 2.0 Data Model (XDM) (Second Edition) [online]*. Posledná modifikácia: 14. decembra 2010. [cit. 2011-01-05]. URL: < http://www.w3.org/TR/xpath-datamodel/>.

[21] Håkon Wium Lie. *Håkon Wium Lie [online]*. [cit. 2011-01-05]. URL: http://people.opera.com/howcome/>.

[22] W3C. *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification [online]*. Posledná modifikácia: 7. decembra 2010. [cit. 2011-01-05].

URL: <http://www.w3.org/TR/CSS21/>.
[23] Hruška, T. *Internetové aplikace (WAP) VI. Programování klienta (JavaScript) (Studijní opora).* Brno, 2007.
[24] ECMA International. *Standard ECMA-262. ECMAScript Language Specification (5th edition) [online]*. Posledná modifikácia: december 2009. [cit. 2011-01-06]. URL:<http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-262.pdf>.
[25] Netscape Communications Corporation. *LiveWire Developer's Guide [online]*. [cit. 2011-01-06]. URL: < http://docsrv.sco.com/INT_LiveWire/CONTENTS.html>.
[26] Wallnoefer, H. *About Rhinola [online]*. Posledná modifikácia: 18. septembra 2007. [cit. 2011-01-06]. URL: < http://mod-gcj.sourceforge.net/rhinola.html>.
[27] Mozilla Foundation. *Rhino: JavaScript for Java [online]*. [cit. 2011-01-06]. URL: <http://www.mozilla.org/rhino/>.
[28] Komunita Wikipedie. *JavaScript [online]*. Posledná modifikácia: 6. januára 2011. [cit. 2011-01-06]. URL: <http://en.wikipedia.org/wiki/JavaScript>.
[29] Mozilla Developer Network. *JavaScript [online]*. [cit. 2011-01-06]. URL: <https://developer.mozilla.org/en/JavaScript>.
[30] Garrett, J. *Ajax: A New Approach to Web Applications [online]*. [cit. 2011-01-06]. URL: <http://www.adaptivepath.com/ideas/essays/archives/000385.php>.
[31] Russell, A. *Comet: Low Latency Data for the Browser [online]*. [cit. 2011-01-06]. URL: <http://infrequently.org/2006/03/comet-low-latency-data-for-the-browser/>.
[32] Kolektív autorov. *Hypertext Transfer Protocol -- HTTP/1.1 (RFC2616) [online]*. Posledná modifikácia: jún 1999. [cit. 2011-01-06]. URL: <http://tools.ietf.org/html/rfc2616>.
[33] Kolektív autorov. *jWYSIWYG [online]*. [cit. 2011-01-07]. URL: <https://github.com/akzhan/jwysiwyg>.
[34] The jQuery Project. *JQuery Events. [online]*. [cit. 2011-01-07]. URL: <http://api.jquery.com/category/events>.
[35] Moxiecode Systems AB. *TinyMCE. [online]*. [cit. 2011-01-07]. URL: <http://tinymce.moxiecode.com/>.
[36] Moxiecode Systems AB. *TinyMCE Configuration. [online]*. [cit. 2011-01-07]. URL: <http://tinymce.moxiecode.com/wiki.php/Configuration>.
[37] Moxiecode Systems AB. *TinyMCE Creating a plugin. [online]*. [cit. 2011-01-07]. URL: <http://tinymce.moxiecode.com/wiki.php/Creating_a_plugin>.
[38] Lunarmedia. *Javascript Compressor. [online]*. [cit. 2011-01-07]. URL: <http://javascriptcompressor.com/>.
[39] Moxiecode Systems AB. *TinyMCE API 3.x. [online]*. [cit. 2011-01-07]. URL: <http://tinymce.moxiecode.com/wiki.php/API3:tinymce.api.3.x>.
[40] Moxiecode Systems AB. t*inymce.Editor. [online]*. [cit. 2011-01-07]. URL: <http://tinymce.moxiecode.com/wiki.php/API3:class.tinymce.Editor>.
[41] CKSource.*CKEditor. [online]*. [cit. 2011-01-08]. URL: <http://ckeditor.com>.
[42] CKSource. *Namespace CKEDITOR.config. [online]*. [cit. 2011-01-08]. URL: <http://docs.cksource.com/ckeditor_api/symbols/CKEDITOR.config.html>.
[43] Wu, G. *CKEditor Plugin Development. [online]*. [cit. 2011-01-09]. URL: <http://www.voofie.com/content/2/ckeditor-plugin-development>.
[44] CKSource. *CKPackager. [online]*. [cit. 2011-01-09]. URL: <http://svn.fckeditor.net/CKPackager/trunk/bin/>.

[45] CKSource. *CKEditor 3 JavaScript API. [online]*. [cit. 2011-01-09]. URL: <http://docs.cksource.com/ckeditor_api/index.html>.

[46] CKSource. *Event driven. [online]*. [cit. 2011-01-09]. URL: <http://docs.cksource.com/FCKeditor_3.x/Design_and_Architecture/Event_Driven>.

[47] Dytrych, J. *Webové služby a komponentní technologie (dizertačná práca)*. Brno, FIT VUT, 2011.

[48] Steigerwald, D. *Třídy, dědičnost a OOP v Javascriptu - I [online]*. Posledná modifikácia: 15. marca 2010. [cit. 2011-01-09]. URL: <http://zdrojak.root.cz/clanky/oop-v-javascriptu-i/>.

[49] Chen, D. *Developing Revolutionary Web Applications using Comet and Ajax.* Sun Tech Days. A Worldwide developer conference. 2008-2009.

[50] Herout, P. *Učebnice jazyka Java*. KOPP, České Budejovice, 2001. ISBN 80-7232-115-3.

[51] Komunita Wikipedie. *Java (programming language)[online]*. Posledná modifikácia: 9. januára 2011. [cit. 2011-01-09]. URL: < http://en.wikipedia.org/wiki/Java_%28programming_language%29>.

[52] Oracle. *Project Grizzly. [online]*. [cit. 2011-01-09]. URL: <http://grizzly.java.net/>.

[53] Oracle. *GlassFish. [online]*. [cit. 2011-01-09]. URL: <http://glassfish.java.net/>.

[54] W3C. *Document Object Model Range [online]*. Posledná modifikácia: 13.novembra 2000. [cit. 2011-01-10]. URL: <http://www.w3.org/TR/DOM-Level-2-Traversal-Range/ranges.html>.

[55] Hatcher, T. *Introducing Drosera [online]*. Posledná modifikácia: 28. júna 2006. [cit. 2011-05-18]. URL: <http://www.webkit.org/blog/61/introducing-drosera/>.

[56] Dytrych, J. *4A Framework: Annotations Anywhere, Annotations [online]*. Posledná modifikácia: 28. apríla 2011. [cit. 2011-05-22]. URL: <http://keg.vse.cz/_slides/dytrych.pdf>.

Zoznam príloh

A Ukázka strukturované anotace

```
<rdf:Description rdf:about="http://example.com/annotations/123456">
  <rdf:type rdf:resource="http://example.com/types/g01/annotation/task"/>
  <a:dateTime rdf:value="2011-01-01T:20:00:00Z" />
  <a:author id="http://example.com/authors/123456" name="Jaroslav Dytrych"
            address="idytrych@fit.vutbr.cz"/>
  <a:source rdf:resource="http://example.com/documents/getDoc?id=123456"/>
  <a:fragment>
    <a:path>/html/body/div[@id='container']/div[@id='main']/div[@id='post1']/DIV[2]/p[1]</a:path>
    <a:offset>22</a:offset>
    <a:length>32</a:length>
    <a:annotatedText>Fakulta informačních technologií</a:annotatedText>
  </a:fragment>
  <a:content>
    <![CDATA[
        ...
      ]]>
  </a:content>
  <a:attribute name="place" type="geoPoint">
    <geo:Point> <geo:lat>55.701</geo:lat> <geo:long>12.552</geo:long> </geo:Point>
  </a:attribute>
  <a:attribute name="date" type="nestedAnnotation">
    <rdf:Description rdf:about="http://example.com/annotations/123457">
      <rdf:type rdf:resource="http://example.com/types/g01/annotation/description"/>
      <a:dateTime rdf:value="2011-01-01T:20:00:00Z" />
      <a:author id="http://example.com/authors/123456" name="Jaroslav Dytrych"
                address="idytrych@fit.vutbr.cz"/>
      <a:source rdf:resource="http://example.com/documents/getDoc?id=123456"/>
      <a:fragment>
        <a:path>/html/body/div[@id='container']/div[@id='main']/div[@id='post1']/p[1]</a:path>
        <a:offset>92</a:offset>
        <a:length>14</a:length>
        <a:annotatedText>14. ledna 2011</a:annotatedText>
      </a:fragment>
      <a:content>
      <![CDATA[
            ...
          ]]>
      </a:content>
      <a:attribute name="date" type="DateTime" rdf:value="2011-01-14T:00:00:00Z"/>
    </rdf:Description>
  </a:attribute>
  <a:attribute name="reason" type="annotationLink" uri="http://example.com/annotations/1234567"/>
</rdf:Description>
```

B Protokol pro přenos anotací mezi klientem a serverem

B.1 Správa sezení

Správa sezení zahrnuje dohodu na verzi protokolu, přihlášení a odhlášení uživatele.

Nejprve klient zahájí spojení tak, že zašle na server zprávu `connect`, kde v atributu uvede nejvyšší verzi protokolu, kterou může využít:

```
<connect protocolVersion="1.0"/>
```

Server odpoví chybou nebo následující zprávou:

```
<connected protocolVersion="1.0" sessionID=""/>
```

V atributu `protocolVersion` server uvede verzi protokolu, kterou bude komunikovat. Server by měl komunikovat stejnou verzí jako klient, nebo nejnižší verzí, která je zpětně kompatibilní s verzí klienta. Pokud klient nabídne novější verzi než server, server použije nejnovější verzi, kterou zná. Pokud klient zjistí, že jeho verze není zpětně kompatibilní, musí přepnout na verzi serveru (případně jinou s ní kompatibilní), nebo se odpojit. Pokud verze protokolu není podporována a server vrátí chybovou zprávu, klient se může pokusit o spojení se starší verzí protokolu.

Vzhledem k tomu, že novější verze protokolu může být zpětně kompatibilní, klient a server mohou implementovat různé verze protokolu. Pokud server či klient umožňuje využít novou funkcionalitu, protistrana se starší verzí protokolu ji nevyužije a zaslané elementy či atributy navíc bude ignorovat. Pokud nová verze protokolu nebude zpětně kompatibilní, server či klient, který ji implementuje, musí spojení s danou kombinací verzí protokolu odmítnout.
Ukončení spojení je signalizováno následující zprávou:

```
<disconnect/>
```

Ve zprávě `connected` server zašle také id sezení (atribut `sessionID`), které bude klient zasílat se všemi následujícími zprávami v atributu `id` elementu session:

```
<session id=""/>
```

Přihlášení je realizováno následující zprávou:

```
<login user="" password=""/>,
```

kde `user` je uživatelské jméno nebo e-mail uživatele a `password` je heslo uživatele. V případě úspěšného přihlášení server na tuto zprávu odpoví seznamem parametrů nastavení a následující zprávou:

```
<logged id="" name=""/>,
```

kde `id` je identifikátor uživatele a `name` jeho zobrazované jméno. V případě neúspěšného přihlášení server odpoví chybovou zprávou.
Odhlášení bude prováděno zprávou `<logout/>`.

Přihlášení lze provést současně se zahájením spojení a dohodou na verzi protokolu, odhlášení společně s jeho ukončením.

B.2 Uživatelé a skupiny

Pro získání informací o profilech uživatelů zašle klient zprávu:

```
<queryPersons filter="" withGroups=""/>
```

Ve filtru lze pro výběr pole využít klíčové slovo `id`, `e-mail` či `name` následované dvojtečkou. Při filtrování dle více polí budou filtry odděleny středníkem. Pokud bude uveden i volitelný atribut `withGroups` s hodnotou `true`, budou v informacích o profilech zahrnuty i informace o členství ve skupinách. Server na tuto zprávu odpoví zprávou:

```
<persons>
  <person id="" login="" name="" email="" photoURI=""/>
</persons>
```

Atribut name obsahuje celé jméno uživatele, `photoURI` slouží k získání URI fotografie, která bude u uživatele zobrazena. Pokud byly požadovány informace o skupinách, bude každá značka `person` obsahovat i značku `userGroups` (viz níže). Značka `person` by mohla obsahovat i další značky a textový obsah s informacemi o profilu uživatele.

Pro získání informací o skupinách uživatelů zašle klient zprávu:

```
<queryUserGroups filter="" withPersons=""/>
```

Ve filtru lze využít URI skupiny nebo její název. Pokud bude uveden i volitelný atribut `withPersons` s hodnotou `true`, budou v informacích o skupinách zahrnuty i informace o jejich členech. V názvu lze využít zástupné symboly „*“ (libovolný počet libovolných znaků). Server na tuto zprávu odpoví:

```
<userGroups>
  <group uri="" name=""/>
</userGroups>
```

Pokud byly požadovány informace o členech skupin, v každé značce `group` bude obsažena i značka `persons` (viz výše).
Pro přihlášení ke skupině uživatelů klient zašle následující zprávu:

```
<join group=""/>
```

kde atribut group obsahuje URI skupiny. K odhlášení uživatele ze skupiny slouží zpráva:

```
<leave group=""/>
```

B.3 Řízení odběru anotací

Klient může přijímat pouze anotace zvolených typů ze zvolených zdrojů. Zdrojem může být jiný uživatel nebo URI, který identifikuje anotační server, skupinu uživatelů či jiný obecný zdroj.
Klient se k odběru anotací přihlašuje následující zprávou:

```
<subscribe>
  <source type="" user=""/>
  <source type="" uri=""/>
  <source type=""/>
  <source user=""/>
  <source uri=""/>
</subscribe>
```

Elementů `source` může být libovolné nenulové množství a mohou mít kombinace parametrů, které jsou uvedeny výše. Parametr `user` identifikuje uživatele, `uri` obecný zdroj anotací. Parametr `type` udává typ anotací, přičemž s typem jsou automaticky vybrány i všechny podtypy. V typu může být využit i zástupný symbol „*“, který nahrazuje libovolný počet libovolných znaků.

Pokud není uveden typ, budou přijímány všechny typy anotací (dle skupin, ve kterých se uživatel nachází) z daného zdroje. Pokud není uveden zdroj, budou přijímány všechny anotace daného typu.
K odhlášení může klient využít zprávu `unsubscribe`:

```
<unsubscribe>
  <source type="" user=""/>
  <source type="" uri=""/>
  <source type=""/>
  <source user=""/>
  <source uri=""/>
</unsubscribe>
```

Pro odhlášení platí stejná pravidla jako pro přihlášení (libovolný počet elementů `source`, atd.). Klient automaticky odebírá všechny anotace od svého přihlášeného uživatele, pokud se od jejich odběru explicitně neodhlásí.

B.4 Synchronizace dokumentu

Synchronizace dokumentu je proces, při kterém server získá kopii aktuální verze anotovaného dokumentu. Pokud server tento dokument získá poprvé, uloží si jej a vrátí klientovi adresu uložené verze, kterou bude klient využívat v anotacích. Pokud má server dokument uložený, porovná novou verzi s uloženou verzí, a pokud se shodují, zašle klientovi adresu uložené verze. Pokud se dokumenty shodují částečně, server vyhodnotí změny. Pokud změny neovlivní žádné anotace, dokument se transparentně aktualizuje (pro klienta stejné jako shoda dokumentů). Pokud by změny ovlivnily některé anotace, server zašle klientovi varování, že některé anotace musely být aktualizovány, a synchronizaci dokončí. V případě zásadnějších změn či zneplatnění anotací dojde k chybě synchronizace a uživatel se musí rozhodnout, zda synchronizaci dokončí a zneplatní tak některé či všechny anotace (vymaže či přesune na úroveň celého dokumentu), nebo nedokončí a bude anotovat uloženou (starší) verzi dokumentu či jiný dokument.

Protože klient může být i jednoduchý textový editor, který nepracuje se strukturovaným textem, server musí podporovat i linearizaci dokumentu. V tomto případě klient dokument linearizuje na prostý text a zašle jej serveru v linearizované podobě. Pokud má server strukturovanou podobu, linearizuje ji a porovná se zaslanou. Pokud se linearizované verze neshodují, dojde k chybě synchronizace. Pokud se shodují, synchronizace bude dokončena a server bude každou následně zaslanou anotaci upravovat pro strukturovanou verzi dokumentu.
Syntaxe:

```
<synchronize resource="http://example.com/documents/doc1.txt" linearize="false" overwrite="false">
  <content>
    <![CDATA[
      ...
    ]]>
  </content>
</synchronize>
```

Klient pošle serveru kopii anotovaného dokumentu a adresu, ze které pochází. Parametr `resource` udává umístění zdroje (např. URI anotované webové stránky). Element content obsahuje obsah daného dokumentu.

Nepovinný parametr `linearized` udává, zda klient pracuje s linearizovanou verzí dokumentu. Výchozí hodnota je false.

Nepovinný parametr `overwrite` umožňuje vynutit synchronizaci v případě, kdy je dokument s daným URI na serveru uložen, ale jeho obsah se neshoduje. Server v tomto případě musí nahradit uložený dokument a upravit (přesunout na úroveň celého dokumentu a doplnit textový obsah o informaci o změně) či vymazat všechny anotace. Klient by tento atribut neměl využít při prvním pokusu o synchronizaci. Jeho použití musí být schváleno uživatelem. Pokud se obsah textu shoduje, server atribut ignoruje. Výchozí hodnota je false.

Při úspěšné synchronizaci server odpoví zprávou:

```
<synchronized resource=""/>
```

Atribut `resource` obsahuje URI kopie anotovaného dokumentu, která je uložena na serveru. Tento URI musí klient využívat v anotacích.

Pokud v průběhu práce dojde k situaci, kdy se obsah anotovaného fragmentu neshoduje s obsahem, který server nalezne na dané pozici v dokumentu, dojde k tzv. rozsynchronizování. V tomto případě server zašle chybovou zprávu a zprávu `<resynchronize/>` (element bez obsahu a atributů), čímž požádá o resynchronizaci.
Klient provádí resynchronizaci zasláním obsahu v následující zprávě:

```
<resynchronize>
  <content>
    <![CDATA[
      ...
    ]]>
  </content>
</resynchronize>
```

Po resynchronizaci je vždy nutné znovu načíst všechny anotace. Server je tedy automaticky zašle v odpovědi.
Pokud klient provádí modifikace dokumentu, musí každou změnu zaslat na server:

```
<textModification path="" offset="" length="">
  <![CDATA[
    Nový obsah vybraného fragmentu ...
  ]]>
</textModification>
```

Atribut `path` udává XPath uzlu DOM dokumentu, ve kterém byla změna provedena. Atributy `offset` a `length` udávají offset a délku změněného fragmentu. V těle elementu `<textModification/>` je potom uveden nový obsah fragmentu. Pokud je vložen nový fragment, délka původního fragmentu je nulová. Pokud je fragment vymazán, element je prázdný. Server následně tuto zprávu rozešle všem klientům pracujícím se stejným dokumentem.

B.5 Přenos typů anotací

Přenos typů anotací probíhá obousměrně. Pokud je přidán, upraven či vymazán typ, klient tuto změnu okamžitě zašle serveru a ten ji rozešle všem ostatním klientům, kterých se týká. Klient však nemusí udržovat kompletní strom typů, ale může mít načtenou pouze jeho část. V tomto případě může server buď zasílat všechny změny, nebo může udržovat informaci o tom, které části stromu má klient načtené, a zasílat pouze informace o změnách v těchto částech. Server musí vždy udržovat kompletní strom typů dané skupiny uživatelů (všechny změny jsou mu zasílány).
Klient o část stromu typů žádá zprávou:

```
<queryTypes filter=""/>
```

Atribut `filter` umožňuje získání určitého podstromu typů. Jedná se o linearizovaný název či URI typu se zástupnými symboly „*“ (libovolný počet libovolných znaků). Filtr tedy umožňuje vybrat podstrom nebo množinu typů, jejichž název či URI obsahuje daný text.

Pokud klient zažádá o strom typů, server na to odpoví zprávou s přidáním typů (viz níže). Pokud filtru nevyhovuje žádný typ, server zašle prázdný seznam typů.
Přenos typů anotací je prováděn následující zprávou:

```
<types>
  <add>
    <type name="" ancestor="" uri="" group="">
      <attribute name="" type="" required=""/>
    </type>
  </add>
  <change/>
  <remove/>
</types>
```

Element `types` obsahuje:

- element `add`, pokud byl přidán typ,
- element `change`, pokud byl upraven typ,
- element `remove`, pokud byl vymazán typ.

V každém ze tří výše uvedených elementů může být obsažen libovolný počet elementů `type`. Atributy tohoto elementu jsou `name` (název typu), `ancestor` (URI rodičovského typu), `uri` (URI typu) a `group` (URI skupiny uživatelů, které typ patří). Pokud je URI rodičovského typu prázdný, jedná se o základní typ. URI jednoznačně identifikuje typy. Pokud není uvedena skupina, určí se z URI typu, podle rodičovského typu nebo podle výchozí skupiny uživatele.

U každého typu mohou být definovány i výchozí atributy. Tyto jsou potom obsaženy v elementech `attribute`. Každý atribut má název (`name`) a typ (`type`). Pokud se jedná o jednoduchý datový typ, je uveden název tohoto typu. Pokud se jedná o vnořenou anotaci či odkaz na anotaci, je uveden očekávaný typ této anotace (informace o vnoření či odkazování zde není potřebná, protože obě varianty jsou zde významově ekvivalentní). Pokud je atribut povinný, má element `attribute` i atribut `required` s hodnotou `true`.

Název typu nelze upravit jinak, než smazáním starého typu a přidáním nového.

B.6 Přenos anotací

Přenos anotací je prováděn obdobně jako přenos typů:

```
<annotations>
  <add>
    <annotation/>
  </add>
  <change/>
  <remove/>
</annotations>
```

Každé přidání, úprava či vymazání anotace jsou ihned zaslány na server. V elementu `annotations` jsou dle provedené operace obsaženy elementy `add` (přidané) `change` (upravené) a `remove` (vymazané). V každém z těchto elementů může být obsažen libovolný počet elementů `annotation`.

Server každou změnu ihned zašle klientům, kterých se týká (u kterých dotčená anotace patří mezi odebírané). Pokud je přidána anotace, je tato anotace vždy zaslána i klientovi, který ji přidal, aby získal přidělený identifikátor anotace.

Po synchronizaci či resynchronizaci dokumentu jsou klientovi jako přidané automaticky zaslány všechny anotace, které patří k tomuto dokumentu a vyhovují klientem definovaným požadavkům na odběr anotací (zdroje a typy).

Pokud klient potřebuje znovu načíst některou anotaci (např. po neúspěšném pokusu o editaci) či všechny anotace, může serveru zaslat jednu z následujících dvou variant zprávy:

```
<reload uri="http://example.com/annotations/123456"/>
<reload all="true"/>
```

Atribut `uri` u první varianty zprávy udává URI požadované anotace, atribut `all` u druhé varianty udává, že mají být znovu zaslány všechny anotace.

B.7 Nabízení anotací

Server může klientovi nabídnout automaticky vygenerované anotace k danému dokumentu či jeho části. Klient o nabídku anotací zažádá následovně:

```
<suggestAnnotations path="" offset="" length="" type=""/>
```

Atributy `path`, `offset` a `length` udávají cestu (XPath), offset a délku fragmentu dokumentu, ke kterému mají být nabídnuty anotace. Pokud je uvedena pouze cesta, budou nabídnuty anotace k celému uzlu DOM dokumentu. Pokud není uveden žádný z těchto atributů, budou nabídnuty anotace k celému dokumentu. Volitelný atribut type udává požadovaný typ nabízených anotací. S tímto typem budou současně nabízeny i všechny jeho podtypy.
Server odpoví nabídkou anotací:

```
<suggestions>
  <annotation tmpId="" confidence=""/>
</suggestions>
```

V elementu `suggestions` může být libovolný počet anotací (elementů `annotation`). Atribut `confidence` udává odhadnutou míru jistoty anotace v procentech. Hodnota může být využita klientem při automatickém přijímání a odmítání anotací. Anotace v nabídce nemají trvalý identifikátor, ale pouze dočasný (`tmpId`). Dočasný identifikátor slouží k informování serveru o manipulaci s nabídkami a vyřazení anotace z nabídky při její aktualizaci.

Pokud klient chce anotaci potvrdit (at' už akcí uživatele či automaticky na základě uživatelského nastavení), vrátí ji serveru mezi přidávanými anotacemi (ve zprávě `annotations`), přičemž dané anotaci (elementu `annotation`) přidá atributy `confirmed` a `tmpId`. Atribut `tmpId` udává dočasný identifikátor potvrzené anotace. Atribut `confirmed` udává způsob potvrzení a může nabývat následujících hodnot:

- `manually` - uživatelem potvrzená anotace,
- `manuallyEdited` - uživatel anotaci editoval a uložil,
- `automatically` - automaticky potvrzená anotace (dle nastavení doplňku),
- `automaticallyEdited` - automaticky potvrzená anotace s provedením automatických úprav.

Pokud dojde ke změně dokumentu, může být potřeba upravit nabídku anotací. V tomto případě server okamžitě zašle aktualizaci nabídky anotací. Pro jednodušší implementaci klienta není podporována úprava nabídnutých anotací. Pokud se některá anotace změní, je odstraněna a server nabídne novou verzi. V elementu `suggestions` tedy může být i libovolný počet elementů `delete`:

```
<suggestions>
  <annotation tmpId="" confidence=""/>
  <delete tmpId=""/>
</suggestions>
```

Pokud klient nemá anotaci s daným dočasným identifikátorem, element `delete` ignoruje. Pokud uživatel (či klient, dle nastavení) některou nabídku odmítne, klient zašle na server zprávu:

```
<refusedSuggestions>
  <suggestion tmpId="" method=""/>
</refusedSuggestions>
```

Atribut `method` udává způsob odmítnutí a může mít jednu z následujících hodnot:

- `manually` - odmítnutí uživatelem,
- `automatically` - automatické odmítnutí na základě nastavení.

Pokud klient nechce přijímat další aktualizace nabídky anotací, zažádá server o nabídky anotací k fragmentu dokumentu s nulovou délkou.

B.8 Přenos nastavení

Nastavení je seznam položek, které mají název a řetězcovou hodnotu. Nastavení lze rozdělit na nastavení serveru a nastavení klienta s tím, že nastavení klienta budou mít prefix „`Client`“ (např.: „`ClientAnnotationTypeColor:Animal ->People->Employee`“ s hodnotou „`green`“). Při zobrazení uživateli se potom některá (známá) nastavení zpracují a zobrazí ve formulářích a ostatní se vypíšou v tabulce pro ostatní nastavení, kde je uživatel může měnit.

Konkrétní položky nastavení závisí na implementaci serveru a klienta. Aby nedocházelo k problémům při využití více různých klientů jedním uživatelem, měly by být názvy položek nastavení prefigovány i typem a názvem klienta (např. „`ClientFFExtAnnotFox`“ bude prefix pro rozšíření Firefoxu nazvané AnnotFox) nebo by měly být takové, aby byl jejich význam zcela zřejmý (např. „`ClientDefaultAnnotationType`“ pro výchozí typ anotace).

Vzhledem k tomu, že je předpokládán malý počet položek a malá frekvence přenášení, vždy je přenášen kompletní seznam položek nastavení. Neuvedení položky tedy povede k jejímu odstranění (je-li možné). Pokud položku není možné odstranit (např. nastavení serveru), bude nastavena na výchozí hodnotu.
Nastavení se přenáší zprávou:

```
<settings>
  <param name="" value=""/>
</settings>
```

Elementů `param`, které tvoří jednotlivé položky, může být libovolný (i nulový) počet. Atribut `name` obsahuje název položky, atribut `value` řetězcovou hodnotu položky.

B.9 Chyby a varování

Chybové zprávy slouží k informování klienta o chybě. Chybová zpráva obsahuje číslo chyby (`number`) a její textový obsah (v elementu `message`). Může obsahovat i doplňující informace, které se týkají konkrétní chyby. Při chybě oprávnění při přístupu k anotacím bude obsažena informace o tom, ke kterým zdrojům byl odepřen přístup. Při nezdařené operaci s existující anotací (úprava či mazání) musí chybová zpráva obsahovat informaci o tom, které anotace se týká (kterou anotaci je třeba znovu načíst). Při problému s atributy musí obsahovat i informaci o tom, kterých atributů se týká.

Textový obsah chybových zpráv bude lokalizován do jazyka nastaveného parametrem „`ServerLanguage`“, který bude mít hodnoty dle ISO 639-2 [27] (varianty pro bibliografické účely). Pokud tento parametr nebude nastaven, zprávy budou v angličtině.
Syntaxe:

```
<error number="1">
  <message>
    <![CDATA[
      Neplatné přihlašovací jméno nebo heslo.
    ]]>
  </message>
</error>
```

```
<error number="2">
  <accessDenied user=""/>
  <accessDenied uri=""/>
  <accessDenied type=""/>
  <accessDenied type="" user=""/>
  <accessDenied type="" uri=""/>
  <message>
    <![CDATA[
      Nemáte oprávnění k prohlížení zvolených anotací.
    ]]>
  </message>
</error>
<error number="3">
  <message>
    <![CDATA[
      Nemáte oprávnění přidávat anotace.
    ]]>
  </message>
</error>
<error number="4">
  <reload uri="http://example.com/annotations/123456"/>
  <message>
    <![CDATA[
      Editace dané anotace není povolena.
    ]]>
  </message>
</error>
<error number="5">
  <reload uri="http://example.com/annotations/123456"/>
  <message>
    <![CDATA[
      Vymazání dané anotace není povoleno.
    ]]>
  </message>
</error>
<error number="6">
  <reload uri="http://example.com/annotations/123456"/>
  <message>
    <![CDATA[
      Anotace tohoto typu musí obsahovat následující atributy: ...
    ]]>
  </message>
  <attribute name="" type=""/>
  <attribute name="" type=""/>
</error>
<error number="7">
  <reload uri="http://example.com/annotations/123456"/>
  <message>
    <![CDATA[
      Hodnota atributu ... musí být v rozsahu ...
    ]]>
  </message>
  <attribute name="" type=""/>
</error>
```

Čísla chyb jsou:

0 Nepodporovaná verze protokolu.
1 Chybné přihlašovací údaje.
2 Nedostatečná oprávnění k požadovaným anotacím.
3 Nelze přidávat anotace - přístup je pouze pro čtení.
4 Editace zvolené anotace není povolena.
5 Mazání anotace není povoleno.
6 Chybí povinné atributy.
7 Nedovolená hodnota atributu.
8 Chybné určení textu pro návrhy anotací.
9 Chyba synchronizace (s daným URI je asociován rozdílný obsah).
10 Nucená synchronizace není povolena.
11 Rozsynchronizování (neshoda anotovaného textu s textem na dané pozici).
12 Editace daného typu není povolena.
13 Mazání daného typu není povoleno.
14 Nelze přidávat typy anotací.
15 Neznámý typ atributu.
16 Chyba v přidávaném typu atributu.
17 Atributy přidávaného typu jsou chybné. Tyto atributy byly vynechány.
18 Chyba v upravovaném typu anotace - změny nelze uložit.
19 Neznámý typ anotace
20 Změna názvu typu či nadtypu není možná.
21 Chyba v nastavení. Změny nelze uložit.
22 Pokus o synchronizaci bez adresy dokumentu.
23 Pokus o synchronizaci bez obsahu dokumentu.
24 Chybná adresa zdroje anotovaného fragmentu.
25 Chybný anotovaný fragment. Anotace bude uložena bez tohoto fragmentu.
26 Chybný atribut anotace. Atribut nelze uložit.
27 Chybná hodnota v atributu rozšířeného typu.
28 Chyba v informacích o způsobu potvrzení nabídnuté anotace.
29 Upravovaná anotace nebyla nalezena. Změny nelze uložit.
30 Mazaná anotace nebyla nalezena. Anotaci nelze vymazat.
31 Vaše přihlášení vypršelo.
32 Chybná zpráva. Pravděpodobně chyba klienta nebo nekompatibilní protokol.
33 Chyba v modulu serveru.
34 Požadovaná anotace nebyla nalezena.
35 Chybná cesta v anotovaném fragmentu. Fragment nebude uložen.
36 Chyba v anotovaném dokumentu.
37 Chybný offset nebo délka v anotovaném fragmentu. Fragment nebude uložen.
38 V upravené anotaci jsou chyby. Změny nelze uložit.
39 Modifikaci textu není možné aplikovat na dokument.
40 Neznámý typ anotací. Nabízení anotací není možné.
41 Chybný formát data v anotaci. Datum bylo upraveno na aktuální.
42 Chybný formát data v atributu. Atribut byl vynechán.
43 Bez synchronizace dokumentu nelze manipulovat s anotacemi.
44 Chyba v informacích o autorovi anotace.
45 Neznámá skupina uživatelů.
46 Neznámá skupina uživatelů v typu anotace. Skupina bude nastavena dle definovaných pravidel.
47 Mazání využitých typů anotací není povoleno. Nejprve je nutno vymazat anotace tohoto typu.
48 Interní chyba serveru způsobila neúspěch při ukládání dat. Žádná data nebyla uložena.
49 Pokus o vytvoření duplicitního typu anotace (shoda URI).
50 Chybný popis modifikace textu.

100 Neznámá chyba.

Pokud dojde k zneplatnění anotace či k jejímu přesunu na globální úroveň, aniž by došlo k chybě, může být potřeba varovat uživatele. V tomto případě server zašle zprávu s varováním. Klient by měl varování zobrazit uživateli a to buď přímo u dotčené anotace, nebo pomocí nějakého postranního panelu či dialogového okna.

Zpráva s varováním má následující syntaxi:

```
<warning number="1" annotation="http://example.com/annotations/123456">
  <![CDATA[
    Anotace byla zneplatněna editací textu.
  ]]>
</warning>
```

Každé varování má číslo (atribut `number`) a textový obsah pro zobrazení uživateli. Pokud se varování týká konkrétní anotace, element `warning` má i atribut `annotation`, který obsahuje URI dané anotace.
Čísla varování jsou:

1 Anotace zneplatněna.
2 Anotace přesunuta na globální úroveň.
3 Anotace automaticky aktualizována.
4 Některé anotované fragmenty v anotaci byly zneplatněny.
100 Jiné varování.

B.10 Potvrzení bez doplňujících dat

Pokud je zaslána zpráva, která vyžaduje provedení nějaké operace, je třeba na ni odpovědět, aby druhá komunikující strana měla potvrzeno přijetí zprávy, případně úspěšné provedení operace, kdy nejsou vrácena data. Pokud není zaslána chybová zpráva, předpokládá se úspěšné provedení operace. Odpověď tedy může obsahovat pouze informace, které jsou důsledkem provedené operace, či jiné užitečné informace. V některých případech však nejsou k dispozici žádné užitečné informace k zaslání a je potřeba, aby server či klient potvrdil úspěšnost operace. V tomto případě pošle následující zprávu:

```
<ok/>
```

C Zjednodušený příklad komunikace mezi klientem a serverem

Klient (zahájení komunikace, uvedení verze protokolu, autentizace):

```
<connect/>
<login/>
```

Server (potvrzení, uvedení verze protokolu, nastavení parametrů):

```
<connected/>
<logged/>
<settings/>
```

Klient (volba zdrojů anotací a anotovaného textu, požadavek na seznam typů):

```
<session/>
<subscribe/>
<synchronize/>
<queryTypes/>
```

Server (adresa zasynchronizovaného zdroje (kopie na serveru), požadované anotace a typy):

```
<synchronized/>
<annotations/>
<types><add/></types>
```

Klient (požadavek na přidání typu anotace):

```
<session/>
<types><add/></types>
```

Server:

```
<ok/>
```

Klient (zašle vloženou anotaci):

```
<session/>
<annotations/>
```

Server (zašle zpět vloženou anotaci s přiděleným identifikátorem):

```
<annotations/>
```

Klient (zašle upravenou anotaci):

```
<session/>
<annotations/>
```

Server (vrátí chybu):

```
<errors/>
```

Klient (požádá o původní obsah anotace):

```
<session/>
<reload/>
```

Server (vrátí požadovanou anotaci):

```
<annotations/>
```

Klient (přidání anotace):

```
<session/>
<annotations/>
```

Server (rozsynchronizování):

```
<errors/>
<resynchronize/>
```

Klient (opakování synchronizace):

```
<session/>
<resynchronize/>
```

Server (všechny anotace pro daný text ze zvolených zdrojů):

```
<annotations/>
```

Klient (požadavek na změnu nastavení):

```
<session/>
<settings/>
```

Server (aktuální nastavení):

```
<settings/>
```

Klient (požadavek na získání nabízených anotací):

```
<session/>
<suggestAnnotations/>
```

Server (nabídka anotací):

```
<suggestions/>
```

Klient (přechod na jiný anotovaný text):

```
<session/>
<synchronize/>
```

Server:

```
<synchronized/>
<annotations/>
```

Klient (odhlášení, ukončení spojení):

```
<session/>
<logout/>
<disconnect/>
```

Server:

```
<ok/>
```

Príloha D. Zdrojový kód jednoduchého modulu pre editor NicEdit

```
1. /**
2. * nicExample
3. * @description: An example button plugin for nicEdit
4. * @requires: nicCore, nicPane, nicAdvancedButton
5. * @author: Brian Kirchoff
6. * @version: 0.9.0
7. */
8.
9. /* START CONFIG */
10. var nicExampleOptions = {
11. buttons : {
12.     'example' : {name : __('Some alt text for the button'),
            type : 'nicEditorExampleButton'}
13.   }/* NICEDIT_REMOVE_START */,iconFiles : {'example' :
      'src/nicExample/icons/save.gif'}/* NICEDIT_REMOVE_END */
14. };
15. /* END CONFIG */
16.
17. var nicEditorExampleButton = nicEditorButton.extend({
18.   mouseClick : function() {
19.     alert('The example save button icon has been clicked!');
20.   }
21. });
22.
23. nicEditors.registerPlugin(nicPlugin,nicExampleOptions);
```

Zdroj: [15].

Príloha E. Zdrojový kód jednoduchého modulu pre editor TinyMCE

```
 1. (function() {
 2.
 3.  tinymce.PluginManager.requireLangPack('example');
 4.
 5.  tinymce.create('tinymce.plugins.ExamplePlugin', {
 6.
 7.   init : function(ed, url) {
 8.     ed.addCommand('mceExample', function() {
 9.       ed.windowManager.open({
10.          file : url + '/dialog.htm',
11.          width : 320 + ed.getLang('example.delta_width', 0),
12.          height : 120 + ed.getLang('example.delta_height', 0),
13.          inline : 1
14.        }, {
15.          plugin_url : url, // Plugin absolute URL
16.          some_custom_arg : 'custom arg' // Custom argument
17.        });
18.      });
19.
20.      ed.addButton('example', {
21.         title : 'example.desc',
22.         cmd : 'mceExample',
23.         image : url + '/img/example.gif'
24.      });
25.
26.      ed.onNodeChange.add(function(ed, cm, n) {
27.         cm.setActive('example', n.nodeName == 'IMG');
28.      });
29.    },
30.    getInfo : function() {
31.      return {
32.        longname : 'Example plugin',
33.        author : 'Some author',
34.        authorurl : 'http://tinymce.moxiecode.com',
35.        infourl : 'http://example.com/example ',
36.        version : "1.0"
37.      };
38.    }
39.  });
40. tinymce.PluginManager.add('example',
tinymce.plugins.ExamplePlugin);
41. })();
```

Zdroj: [37].

Príloha F. Obsah priloženého DVD

Priložené DVD obsahuje:

- Zdrojový súbor a súbor PDF textovej časti tejto diplomovej práce.
- Súbor PDF príloh A, B a C, ktorý je prevzatý z [47].
- Zdrojový súbor a súbor PDF príloh D, E, F a G.
- Zdrojové súbory editora anotácií.
- Programovú dokumentáciu.
- Inštalačný a užívateľský manuál.
- Aktuálnu verziu editora TinyMCE.
- Aktuálnu verziu servera pre kolaboratívne anotovanie - 4A Framework.
- Textový súbor readme.txt, ktorý bližšie popisuje štruktúru priloženého DVD.

Príloha G. Inštalačný manuál

Sysémové požiadavky:

- Webový prehliadač so zapnutou podporou jazyka JavaScript.
- Editor TinyMCE, pre ktorý chceme editor anotácií nainštalovať ako rozšírenie.

Postup inštalácie:

- Nakopírujte súbory editora anotácií z priloženého DVD do požadovaného umiestnenia.
- Nakopírujte plugin pre TinyMCE s názvom *aed* do zložky *plugins* v adresárovej štruktúre TinyMCE.
- V konfigurácii editora TinyMCE je potrebné povoliť plugin *aed* a taktiež do panela nástrojov pridať tlačítko s rovnakým názvom *aed*.
- V hlavičke dokumentu, ktorý obsahuje editor TinyMCE, je potrebné pridať odkazy na všetky zdrojové súbory editora anotácií tak, ako je to v súbore *aed-tinymce.html*, ktorý je na priloženom DVD. Odkazy na zdrojové súbory musia smerovať do miesta, kde bol editor anotácií nakopírovaný.
- Pred inicializáciou editora TinyMCE je potrebné inicializovať editor anotácií, príklad je uvedený v súbore *aed-tinymce.html*.