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

ANOTACE VE WYSIWYG TEXTOVÝCH EDITORECH

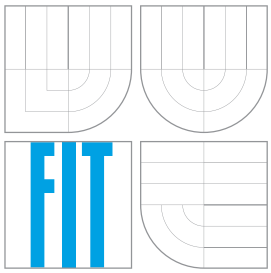
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETR LOUKOTA

BRNO 2012

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

ANOTACE VE WYSIWYG TEXTOVÝCH EDITORECH

ANNOTATIONS IN WYSIWYG TEXT EDITORS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETR LOUKOTA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAROSLAV DYTRYCH

BRNO 2012

Zadání bakalářské práce/13792/2011/xlouko00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2011/2012

Zadání bakalářské práce

Řešitel: **Loukota Petr**
Obor: Informační technologie
Téma: **Anotace ve WYSIWYG textových editorech**
Annotations in WYSIWYG Text Editors
Kategorie: Web

Pokyny:

1. Seznamte se s různými WYSIWYG textovými editory vytvořenými v jazyce JavaScript (např. TinyMCE, CKEditor, Aloha apod.).
2. Prostudujte 4A framework a dostupný editor anotací.
3. Navrhněte opravy a vylepšení editoru anotací tak, aby splňoval vybrané požadavky uživatelů, přičemž se zaměřte na manipulaci s atributy typů anotací a na zprovoznění nabízení anotací.
4. Implementujte navržené řešení.
5. Zhodnoťte dosažené výsledky a srovnejte s alternativními přístupy.

Literatura:

- dle doporučení vedoucího

Při obhajobě semestrální části projektu je požadováno:

- Body 1, 2 a 3.

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

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% 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: 1. listopadu 2011
Datum odevzdání: 16. května 2012

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 bakalářská práce se zabývá návrhem oprav a vylepšení dostupného editoru anotací a jejich implementací. Editor anotací pro WYSIWYG textové editory je napsán v jazyce JavaScript a vyvíjen v prostředí editoru TinyMCE. Je nezávislý na webovém prohlížeči a v budoucnu jej bude možné použít s dalšími WYSIWYG editory jako CKEditor či Aloha. Návrh je zaměřen na manipulaci s atributy typů, zabývá se však i novým konceptem nastavení editoru anotací. Součástí řešení je také zpracování nabídek anotací zasílaných serverem. Tato práce popisuje návrh a implementaci zmíněných vylepšení a ve svém závěru se věnuje možnostem dalšího vývoje.

Abstract

This bachelor's thesis deals with the design of fixes and improvements of available annotations editor and their implementation. The annotations editor for WYSIWYG text editors is created in JavaScript language and developed in the environment of TinyMCE editor. It is independent on a web browser and it will be possible to use it with other WYSIWYG editors like CKEditor or Aloha in future. The design is focused on manipulation with attributes of types but it also deals with the new concept of settings of annotations editor. The processing of suggested annotations that are sent by a server is also part of the solution. This thesis describes the design and implementation of mentioned improvements and pay attention to the possible ways of another development at its end.

Klíčová slova

JavaScript, WYSIWYG, textový, editor, TinyMCE, editor anotací, anotace, web, rozšíření, vylepšení

Keywords

JavaScript, WYSIWYG, text, editor, TinyMCE, annotations editor, annotation, web, extension, improvement

Citace

Petr Loukota: Anotace ve WYSIWYG textových editorech, bakalářská práce, Brno, FIT VUT v Brně, 2012

Anotace ve WYSIWYG textových editorech

Prohlášení

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

.....
Petr Loukota
8. května 2012

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce Ing. Jaroslavu Dytrychovi za věcné připomínky k návrhům vylepšení a rychlou odbornou pomoc při řešení problémů.

Obsah

Kapitola 1

Úvod

Cílem této bakalářské práce je navrhnout opravy a vylepšení dostupného editoru anotací. Návrh je zaměřen především na manipulaci s atributy typů, zabývá se však i novým konceptem nastavení editoru anotací. Součástí řešení je také zpracování nabídek anotací, které jsou zasílány serverem. Tato práce popisuje nejen samotný návrh, ale také implementaci a testování navržených oprav a vylepšení. Závěrem se pak věnuje možnostem, kterými by se editor anotací mohl vyvíjet do budoucna.

Kapitola 2 se zabývá analýzou dostupného editoru anotací s důrazem na jeho nedostatky, možné opravy a vylepšení. Její úvodní část je zaměřena na popis WYSIWYG editorů a zejména na editor TinyMCE, v jehož prostředí je editor anotací vyvíjen. Definuje také 4A Framework coby návrh systému pro kolaborativní poloautomatické strukturování znalostí. V další části jsou popsány nedostatky editoru anotací týkající se manipulace s atributy typů, nastavení a nabízení anotací.

Popis technologií, které byly využity pro dosavadní vývoj editoru anotací, a tak jsou využity také pro jeho opravy a vylepšení, je k dispozici v kapitole 3. Tato kapitola se věnuje technologiím HTTP, HTML, XHTML, XML, XPath, DOM, CSS, JavaScript, AJAX a Comet.

V kapitole 4 je popsán návrh oprav a vylepšení dostupného editoru anotací. První část kapitoly se zabývá manipulací s atributy typů a navrhuje řešení všech problémů, které s touto problematikou souvisejí, další část je věnována návrhu nastavení editoru anotací. V poslední části kapitoly je popsán návrh nabízení anotací, které by editor anotací měl umožňovat.

Kapitola 5 je věnována implementaci nejdůležitějších oprav a vylepšení editoru anotací, blíže se pak zabývá závažnějšími problémy, které se při implementaci vyskytly a které bylo třeba řešit. Jedná se například o validaci data a času v jazyce JavaScript nebo práci s atributy typu *anyAnnotation*.

Kapitola 6 se zabývá možnostmi testování navržených oprav a vylepšení editoru anotaci a nástroji, které k testování byly využity. Dále popisuje několik vybraných nedostatků, které byly při testování odhaleny, spolu s jejich řešeními.

Závěr celé práce přináší souhrn poznatků získaných studiem technologií a dostupného editoru anotací, shrnuje také výsledky, kterých bylo implementací navržených oprav a vylepšení dosaženo, a možnosti dalšího vývoje.

Kapitola 2

Analýza dostupného editoru anotací

Tato kapitola se věnuje analýze dostupného editoru anotací, jehož vývojem se zabýval Ing. Martin Kleban ve své diplomové práci [11]. Důležitou součástí této bakalářské práce je zhodnocení současného stavu editoru anotací, které pomůže odhalit nedostatky a rovněž předpokládané funkce, které však v implementaci prozatím zcela chybí. Díky této analýze pak bude možné navrhnout opravy a vylepšení tak, aby takto vylepšený editor anotací splňoval vybrané požadavky uživatelů.

Úvodní část této kapitoly je věnována WYSIWYG editorům, a to s důrazem na TinyMCE editor, se kterým v současné době editor anotací pracuje a který je tak využit jako vývojové prostředí editoru anotací. Následně je popsáno propojení, díky němuž editor anotací komunikuje s editorem TinyMCE, a také 4A framework, který obsahuje mimo jiné i návrh vizuální podoby klientů, tedy i editoru anotací pro WYSIWYG textové editory. Zbytek kapitoly je pak věnován samotnému editoru anotací a jeho současnému stavu.

2.1 WYSIWYG textové editory

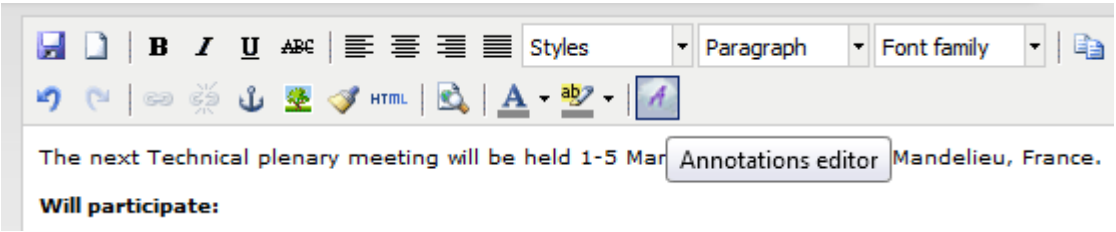
WYSIWYG (What You See Is What You Get) textovými editory rozumíme editory, které uživateli psaný text zobrazují v takové podobě, jakou bude skutečně mít výsledný dokument. WYSIWYG je způsob editace dokumentu, díky němuž je úprava daného dokumentu rychlá a přehledná, a dnes jsou tímto pojmem nejčastěji označovány textové procesory. Pojmem *WYSIWYG editory* obvykle chápeme editory pro tvorbu celých webových stránek takovým způsobem, že HTML kód je generován automaticky. Teprve WYSIWYG textové editory umožňují tvorbu pouze náplně webových stránek. V praxi mají pojmy *WYSIWYG editor* a *WYSIWYG textový editor* často stejný význam plynoucí z kontextu, v této práci jsou oba pojmy použity pro WYSIWYG textové editory napsané v jazyce JavaScript. [26]

WYSIWYG textové editory jsou tedy využívány především při tvorbě obsahu webových stránek, např. článků. Tyto editory rovněž tvoří HTML kód, který je uživatelům skrytý, a text zobrazují tak, jak bude na stránce skutečně vypadat. WYSIWYG textové editory jsou dnes součástí většiny redakčních systémů (CMS – Content Management Systems), přičemž existuje několik volně dostupných editorů, které může zdarma využít kterýkoliv CMS. Mezi ty nejpopulárnější dnes patří například TinyMCE [14], CKEditor [1] či Aloha [5].

2.1.1 TinyMCE a jeho rozšíření

TinyMCE je v současné době jeden z nejpopulárnějších WYSIWYG textových editorů. Je napsaný v jazyce JavaScript a dnes jej využívá řada úspěšných redakčních systémů. Podle domovských stránek [14] je TinyMCE aktuálně ve verzi 3.5b3.

S ohledem na popularitu TinyMCE byl právě tento editor zvolen jako WYSIWYG editor, v jehož prostředí je vyvíjen editor anotací. Tato volba vyžadovala nejprve vytvoření jednoduchého rozšíření editoru TinyMCE, které umožní především spouštět editor anotací z jeho nástrojového panelu. Toto rozšíření bude v budoucnu třeba vytvořit pro každý další zvolený WYSIWYG editor. Zmíněné rozšíření editoru TinyMCE je zobrazeno na obrázku 2.1.



Obrázek 2.1: Rozšíření editoru TinyMCE, které umožňuje spouštět editor anotací

2.2 4A Framework

4A Framework [20] vznikl jako součást disertační práce na Fakultě informačních technologií VUT v Brně. Koncept samotné disertační práce je popsán v pojednání k tématu disertační práce [2], přičemž 4A Frameworku se blíže věnuje také samostatná prezentace [3]. Součástí disertační práce je vytvoření systému, který by umožňoval kolaborativní poloautomatické strukturování znalostí v reálném čase, a 4A Framework tento systém navrhuje. 4A Framework zahrnuje nejen návrh jednotlivých klientů, ale také specifikaci protokolu pro komunikaci se serverem a dále popis formátu anotací.

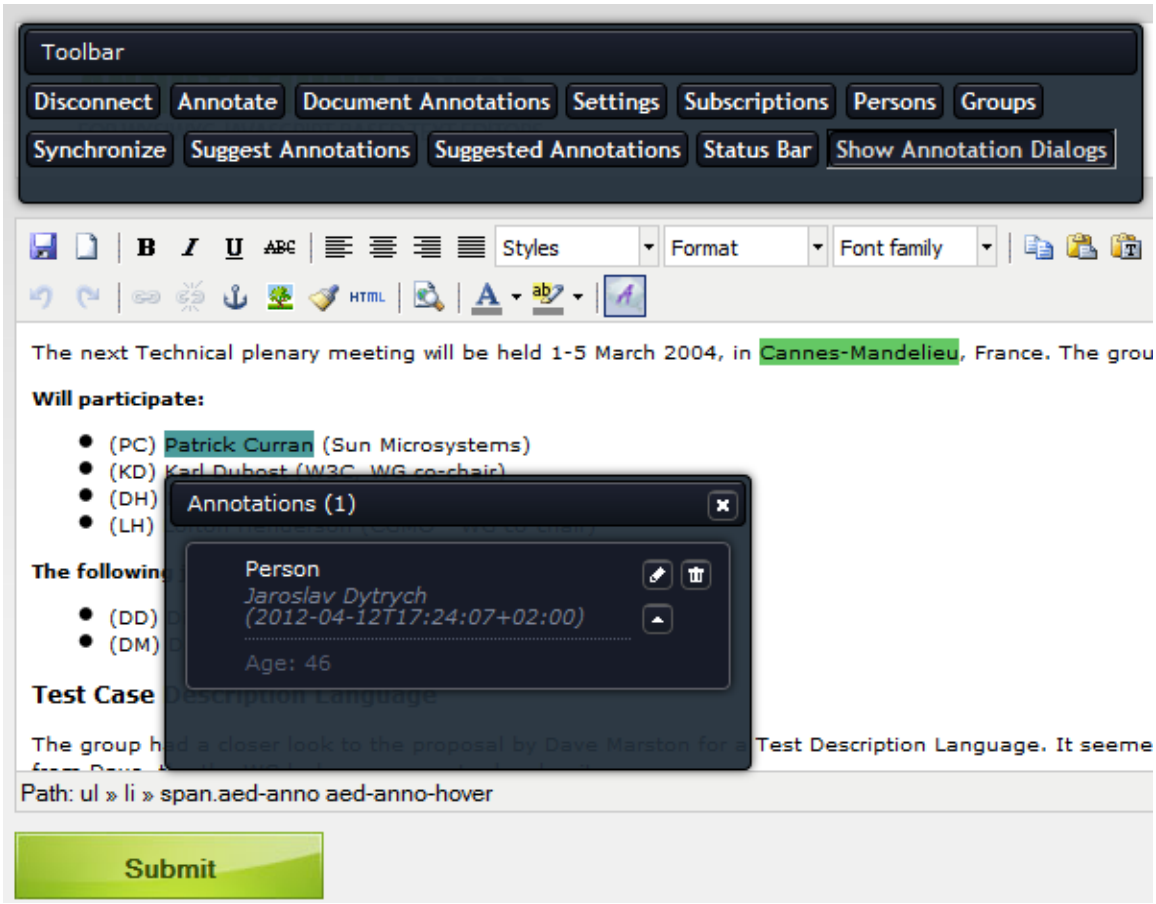
Klientem je chápán editor anotací, který uživateli umožňuje anotovat dokument nebo jeho části a zasílat vytvořené anotace serveru. Server pak provádí nejen správu anotací, ale také správu typů anotací a jejich atributů, uživatelů, skupin i nastavení. Tato bakalářská práce se tedy zabývá vylepšením dostupného editoru anotací, přičemž pojmem *editor anotací* je v této práci vždy myšlen editor anotací pro WYSIWYG textové editory. Nezávisle na tomto klientovi je vyvíjen i editor anotací jako rozšíření pro webový prohlížeč Firefox a další.

2.3 Editor anotací

Editor anotací pro WYSIWYG textové editory umožňuje anotování právě vytvářeného dokumentu. V dostupné verzi je hotová základní konstrukce editoru, editor lze tedy spustit, připojit se k serveru a rovněž s ním komunikovat. Editor anotací disponuje hlavními funkcemi, je možné vytvořit, editovat či smazat anotaci. Zatímco anotace vytvořené pro určitou část dokumentu se zobrazují přímo v textu zvýrazněním daného fragmentu, anotace celého dokumentu lze nalézt v samostatném dialogu. Při anotování se doposud nepracuje s atributy typů, které uživatelům značně usnadní vytváření a editaci anotací. Atributům typů se více věnuje kapitola 4.

Dostupný editor anotací dále obsahuje jednoduché nastavení, které však není pro běžné používání zcela ideální. K dispozici je také základní správa uživatelů a skupin. Uživatele je možné filtrovat dle jména a aktuálně přihlášený uživatel se může přidat či odebrat ze zvolené skupiny. Anotovaný dokument lze manuálně synchronizovat se serverem. V takovém případě editor odstraní všechny zobrazené anotace a vytvoří je znovu podle XML zaslaného serverem. Editor anotací je připraven pro lokalizaci, nyní však existuje pouze jeho verze v anglickém jazyce. Nabízení anotací, které je více popsáno v kapitole 4, dostupný editor anotací neumí.

Na obrázku 2.2 je zobrazeno uživatelské rozhraní, které bylo navrženo tak, aby editor anotací mohl poskytovat všechny požadované funkce. Grafická podoba dialogů, které prozatím neposkytují funkčnost dle specifikace, není dokončena.



Obrázek 2.2: Dostupný editor anotací

2.3.1 Rozhraní pro spolupráci s WYSIWYG editory

Přestože současný editor anotací pracuje pouze s editorem TinyMCE, jeho návrh byl vytvořen tak, aby jej v budoucnu bylo možné používat i s kterýmkoliv jiným WYSIWYG editorem. Aby editor anotací nebyl přímo závislý na zvoleném WYSIWYG editoru, bylo nezbytné mimo rozšíření pro daný editor vytvořit také rozhraní pro spolupráci editoru anotací s WYSIWYG editory. Toto rozhraní umožňuje jednotný způsob práce se všemi WYSIWYG editory, pro každý další zvolený WYSIWYG editor tedy postačí vytvořit rozšíření implementující toto rozhraní.

2.3.2 Manipulace s atributy typů

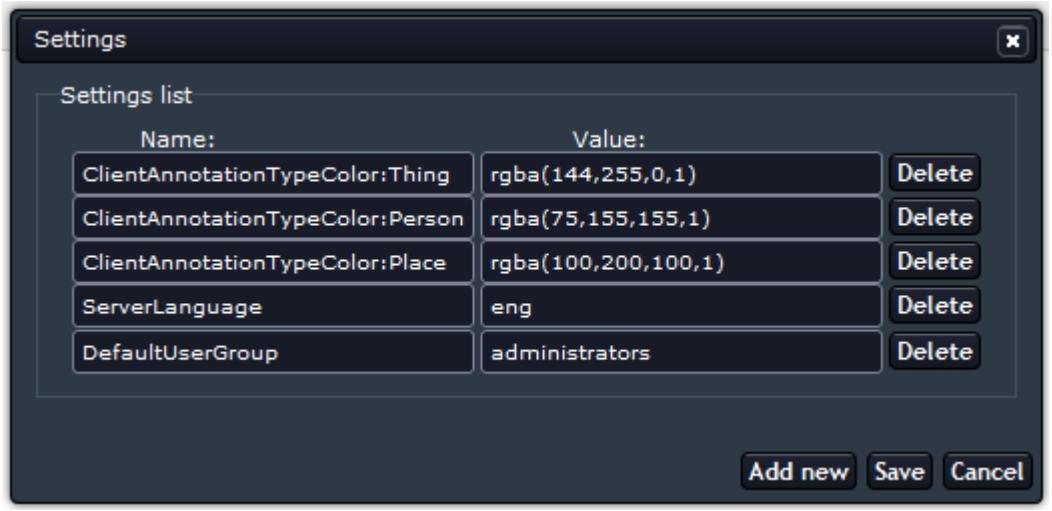
Manipulace s atributy typů usnadní uživateli vytváření a editaci anotací. S výběrem typu anotace se mají načíst jeho atributy, kterým už uživatel pouze přiřadí hodnotu, nemusí tedy každý atribut přidávat manuálně. Dostupný editor anotací žádným způsobem nepracuje s atributy typů.

Atributy typů mohou být jednoduché nebo strukturované. Editor anotací nabízí sedm jednoduchých typů atributů (*Integer*, *String*, *Date* a další), přičemž každý atribut může obsahovat pouze povolené hodnoty. Strukturovaný atribut naopak obsahuje vnořenou anotaci nebo odkaz na již existující anotaci, přičemž tento může mít i další atributy – jednoduché či strukturované. Dostupný editor anotací neřeší manipulaci s atributy typů, proto ani neošetřuje povolené hodnoty, nezabývá se změnou strukturovaného atributu na jednoduchý, kdy je třeba odstranit všechny vnořené atributy strukturovaného atributu, neřeší ani načtení atributů odkazované anotace, které však nesmějí být editovány, a další. Uživatel by také měl mít možnost přidávat atributy k daným typům nebo je mazat.

Manipulace s atributy typů je rozsáhlý problém, který se svou implementací bude vyžadovat zásah do více částí editoru. Již hotovou funkcionalitu bude třeba opravit a vylepšit. Editor anotací je navržen tak, aby manipulaci s atributy typů bylo možné doplnit.

2.3.3 Nastavení

Dostupný editor anotací umožňuje uživateli zadat známé nastavení pomocí jeho jména a hodnoty, jak je patrné z obrázku 2.3. Pokud uživatel nezná přesný název nastavení, nemůže takové nastavení použít. Editor anotací by ale měl v jednom dialogu uživateli umožnit zadat pouze hodnoty všech známých nastavení, a to tak, aby každé nastavení bylo co nejlépe přizpůsobené. Nastavení hlavní uživatelské skupiny by mělo jít vybrat ze skupin, do kterých je uživatel přihlášen, nastavení barev typů anotaci by mělo být možné zadat výběrem zvolené barvy a tak dále. Uživateli současně musí být zachována možnost zadat nové, pro editor anotací neznámé nastavení.



Obrázek 2.3: Nastavení dostupného editoru anotací

2.3.4 Nabízení anotací

Editor anotací má umět zobrazit nabídky anotací, které mu zašle server, podobně, jako jsou zobrazovány anotace. Uživatel nejprve musí mít možnost určit, zda chce nabídky obdržet pro celý dokument nebo jeho část a anotace jakého typu mu mají být nabídnuty. Poté, co server zašle nabídky anotací, editor anotací musí nabídky zobrazit v textu (nabídky anotací celého dokumentu současný návrh nezahrnuje) a uživateli dovolit nabídnuté anotace přijmout, odmítnout či přijmout po editaci. Po přijetí nabídky má dojít k vytvoření běžné anotace, po odmítnutí k odstranění nabídky. Pokud se navíc jedná o nabídku vnořenou v jiné nabídce, nesmí být možné takovou nabídku samostatně přijmout ani odmítnout. Dostupný editor anotací nemá implementovánu žádnou část nutnou pro správné fungování nabízení anotací.

Kapitola 3

Použité technologie

Následující kapitola shrnuje technologie, které byly využity pro vývoj editoru anotací. Tyto technologie byly voleny s ohledem na požadavky výsledného řešení a jedná se zejména o technologie běžně používané pro tvorbu webových stránek a webových aplikací. Tato bakalářská práce se věnuje dalšímu rozšíření a vylepšení dostupného editoru anotací, proto také využívá již dříve zvolené technologie.

U každé z technologií se zaměřím na její stručný popis a vznik, zmíním se také o aktuální verzi dané technologie. Tato kapitola popisuje protokol HTTP, jazyk HTML, CSS, JavaScript, XML a XPath, zabývá se však i zajištěním asynchronní komunikace pomocí AJAX a Comet.

3.1 HTTP

HTTP (Hypertext Transfer Protocol) je v současné době nejpoužívanějším protokolem pro přenos dat mezi klientem a serverem. Činnost tohoto protokolu je založena na principu dotaz-odpověď, přičemž je určen pro výměnu hypertextových dokumentů ve formátu HTML. Díky standardu MIME (Multipurpose Internet Mail Extensions) je však možné tímto protokolem přenášet i libovolné jiné soubory. [6]

Klient, který po serveru požaduje nějaký dokument, vytvoří požadavek, který mimo jiné obsahuje identifikátor požadovaného dokumentu. Server pak na takovou žádost odpoví a za odpovědí zašle požadovaný dokument. Při opakovaném požadavku klienta server nepozná, zda nový požadavek souvisí s předchozím, proto protokol HTTP nazýváme bezstavovým protokolem. [6]

První verze protokolu HTTP vznikla v roce 1991 a nesla označení HTTP/0.9. Server tehdy posílal přímo požadovaný dokument, jednalo se tedy o vůbec nejjednodušší implementaci. Verze HTTP/1.0, která mimo jiné využívá internetový standard MIME pro identifikaci typu dokumentu, pochází z roku 1996. Aktuální verze tohoto protokolu HTTP/1.1, která je stabilní, a tak se již dále nevyvíjí [34], však vznikla ještě o rok později, přepsána pak byla v roce 1999 a její specifikace je dána normou RFC 2616 [12].

3.1.1 Identifikátor zdroje

Jednotlivá místa v prostředí Internetu jsou označována zdroje. Každý takový zdroj poskytuje nějaká data a každý zdroj má také svůj jednoznačný identifikátor. V současné době existují dva typy identifikátorů – URL (Uniform Resource Locator) a URN (Uniform Resource Name). Oba typy identifikátorů se souhrnně označují URI (Uniform Resource Identifier),

přičemž protokol HTTP pro identifikaci zdrojů využívá URL. URL specifikuje místo v síti, jde tedy o identifikátor závislý na umístění v síti. Oproti tomu URN pojmenovává zdroj, a tak je na umístění v síti nezávislý. [6]

3.1.2 Hypertextový dokument

Hypertext je takový text, který ve svém obsahu zahrnuje odkazy. Hypertextovým dokumentem pak tedy rozumíme dokument, ve kterém převládá běžný text, avšak doplněný o odkazy na další hypertextové dokumenty. [27]

3.2 HTML a XHTML

HTML (Hypertext Markup Language) je značkovací jazyk, který je základním jazykem využívaným při tvorbě webových stránek. Jedná se o nejznámější aplikaci univerzálního značkovacího jazyka SGML (Standard Generalized Markup Language). Pro HTML je charakteristická množina značek a atributů, které udávají význam jednotlivých částí dokumentu. Dokument ve formátu HTML je hypertextový dokument. Každá značka v jazyce HTML může být párová či nepárová. Nepárová značka má specifický význam v daném místě dokumentu, párová značka pak strukturuje obsah dokumentu nacházející se mezi počáteční a koncovou značkou dle významu značky a jejích atributů. [29]

Vývoj jazyka HTML začal v roce 1991, kdy byla vydána jeho první verze s označením HTML 0.9. Do roku 1999 pak vzniklo několik dalších verzí včetně HTML 4.0, do jejíž specifikace přibyly nové značky. Chyby této verze byly opraveny v následném HTML 4.01, které je dodnes také nejčastěji používanou verzí jazyka HTML. Od roku 2007 se dále připravuje HTML 5, které přidává další nové značky například pro podporu přehrávání videí. Dokončení specifikace je však i dnes otázkou ještě několika let. Ani podpora HTML 5 ze strany webových prohlížečů není úplná, proto je jeho použití při vývoji webových stránek mnohdy problematické. [28]

Pod pojmem XHTML se rozumí varianta jazyka HTML, která je aplikací jazyka XML (Extensible Markup Language) [29]. Původně se předpokládalo, že jazyk XHTML nahradí HTML, nakonec se tak ale nestalo. S vývojem HTML 5 se pracuje také na variantě XHTML 5. Aktuální verze XHTML označena XHTML 1.1 pochází z roku 2010 [28].

3.3 XML

XML (Extensible Markup Language) je jazyk vyvinutý a standardizovaný konsorciem W3C, který slouží jako jednoduchý textový formát na reprezentaci strukturovaných dat. Jak již ze samotného názvu vyplývá, XML je snadno rozšiřitelný, byl odvozen od jazyka SGML jeho zjednodušením. Jazyk XML je určen především pro výměnu dat mezi aplikacemi.

Od vzniku jazyka XML se vyvinulo několik souvisejících technologií. Mezi ně patří například XLS [32] sloužící pro transformaci XML dokumentu na jiný XML, HTML či textový dokument, dále jazyk XQuery [31], který umožňuje pokládání dotazů nad daty ve formátu XML, nebo technologie XPath, kterou se budeme zabývat v samostatné kapitole. Poslední verze jazyka XML je pátá revize verze 1.0 pocházející z roku 2008. [35]

3.4 XPath

XPath (XML Path Language) je jazyk, který umožňuje zpracovat hodnoty odpovídající datovému modelu XDM (XQuery and XPath Data Model). Datový model reprezentuje XML dokument v podobě stromu a zároveň vymezuje povolené hodnoty výrazů v jazyce XPath. Výsledkem výrazu v jazyce XPath může být výběr uzlů ze vstupního dokumentu, atomická hodnota nebo libovolná sekvence povolená datovým modelem.

XPath 1.0 je podmnožinou XPath 2.0, přičemž XPath 2.0 rozšiřuje předchozí verzi například o širší podporu datových typů. XPath 2.0 je zpětně kompatibilní s XPath 1.0 a téměř všechny výrazy v XPath 1.0 vracejí stejné výsledky i v XPath 2.0. [30]

3.5 DOM

DOM (Document Object Model) neboli objektový model dokumentu je objektově orientovaná reprezentace dokumentů ve formátech HTML a XML. Pomocí DOM je možné přistupovat k danému dokumentu jako ke stromu a modifikovat tak jeho obsah. Jedná se o platformně a jazykově nezávislé rozhraní, které bylo jako reakce na nekompatibilitu původních rozhraní v různých webových prohlížečích standardizováno konsorciem W3C. V současné době tak lze vytvářet skripty, které správně fungují ve většině webových prohlížečů.

Přestože zmíněný standard W3C definuje vazbu DOM na jazyky JavaScript a Java [41], není DOM na těchto jazycích přímo závislý a jeho implementace existuje i pro řadu dalších programovacích jazyků. První verze standardu W3C vznikla v roce 1998 a od té doby byl standard dále upravován. V současné době se dle verzí dělí do několika úrovní, přičemž existují úrovně DOM Level 0 až DOM Level 3. Každá aplikace, která podporuje určitou úroveň (DOM Level), pak musí implementovat všechny povinné moduly této úrovně i všech nižších úrovní. [8]

3.6 CSS

CSS (Cascading Style Sheets) je jazyk, který slouží pro popis způsobu zobrazení webových stránek. Pomocí CSS je možné definovat rozložení stránky, barvy a písmo a přizpůsobit tak stránku různým typům zobrazení. Jazyk CSS je nezávislý na jazyce HTML a lze jej použít i s jakýmkoliv jiným jazykem založeným například na jazyce XML. Hlavní výhodou CSS je oddělení obsahu dokumentu od jeho vizuální podoby, která je tak definována v samostatném CSS souboru. Jediný styl pak lze navíc využít pro více nezávislých dokumentů. [29]

Syntaxe jazyka CSS sestává z pravidel, z nichž každé obsahuje selektor a dále výčet vlastností, které se k němu vztahují. Selektor určuje, kterým elementům v dokumentu budou přiřazeny dané vlastnosti, přičemž selektory je možné různě kombinovat. Deklarace vlastnosti je pak vždy složena z názvu a hodnoty vlastnosti, které jsou od sebe navzájem odděleny dvojtečkou.

První návrh jazyka CSS vznikl v roce 1994 a jeho autorem byl Håkon Wium Lie [13]. V současné době je CSS udržováno konsorciem W3C. Aktuální verze CSS pochází z června roku 2011 [33], kdy byla vydána první revize CSS verze 2 obsahující rozšíření, která již dnes podporuje většina webových prohlížečů. Vedle CSS2.1 se dále pracuje také na CSS3, jehož některé vlastnosti již nyní lze při tvorbě webových stránek využít.

3.7 JavaScript

JavaScript je interpretovaný, objektově-orientovaný programovací jazyk, který je svou syntaxí podobný jazyku C, C++ či Java. Zde však podobnost končí, JavaScript má potlačenu typovou kontrolu a mnohé další myšlenky přebírá z jazyka Perl – příkladem může být práce s poli či regulárními výrazy. JavaScript se dnes využívá při tvorbě webových stránek, vkládá se přímo do HTML dokumentu a lze pomocí něj vytvořit animace, efekty, ale též jednoduché GUI s tlačítky a textovými poli. Klientský JavaScript je dnes podporován většinou webových prohlížečů. [7]

JavaScript se nejprve vyvíjel pod označením LiveScript, v produktech Microsoft je také označován jako JScript. Na počátku vzniku jazyka JavaScript stála společnost Netscape a později také Sun Microsystems. Organizace ECMA vytvořila a publikovala několik verzí standardu jazyka JavaScript, v roce 1997 pak vznikl standard ECMA-262, který JavaScript definoval, ovšem pod označením ECMAScript [40]. Pod pojmem JavaScript dnes tedy rozumíme jeden jazyk, byť ne se zcela ustáleným názvem, právě název JavaScript se však nejvíce prosadil. Nejnovější verzí standardu ECMA-262 je edice 5.1 vydaná v červnu roku 2011 [16].

Pojem *klientský JavaScript* vznikl integrací JavaScriptu do webových prohlížečů. Kromě univerzálních prostředků jazyka využívá klientský JavaScript také DOM, který umožňuje pracovat s obsahem dokumentů, a dále rozhraní pro interakci s událostmi webového prohlížeče. Klientský JavaScript je také možné obohatit o další knihovny, JavaScript lze však použít i na straně serveru. [7]

Zajímavostí jazyka JavaScript je způsob implementace tříd, které jsou typické pro objektově-orientované jazyky. JavaScript nemá třídy, jak je známe z jiných jazyků, za třídu však považujeme samotný konstruktor, který využívá vlastnost *prototype*. [25]

Při vývoji editoru anotací bude často potřeba manipulovat s textovými řetězci, k čemuž v jazyce JavaScript slouží objekt *String* [37]. Tento objekt mimo jiné nabízí metodu *substr()*, kterou lze využít pro zjištění, čím určitý textový řetězec končí [21]. Důležitá je také manipulace s číselnými hodnotami, které lze rozpoznat například pomocí funkce *isFinite()* [36].

3.8 AJAX

AJAX (Asynchronous JavaScript and XML) je technologie, která umožňuje měnit obsah webových stránek, aniž by stránka musela být opětovně načtena. Tato technologie se využívá při vývoji webových aplikací a dnes ji podporuje většina webových prohlížečů. Pod pojmem AJAX chápeme několik technologií, které se využívají současně, jde zejména o HTML (XHTML), CSS, DOM, JavaScript a objekt XMLHttpRequest - slouží pro asynchronní komunikaci se serverem.

Za autora pojmu AJAX je považován Jesse James Garrett, který jej v roce 2005 představil ve svém článku nazvaném *Ajax: A New Approach to Web Applications* [4]. AJAX dnes používají například některé aplikace společnosti Google, které se též zasloužily o jeho popularitu. Mezi nevýhody technologie AJAX patří vyšší počet HTTP požadavků, a tedy větší objem přenášených dat. [9]

3.9 Comet

Comet je koncept, který se využívá při tvorbě webových aplikací a který umožňuje zasílání dat ze serveru klientům v libovolném okamžiku. Klient nejprve provede inicializaci komunikace, odpověď serveru pak přijde jako reakce na událost serveru, tedy bez další akce klienta. Jednou z možností přenosu dat je vytvoření jednoho udržovaného spojení, což výrazně snižuje dobu odezvy. Comet je podobný technologii AJAX, je tedy založený na asynchronní komunikaci mezi klientem a serverem. Koncept Comet je vhodný pro víceuživatelské webové aplikace, které vyžadují komunikaci v reálném čase. Autorem pojmu Comet je Alex Russell, který jej v roce 2006 představil v jednom z článků na svém blogu [19].

Kapitola 4

Návrh oprav a vylepšení

V této kapitole se zabývám návrhem oprav a vylepšení dostupného editoru anotací, které vycházejí z analýzy jeho současného stavu popsané v kapitole 2. Vylepšení editoru jsem navrhoval dle jeho požadovaných vlastností a nejdůležitějších nedostatků, které jsem objevil v dostupném editoru anotací.

První část této kapitoly je zaměřena na návrh manipulace s atributy typů. Bez ní je anotování dokumentu zdlouhavé a nenabízí možnosti dle specifikace, nebot' uživatel musí každý atribut manuálně vytvořit. V kapitole se dále věnuji návrhu nového nastavení editoru anotací, které má za cíl skrýt před uživatelem implementační detaily a poskytnout přizpůsobené nastavení. Poslední část je věnována nabízení anotací, kde se kromě možnostem implementace věnuji návrhu dialogu pro odeslání žádostí o nabízení serveru, dialogu pro zobrazení nabídnutých anotací a také zobrazení nabídnutých anotací v textu dokumentu.

4.1 Manipulace s atributy typů

Manipulace s atributy typů uživateli usnadní vytváření nových anotací. Atributy typů jsou uloženy u každého typu a při vytváření anotace daného typu by s nimi mělo být možné manipulovat. Manipulace zahrnuje nejen načtení atributů, ale také změny jejich typů, přidávání či odstraňování atributů z daného typu a v některých případech i přesun atributů mezi úrovněmi stromu atributů. Uživatel bude mít možnost ke každému typu přidat atributy a při dalším vytváření anotace tohoto typu už pouze vyplnit hodnoty načtených atributů.

Každý atribut je nějakého typu. Tento typ může být jednoduchý, v takovém případě je typ atributu některý ze sedmi jednoduchých typů, které editor anotací podporuje, nebo strukturovaný. Kterýkoliv uživatelem vytvořený typ je strukturovaný typ. Atributy jednoduchých typů jsou nazývány jednoduché atributy, atributy strukturovaných typů naopak strukturované atributy. Atributy strukturovaného typu mohou obsahovat vnořenou anotaci nebo odkaz na již existující anotaci a mohou rovněž mít další atributy. Strom atributů tedy může mít libovolný počet úrovní.

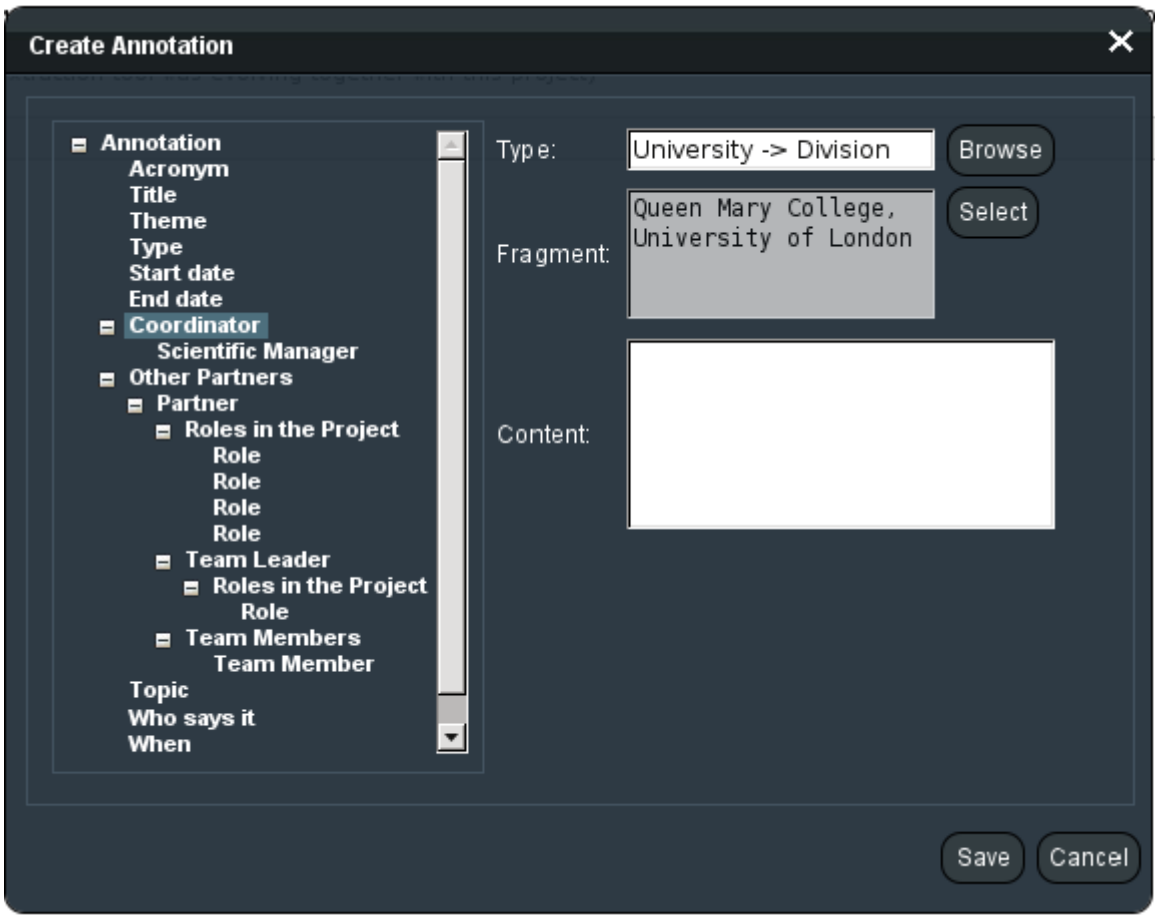
Načítání atributů daného typu bude nutné provádět jak při výběru typu anotace, a to zadáním názvu typu nebo výběrem pomocí dialogu *Types*, tak při výběru části dokumentu pro anotování. Bude nutné zajistit, aby uživatel mohl anotovat celý dokument nebo některou jeho část a nemohlo tedy dojít k nekonzistentnímu stavu. Při zobrazení dialogu *Annotate* proto bude stisknuto tlačítko *Annotate whole document*, které bude možné odškrtnout pouze výběrem určité části dokumentu o nenulové délce.

S načítáním atributů bude třeba zajistit správné načtení strukturovaných atributů, které

ve své vlastnosti nazvané *Type* neobsahují název typu, ale URI daného strukturovaného typu. Z této URI bude třeba získat název typu atributu, případně cestu, je-li typ atributu podtypem některého z typů. Při načítání atributů budou načteny atributy pouze do jedné úrovně stromu, nebudou tedy načítány atributy typů strukturovaných atributů. Každý typ strukturovaného atributu totiž může obsahovat další strukturované atributy, a tak by mohlo dojít k nekonečnému načítání atributů. Uživatel pak v poli *Selection* strukturovaného atributu bude upozorněn, že neanotuje žádnou část dokumentu, ani nevytváří odkaz na existující anotaci. K načtení atributů strukturovaného atributu dojde teprve při výběru části či celého dokumentu pro anotování nebo při změně typu strukturovaného atributu.

Každý uzel stromu atributů bude reprezentovat právě jeden atribut, přičemž hodnoty atributů budou ukládány k daným uzlům a při přechodu mezi uzly budou správně zobrazeny. Kromě načítání atributů je třeba zajistit také jejich odstraňování v případě, že uživatel například dvakrát po sobě vybere jiný typ. Budou však odstraňovány pouze atributy, které nebudou mít vyplněnou hodnotu. Podobným způsobem bude zajištěna také manipulace s atributy strukturovaných atributů.

Kromě toho, že bude možné vytvořit anotaci obsahující atributy daného typu, je nyní také možné k anotaci přidat libovolný nový atribut – jednoduchý i strukturovaný. Každý atribut pak lze ze stromu atributů dané anotace také odstranit, toto přidávání a odstraňování se však týká pouze atributů vytvářené anotace, nikoliv atributů uložených u daného typu. Přidávání a odstraňování atributů daného typu bude rovněž třeba zajistit.



Obrázek 4.1: Návrh manipulace s atributy typů

Na obrázku 4.1 je zobrazen návrh manipulace s atributy typů, který je možné nalézt v prezentaci věnující se 4A Frameworku [3].

4.1.1 Strukturované atributy

Zatímco jednoduché atributy obsahují pouze hodnotu daného atributu, strukturované atributy mohou obsahovat vnořenou anotaci, odkaz na existující anotaci, mohou však být také bez hodnoty nebo typu *anyAnnotation*. V případě, že strukturovaný atribut má zadaný typ a obsahuje nějaké atributy, musí být vybrána část dokumentu pro anotování nebo odkaz na existující anotaci. V opačném případě se jedná o chybně vyplněný atribut a vytvářenou anotaci nebude možné uložit. Pokud strukturovaný atribut nemá vyplněný žádný typ, pak je považován za atribut typu „jakákoliv anotace“ a bude mu přiřazen typ *anyAnnotation*. V tomto případě však strukturovaný atribut nesmí odkazovat na jinou anotaci, anotovat žádnou část dokumentu a ani nesmí obsahovat žádné atributy.

Při editaci existující anotace je nutné zobrazit anotaci tak, jak byla vytvořena. Jestliže tedy některý z atributů bude typu *anyAnnotation*, bude opět nezbytné vytvořit atribut s nevyplněným typem. Pokud strukturovaný atribut bude obsahovat vnořenou anotací, musí být při editaci v poli *Selection* také správně zobrazen text části nebo celého dokumentu, který je anotován.

4.1.2 Odkaz na existující anotaci

Strukturovaný atribut může obsahovat odkaz na existující anotaci, v takovém případě bude v poli *Selection* daného atributu zobrazena URI odkazované anotace. Odkazovanou anotaci při vytváření nové anotace nebude možné žádným způsobem editovat, budou však načteny její atributy. K odkazované anotaci tedy nebude možné přidat nový atribut ani odstranit stávající, atributy odkazované anotace však budou k dispozici ve stromu atributů a bude možné je běžným způsobem zobrazit. Rovněž dojde k načtení vlastnosti *Content* odkazované anotace, ani tu však nebude možné editovat.

Uživatel může vytvořit odkaz pouze na takovou anotaci, která je stejného typu jako strukturovaný atribut. Jakmile vybere odkazovanou anotaci, dojde k jejímu načtení. Tomu bude předcházet odstranění všech atributů, které případně daný strukturovaný atribut má, bez ohledu na to, zda mají vyplněnou hodnotu či nikoliv. Pokud uživatel zruší odkaz na existující anotaci například výběrem části dokumentu pro anotování, budou odstraněny všechny načtené atributy odkazované anotace. Správné načtení odkazované anotace bude třeba zajistit také při editaci anotace, která takovou anotaci obsahuje.

4.1.3 Změna typů atributů

Každý atribut je některého z jednoduchých či strukturovaných typů, typy atributů však bude možné měnit. Při změně typu atributu bude nezbytné přenést jeho vyplněnou hodnotu. Uživatel bude mít možnost změnit typ atributu zadáním nového názvu typu nebo výběrem nového typu pomocí dialogu *Types*. Při obou těchto možnostech bude chování editoru anotací totožné. Při zadávání názvu typu budou uživateli nabízeny typy, které odpovídají zadanému textu, přičemž bude možné odlišit, které nabízené typy jsou jednoduché a které strukturované.

Pokud uživatel změní atribut jednoduchého typu na atribut jiného jednoduchého typu, dojde k přenosu vyplněné hodnoty atributu. V případě změny na strukturovaný atribut bude hodnota vložena do vstupního pole *Content*. Podobným způsobem bude třeba zajistit

přesun vyplněných hodnot při změně strukturovaného atributu na strukturovaný atribut jiného typu.

Se změnou typů atributů je třeba se zabývat přesunem atributů strukturovaného atributu, a to v případě jeho změny na jednoduchý atribut. V takovém případě budou všechny atributy, které mají vyplněnou hodnotu a ještě se nenacházejí ve stromu o úroveň výše, přesunuty právě do této úrovně. Pokud je některý atribut jednoduchého typu a tento typ bude změněn na neznámý či bude úplně odstraněn, bude tento atribut změněn na strukturovaný.

4.1.4 Kontrola hodnot atributů

Při uložení anotace bude třeba zkontrolovat, zda všechny atributy mají vyplněny korektní hodnoty. Pokud atribut není vyžadovaný, viz dále, nemusí mít vyplněnou hodnotu. Pokud však vyplněnou hodnotu má, musí tato hodnota odpovídat typu atributu. Jednoduchému atributu typu *Integer* uživatel bude mít možnost vyplnit celé číslo, jiná hodnota bude považována za chybnou, u atributu typu *GeoPoint* pak bude povolené také desetinné číslo. Hodnota atributu typu *String* nebude nijak omezena a hodnoty atributů typu *Date*, *Time* a *DateTime* se budou řídit podle standardu RFC 3339 [17]. Atribut typu *Boolean* bude mít povoleny pouze hodnoty *0* a *1* či řetězec *false* nebo *true*. U každého jednoduchého atributu budou navíc zobrazeny příklady hodnot, které lze pro daný typ vyplnit.

Strukturovaný atribut musí vždy bud' obsahovat vnořenou anotaci, odkaz na anotaci nebo být bez hodnoty či typu *anyAnnotation*, jak bylo popsáno výše. V jiném případě se bude jednat o chybně vyplněný atribut. Aby anotaci bylo možné uložit, musí být rovněž vyplněn její typ. Pokud vyplněný typ anotace nebo strukturovaného atributu neexistuje, bude při uložení anotace vytvořen.

4.1.5 Přidávání a odstraňování atributů

Uživatel bude mít možnost přidat nové atributy k atributům daného typu. U nově vytvořeného atributu bude možné nastavit, zda po uložení anotace má být daný atribut přidán k typu či nikoliv. Rovněž bude možné zvolit, zda atribut má být vyžadován. V takovém případě bude vždy před uložením anotace nezbytné vyplnit hodnotu daného atributu. Tato nastavení pak bude možné kdykoliv změnit. K přidávání atributů k daným typům dojde při uložení anotace, přičemž atributy budou přidány jak k typu anotace, tak k typům všech vnořených anotací. Pokud už daný typ obsahuje daný atribut, bude jej možné změnit, tedy nahradit jej atributem stejného jména, avšak jiného typu. Rovněž bude třeba zajistit správné chování přidávání atributů, pokud v některé úrovni stromu budou měněny atributy typu, který už byl změněn v některé z vyšších úrovní.

Při zobrazení každého atributu bude nezbytné kontrolovat, zda tento atribut patří mezi atributy daného typu či nikoliv. Pokud ne, uživateli bude nabízeno nastavení, zda atribut přidat. Pokud daný typ již atribut obsahuje, bude nabízeno nastavení s možností jeho změny. Protože nastavení, zda má být atribut přidán k atributům daného typu, a nastavení, zda atribut má být vyžadován, se k atributům vytvářené anotace neukládají, při editaci anotace bude nezbytné všem atributům nastavit tato nastavení na výchozí hodnoty. Nastavení, zda má být atribut vyžadován, se ukládá pouze k atributům daného typu, odkud bude načteno vždy při načítání atributů tohoto typu.

Každý atribut bude také možné odstranit z atributů daného typu. Při odstraňování atributu z vytvářené anotace se provede kontrola, a pokud atribut náleží danému typu, uživateli bude nabídnuto také odstranění z tohoto typu. Atributy typů bude tedy možné odstraňovat pouze jednotlivě.

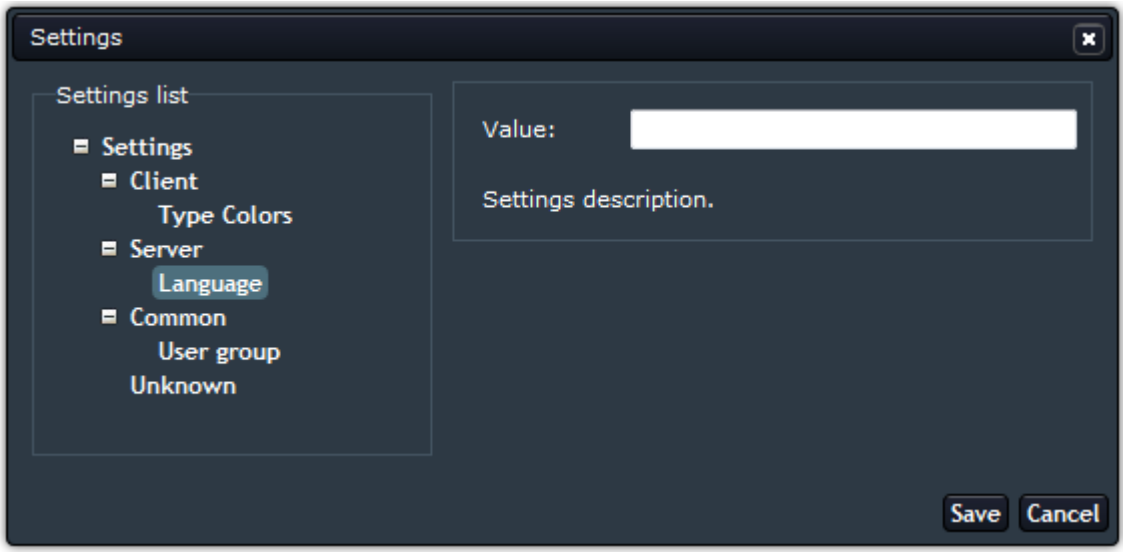
4.2 Nastavení

Nastavení editoru anotací by mělo před uživateli skrývat implementační detaily a poskytovat pohodlné zadávání hodnot jednotlivých nastavení. Za tímto účelem bude v dialogu *Settings* k dispozici strom podobný tomu, jaký lze vytvářet při vytváření nové anotace. Tento strom, který se bude nacházet v levé části dialogu, však nebude možné měnit a každý uzel bude představovat právě jedno nastavení. V pravé části dialogu pak bude možné zadat hodnoty jednotlivých nastavení.

Strom v dialogu nastavení bude mít kořenový uzel nazvaný *Settings*, který bude obsahovat uzly *Client*, *Server*, *Common* a *Unknown*, jak je zobrazeno na obrázku 4.2. V každém z těchto čtyř uzlů pak budou k dispozici konkrétní nastavení podle toho, o jaké nastavení se jedná, například nastavení týkající se pouze klienta bude možné nalézt pod uzlem *Client*. Výjimku bude tvořit uzel *Unknown*, který nebude obsahovat další uzly, ale tabulku se všemi neznámými nastaveními, viz dále.

Při zobrazení dialogu *Settings* se vytvoří uzly pro všechna známá nastavení, kterými v současné době jsou *DefaultUserGroup* (výchozí uživatelská skupina), *ServerLanguage* (jazyk serveru) a *ClientAnnotationTypeColor* (barvy typů anotací), a vyplní se hodnoty zaslané serverem. Pokud server některé ze známých nastavení nepošle, vyplní se výchozí hodnota a nastavení se při prvním uložení vytvoří. V případě, že editor anotací od serveru obdrží některé další, neznámé nastavení, bude toto nastavení zobrazeno v tabulce v uzlu nazvaném *Unknown*. Každý řádek této tabulky bude obsahovat právě jedno neznámé nastavení.

Zatímco nastavení *ServerLanguage* bude obsahovat pouze jedno vstupní pole pro vyplnění hodnoty, v nastavení *ClientAnnotationTypeColor* bude k dispozici tabulka pro zadání barev konkrétních typů anotací v jednotlivých řádcích této tabulky. Každý řádek bude navíc obsahovat také tlačítko *Select*, které umožní zobrazení dialogu pro pohodlný výběr typu a barvy. V nastavení *DefaultUserGroup* pak budou nabízeny k výběru všechny skupiny, do kterých je uživatel přihlášen. Před uložením nastavení budou vždy kontrolovány hodnoty všech nastavení a v případě, že některá z hodnot nebude korektní, nastavení nebude možné uložit. Uživatel také bude upozorněn na to, které nastavení je nesprávně vyplněno. Každý uzel stromu navíc bude obsahovat stručný popis, k čemu dané nastavení slouží.



Obrázek 4.2: Návrh podoby nastavení vytvořený pomocí nástroje Firebug

4.2.1 Barvy typů anotací

Uživatel bude mít možnost vybrat barvy typů v samostatném dialogu *Annotation Type Colors*. Tento dialog bude nabízet tři vstupní pole, přičemž jedno bude sloužit pro zadání typu, druhé pro výběr barvy a třetí pro zadání hodnoty alfa. Při zadávání typu budou uživateli nabízeny typy, jejichž název odpovídá zadanému textu, podobně, jako tomu je v dialogu pro vytváření anotací. Rovněž bude možné vybrat typ zobrazením dialogu *Types*. Pole pro zadání barvy typu by vždy mělo mít barvu pozadí dle vybrané barvy, přičemž pro výběr barvy bude využita některá z volně dostupných knihoven nabízejících paletu barev. Hodnota alfa, která slouží k nastavení průhlednosti, se bude zadávat v procentech.

Při potvrzení dialogu *Annotation Type Colors* bude nutné zkontrolovat vyplněné hodnoty a v případě nekorektní hodnoty uživateli nahlásit chybu. Pokud dialog bude vyplněný v pořádku, jeho hodnoty bude třeba přenést do patřičného řádku tabulky. Rovněž bude nutné do dialogu *Annotation Type Colors* při jeho zobrazení přenést hodnoty, které uživatel již zadal do daného řádku tabulky, kde je barva typu očekávána ve formátu RGBA. Tím bude zajištěna interakce mezi dialogem nastavení a dialogem pro snadný výběr barvy typů.

4.3 Nabízení anotací

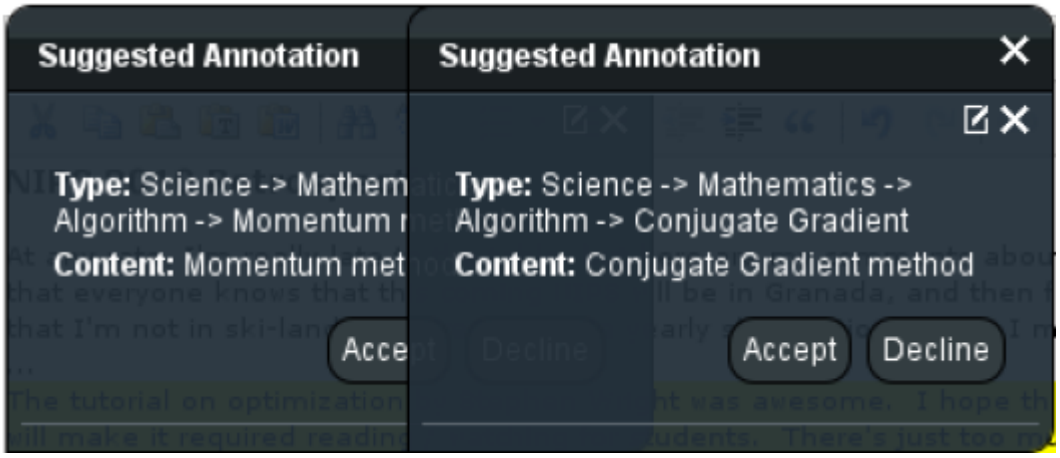
Editor anotací má umožňovat nabízení anotací. Nabídnutou anotací rozumíme anotaci, která se zobrazí v textu dokumentu a kterou uživatel musí nejprve potvrdit, aby se stala běžnou anotací. Pokud nabídku odmítne, server už ji znovu nepošle. Zobrazení nabídnutých anotací má být podobné jako zobrazení běžných anotací, nabídky však musejí být od anotací oddělené, protože s nimi mají být prováděny jiné akce a nesmí být například možné vytvořit odkaz na nabídnutou anotaci.

Editor anotací musí mít možnost nejen zobrazit přijaté nabídky, ale také uživateli umožnit o nabízení anotací požádat. K tomu bude nezbytné zajistit správnou tvorbu XML. Nabízení anotací si rovněž vyžaduje vytvoření dvou nových nastavení. Nastavení *Suggestion confidence* slouží jako práh důvěryhodnosti každé zaslané nabídky, pomocí *Refuse suggestion* pak uživatel může nastavit, zda má být vyžadováno potvrzení při odmítnutí některé z nabídek, viz dále. Návrh funkcionality nabízení anotací lze shrnout do následujících částí:

1. Žádost o nabízení anotací – Uživatel v dialogu *Suggest Annotations* požádá o nabízení anotací pro celý dokument nebo jeho vybranou část. V případě, že vybere určitý typ, budou mu nabízeny pouze anotace daného typu. Po potvrzení dialogu dojde k vytvoření XML a jeho zaslání serveru.
2. Vytvoření nabídnutých anotací – Editor anotací jako reakci na odpověď serveru vytvoří nabídky anotací, které mu server zaslal. Pokud má některá z nabídek hodnotu *confidence* nižší, než jakou si uživatel zvolil v nastavení *Suggestion confidence*, dojde k automatickému odmítnutí nabídky. Zároveň se uloží stav dialogu *Suggest Annotations*, aby mohl být při opětovném zobrazení dialogu obnoven. Každá nabídka bude instancí dané třídy, přičemž tyto instance bude shromažďovat třída, která umožní manipulaci se všemi nabídnutými anotacemi.
3. Zobrazení nabídnutých anotací – Při umístění myši nad určitý fragment textu se zobrazí dialog dané nabídnuté anotace. Ten bude zahrnovat název typu nabídnuté anotace, atributy této nabídky a případně i možnost zobrazení vnořené nabídky. V tomto dialogu bude uživatel mít možnost přijmout, odmítnout či editovat danou nabídku.

4. Přijmutí nabídnuté anotace – Pokud uživatel přijme danou nabídku, editor anotací na žádost serveru tuto nabídku odstraní a vytvoří novou anotaci odpovídající potvrzené nabídce. Při potvrzení bude serveru zasláno XML s patřičným dočasným identifikátorem nabídky. Editor anotací musí na žádost serveru nejen vytvořit, ale také odstranit danou nabídku, k čemuž bude sloužit metoda v třídě shromažďující všechny nabídky.
5. Odmítnutí nabídnuté anotace – Odmítnutí dané nabídky bude vyžadovat potvrzení dle aktuálního nastavení *Refuse suggestion*. V případě odmítnutí bude serveru zasláno XML s identifikátorem dané nabídky, server požádá editor anotací o odstranění nabídky a příště již tuto anotaci nenabídne.
6. Editace nabídnuté anotace – Každou nabídku bude možné editovat podobně jako anotaci. Při potvrzení editované nabídky dojde k vytvoření XML nové anotace, ovšem s dočasným identifikátorem dané nabídky. Server v takovém případě požádá editor anotací o odstranění dané nabídky a vytvoření nové anotace.
7. Vnořená nabídnutá anotace – Vnořené nabídnuté anotace budou v dokumentu zobrazeny stejně jako nabídnuté anotace, nesmí však být možné je samostatně přijmout ani odmítnout, jejich editace pak musí přejít na editaci rodičovské nabídky. V případě, že je nabídka vnořena vícekrát, dojde k editaci nabídnuté anotace na nejvyšší úrovni.
8. Zrušení nabízení anotací – Uživatel bude mít možnost kdykoliv změnit podmínky nabízení anotací, například vybrat jinou část dokumentu či typ nabízených anotací. Bude také možné nabízení úplně zrušit, v takovém případě dojde k odstranění všech zobrazených, nepotvrzených nabídek.

Na obrázku 4.3 je zobrazen návrh podoby nabídnutých anotací, který lze nalézt v prezentaci o 4A Frameworku [3].



Obrázek 4.3: Návrh podoby nabídnutých anotací

4.3.1 Žádost o nabízení anotací

Uživatel bude mít možnost zažádat o nabízení anotací v dialogu *Suggest Annotations*. Tento dialog bude nabízet podobnou funkcionalitu, jakou už disponuje dialog pro vytváření anotací. Jeho hlavní část budou tvořit dvě vstupní pole označené *Selection* a *Type*. V poli *Selection* se uživateli pouze zobrazí, jakou část dokumentu má vybránu, do pole *Type* bude

mít možnost zadat typ anotací.

Pro nabízení anotací pro celý dokument bude sloužit tlačítko *Select whole document*, pro zrušení nabízení tlačítko *No suggestions*. V případě, že bude některé z těchto tlačítek stisknuto, v poli *Selection* se zobrazí speciální text. Při zadávání typu se uživateli nabídnou typy, které odpovídají zadanému řetězci, jako je tomu v dialogu *Annotate*. Typ bude možné vybrat také pomocí tlačítka *Browse*, které zobrazí dialog typů.

Dialog *Suggest Annotations* musí vždy zobrazovat, pro jakou část dokumentu a jakého typu jsou anotace nabízeny, proto je třeba uchovávat jeho stav, který se obnoví při opětovném zobrazení dialogu. K tomu bude potřeba nový objekt a metody, které provedou uložení stavu dialogu a poté jeho načtení.

4.3.2 Zobrazení nabídnutých anotací

Nabídnuté anotace se v dokumentu zobrazí podobně jako běžné anotace, tedy vyznačením fragmentu textu, ke kterému daná nabídka patří. Aby bylo možné anotace a nabídky snadno odlišit, budou nabídnuté anotace navíc čárkovaně orámovány. Dialog daného fragmentu bude mít stejnou podobu jako dialog fragmentu anotace, k dispozici však budou jiná tlačítka s jinými akcemi – přijmout, odmítnout, editovat.

Nabídky anotací navíc budou k dispozici pohromadě v jediném dialogu nazvaném *Suggested Annotations*. Tento dialog není ve specifikaci editoru anotací, poskytne však možnost rychlého potvrzování nabídek. Aby u každé nabídky v tomto dialogu bylo jasné, ke které části dokumentu se vztahuje, při umístění myši nad libovolnou nabídku se zobrazí dialog daného fragmentu. Při kliknutí na nabídku dialog zůstane zobrazený stejným způsobem, jako při kliknutí na daný fragment. S nabídnutými anotacemi v dialogu *Suggested Annotations* bude možné provádět stejné akce jako v dialogu daného fragmentu.

Při manipulaci s nabídnutými anotacemi bude vždy třeba zjistit, zda se nejedná o vnořenou nabídku, a případně mít k dispozici odkaz na rodičovskou nabídku. Při vytváření nabídek tedy bude třeba každé vnořené nabídce přiřadit dočasný identifikátor rodičovské nabídky, a tím bude možné od sebe nabídky snadno rozpoznat.

Kapitola 5

Implementace

Tato kapitola stručně popisuje implementaci nejdůležitějších oprav a vylepšení dostupného editoru anotací, které byly navrženy v kapitole 4. Blíže se pak věnuje závažnějším problémům, které se při implementaci vyskytly a které bylo třeba řešit.

5.1 Manipulace s atributy typů

Protože požadavek na manipulaci s atributy vzniká většinou jako reakce na akci uživatele, přibyla implementace metod, které obsluhují dané události. Implementace manipulace s atributy typů vyžadovala také zásah do existujícího dialogu pro vytváření anotací, kde bylo třeba doplnit stávající funkcionalitu a rovněž přidat nové funkce.

- Třída *AEd.ui.DlgAnnotate* – implementuje funkcionalitu dialogu pro vytváření a editaci anotací.
- Třída *AEd.editors.Editor* – třída, která slouží pro vytvoření instance editoru anotací a obsluhu událostí vzniklých v rámci této instance.

Hlavní úlohu při manipulaci s atributy typů, která je zobrazena na obrázku 5.1, plní metoda *loadAttrsOfType()* provádějící načtení atributů. Tato metoda je volána jak při výběru daného typu, tak při označení části textu pro anotování nebo výběru celého dokumentu, z čehož vyplývají důležité požadavky, které metoda musí splňovat. Tím nejdůležitějším je zachování pořadí atributů při opětovném načtení.

Metoda *loadAttrsOfType()* nejprve odstraní atributy, kterým uživatel zatím nevyplnil hodnotu a které zároveň nejsou atributy daného typu. To zaručuje, že uživatel nepřijde o vyplněné atributy a zároveň nebude změněno pořadí atributů v případě, že jsou načítány atributy totožného typu, například při označení části dokumentu. V případě, že některý z atributů, který náleží danému typu a nebyl tedy odstraněn, má změněný typ, provede se jeho odstranění dodatečně při procházení atributů daného typu. Pokud po tomto ověření daný atribut typu ještě není ve stromu, přidá se, v opačném případě je ignorován.

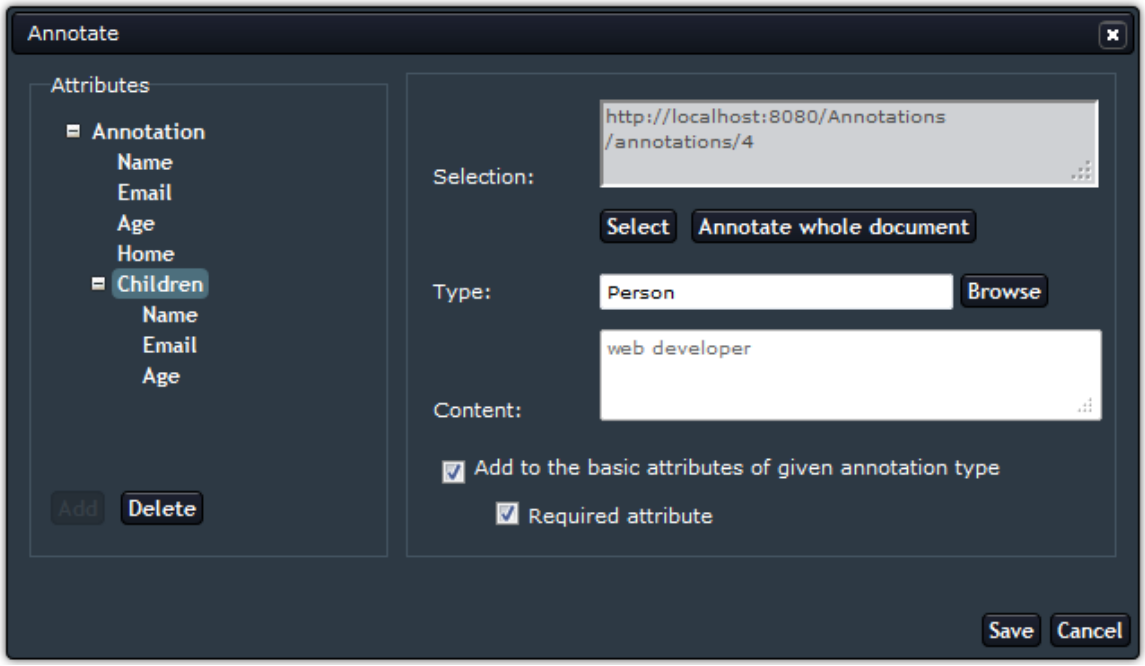
O kontrolu hodnot atributů, které uživatel vyplnil, se před uložením nové anotace stará metoda *checkAttributes()*. Tato metoda prochází všechny uzly stromu a dle typu každého atributu ověří zadanou hodnotu. Protože strukturovaný atribut může mít další atributy, je v takovém případě nezbytné metodu volat rekurzivně, aby byly zkontrolovány uzly ve všech úrovních stromu.

Metoda *loadAnnotation()*, která doposud sloužila pouze pro načtení existující anotace v případě její editace, byla dále vylepšena tak, aby umožňovala také načtení odkazované

anotace. Metoda rozlišuje, zda načítá anotaci či pouze odkazovanou anotaci. V prvním případě dojde nejprve k odstranění stávajícího stromu a vytvoření nového, do jehož kořene je načtena daná anotace spolu s jejími atributy. V případě odkazované anotace je strom zachován a anotace a její atributy jsou načteny do vybraného uzlu (strukturovaného atributu).

Třída *AEd.editors.Editor* vytváří instance většiny tříd a může tedy přistupovat k objektům a volat metody z různých částí editoru anotací. Tato třída dále implementuje obslužné metody, jejichž funkcionalita může být zahrnuta přímo v těchto metodách nebo mohou být volány metody z jiných tříd. To záleží především na tom, zda daná obslužná metoda vyžaduje práci s objekty a metodami dalších tříd či nikoliv.

U každého atributu lze nastavit, zda má být přidán mezi atributy daného typu, a v případě, že daný typ již takový atribut obsahuje, nabízí se jeho změna. Tento koncept si vyžádal kontrolu atributů typu při každém zobrazení a také každé změně kteréhokoliv z atributů dané anotace.



Obrázek 5.1: Manipulace s atributy typů

5.1.1 Atribut typu „jakákoliv anotace“

V průběhu implementace oprav a vylepšení editoru anotací vznikl požadavek, aby uživatel mohl uložit atribut typu „jakákoliv anotace“. Taková anotace by byla reprezentována strukturovaným atributem, který by neměl vyplněný typ, neanotoval by žádnou část dokumentu a ani neodkazoval na již existující anotaci. Za běžných okolností by takový atribut byl považován za chybný a nebylo by možné jej uložit, proto bylo nezbytné zajistit, aby k takovému atributu byl při odeslání doplněn speciální typ *anyAnnotation*. Naopak při obdržení anotace s tímto typem atributu klient musí zajistit správnou interpretaci atributu, tedy opět s prázdným typem. Uživatel tak může vytvořit atribut daného jména a až později se rozhodnout, jakou hodnotu bude mít, což je žádoucí především při importu ontologií. O možnost vytvoření atributu tohoto typu byl dodatečně doplněn také návrh manipulace s atributy typů z důvodu jeho kompletnosti.

5.1.2 Pomocné metody

Při manipulaci s atributy typů bylo třeba implementovat také několik pomocných metod včetně metod pro kontrolu formátů data a času a metody pro konverzi textového řetězce na hodnotu typu *Boolean*.

Ověření formátů data a času si vyžádala kontrola hodnot atributů typu *Date*, *Time* a *DateTime*. Jazyk JavaScript nenabízí funkce, které by takovou kontrolu provedly, a tak bylo nezbytné implementovat vlastní metody. Tyto metody provádějí zpracování zadaného textového řetězce a v případě, že tento řetězec odpovídá očekávanému formátu data nebo času, jde o povolenou hodnotu daného typu atributu. Ověření hodnoty atributu typu *DateTime* vyžadovalo použití metod pro kontrolu data i času, metoda pro kontrolu času pak využívá také metodu pro ověření časového pásma. [10]

V jazyce JavaScript dále není možné porovnat textový řetězec *true* či *false* s těmito hodnotami typu *Boolean*. Tyto hodnoty se nerovnají a pro jejich správné porovnání je třeba provést konverzi. Jednou z možností, jak převést textový řetězec na hodnotu typu *Boolean*, je vlastní metoda, která vrátí *true* v případě, že jejím vstupem je textový řetězec *true*, v jiném případě vrátí *false*. [22]

5.2 Nastavení

Aby dialog nastavení editoru anotací mohl dle návrhu nabídnout strom, ve kterém uzly představují jednotlivá nastavení, bylo nezbytné upravit HTML dokument, který udává podobu dialogu nastavení, a také doplnit novou funkcionalitu právě dialogu představujícímu nastavení editoru anotací.

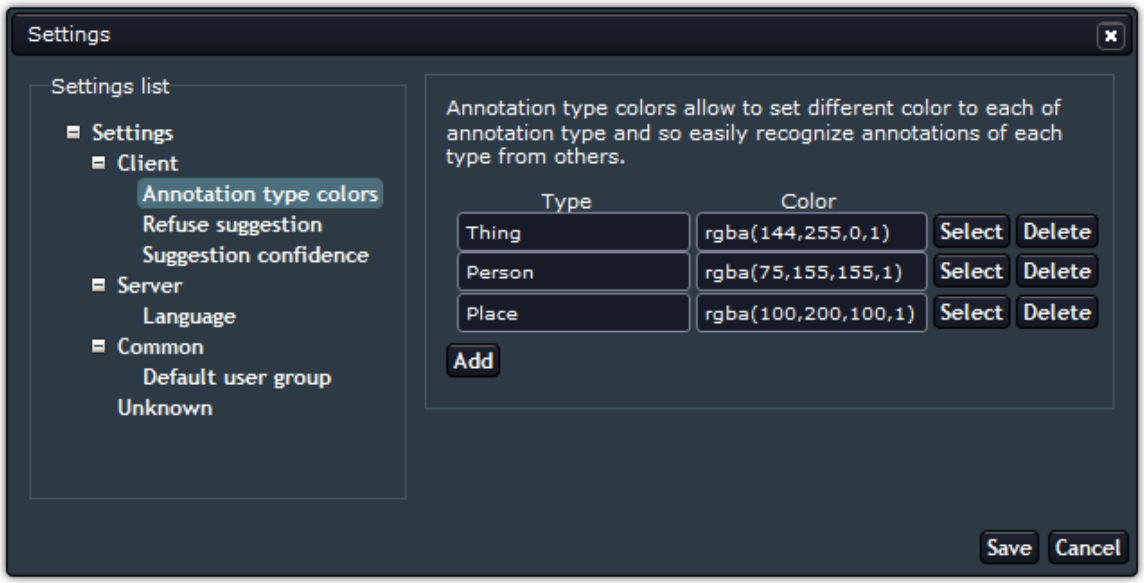
- Dokument *settings.html* – udává podobu dialogu nastavení.
- Třída *AEd.ui.DlgSettings* – implementuje funkcionalitu dialogu pro nastavení editoru anotací.

Protože je třeba, aby i tento dialog reagoval na akce uživatele například při potvrzení nastavení, bylo také nezbytné implementovat několik obslužných metod. Metoda *checkSettings()* se stará o kontrolu hodnot nastavení a v případě, že je některé nekorektní, nedovolí nastavení uložit. Tato metoda postupně prochází každý uzel stromu, který představuje konkrétní nastavení, a provede ověření hodnoty podle toho, o jaké nastavení se jedná. Pokud metoda zkontroluje celý strom a nenalezne nekorektní hodnotu, nastavení je v pořádku. V opačném případě se zastaví na daném uzlu a tento uzel nastaví jako vybraný.

Implementace dialogu nastavení vyžadovala vylepšení metod pro práci s tabulkou tak, aby je bylo možné využít pro libovolný HTML element *table*. Dále bylo třeba doplnit metodu pro přidávání řádků tabulky o možnost přidání dalšího sloupce pro tabulku s nastavením barev typů, který by obsahoval tlačítko s možností zobrazení dialogu *Annotation Type Colors* pro snadný výběr barvy. V rámci lepší práce s tabulkami byla také implementována metoda, která odstraní všechny prázdné řádky dané tabulky.

Protože dialog *Annotation Type Colors* usnadňuje nastavení barvy typů, měl by být modální. Tento dialog však umožňuje zobrazení dalšího modálního dialogu pro výběr typu, a jak jsem zjistil v průběhu implementace, knihovna pro uživatelské rozhraní nedokáže správně zobrazit dva modální dialogy. Z tohoto důvodu je dialog *Annotation Type Colors* implementován jako běžný dialog.

Na obrázku 5.2 je zobrazen dialog nastavení s tabulkou pro výběr barvy typů.



Obrázek 5.2: Vylepšené nastavení editoru anotací

5.2.1 Výběr barvy typů

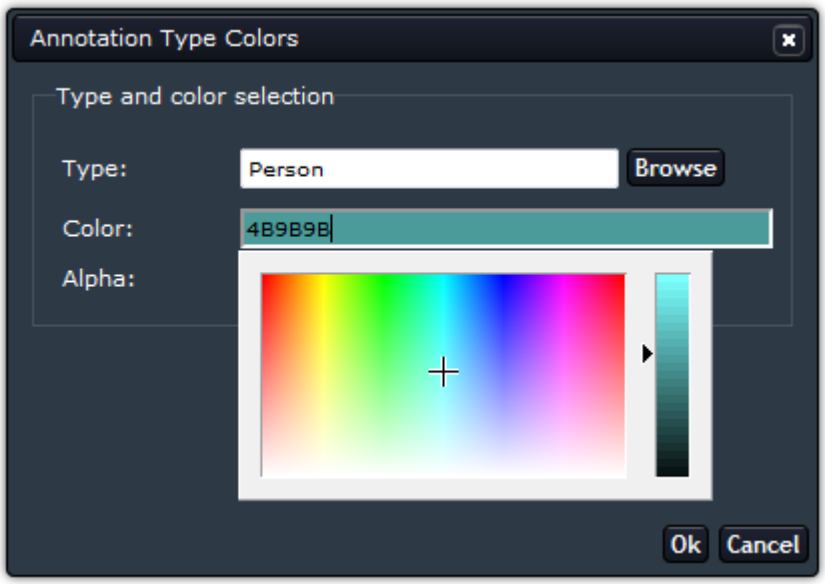
V nastavení editoru anotací je možné zvolit barvu typů, a tak od sebe odlišit anotace jednotlivých typů. K dispozici je samostatný dialog *Annotation Type Colors*, který umožňuje pohodlný výběr barvy. Implementace tohoto dialogu si vyžádala vytvoření HTML dokumentu představujícího jeho vzhled a také třídy, která implementuje jeho funkcionalitu.

- Dokument *typecolors.html* – udává podobu dialogu pro výběr barvy typů anotací.
- Třída *AEd.ui.DlgTypeColors* – implementuje funkcionalitu dialogu pro výběr barvy typů anotací.

Základem dialogu pro výběr barvy typů jsou tři vstupní pole, která slouží pro zadání typu, barvy a hodnoty alfa. Stejně jako v dialogu pro vytvoření anotace jsou i zde uživateli nabízeny typy, jejichž názvy odpovídají zadanému textu, barva je pak očekávána v hexadecimálním tvaru a alfa v procentech. Pro výběr barvy byla využita knihovna JSColor [18], která uživateli poskytuje jednoduchou paletu barev a nastavuje pozadí daného vstupního pole dle vybrané barvy. Dialog s touto knihovnou je zobrazen na obrázku 5.3.

5.2.2 Převod mezi decimální a hexadecimální soustavou

Nastavení editoru anotací pracuje s barvami typů ve formátu RGBA, knihovna JSColor však podporuje pouze hexadecimální tvar. Proto bylo třeba zajistit převod číselných hodnot z decimální do hexadecimální soustavy [39] a zpět [38], k čemuž byly využity dostupné algoritmy. Uživatel si nyní může vybrat, ve kterém z těchto formátů barvu zadá. Při převodu hodnot bylo třeba se zabývat celočíselným dělením v jazyce JavaScript, kterého jsem dosáhl pomocí zaokrouhlování [23].



Obrázek 5.3: Výběr barvy typů anotací

5.3 Nabízení anotací

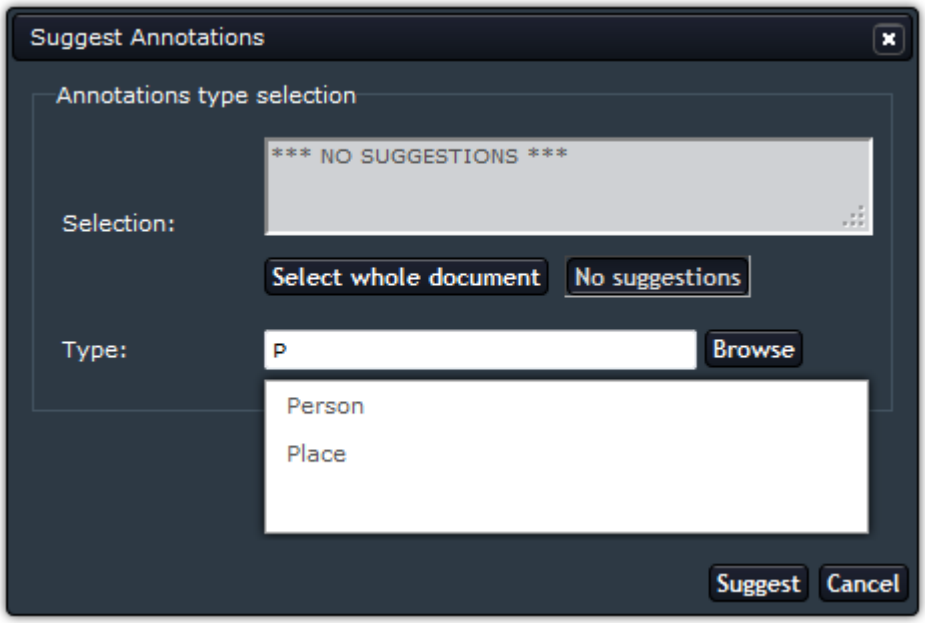
Nabízení anotací obnášelo implementaci dialogu *Suggest Annotations*, který uživateli umožňuje požádat server o nabízení, ale také tříd, které představují jak jednotlivé nabídky, tak tyto nabídky shromažďují.

- Dokument *suggestanno.html* – udává podobu dialogu pro žádost o nabízení anotací.
- Třída *AEd.ui.DlgSuggestAnnotations* – implementuje funkcionalitu dialogu *Suggest Annotations* pro žádost o nabízení anotací.
- Třída *AEd.ui.DlgSuggestedAnnotations* – implementuje funkcionalitu dialogu *Suggested Annotations*, který zobrazuje všechny nabídnuté anotace.
- Třída *AEd.entities.Suggestion* – představuje jednu nabídnutou anotací.
- Třída *AEd.entities.SuggestionsManager* – shromažďuje a spravuje nabídnuté anotace, vytváří instance třídy AEd.entities.Suggestion.

Aby editor anotací zůstal jednotný, zobrazení nabídnutých anotací je implementováno podobně jako zobrazení anotací, s nabídkami však lze provádět jiné akce. Nabídnuté anotace je možné dále nalézt pohromadě v jediném dialogu *Suggested Annotations*, kde jsou umístěny podobně jako anotace celého dokumentu v dialogu *Document Annotations*.

Dialog, který uživateli umožňuje zažádat o nabízení anotací, musí vždy odpovídat aktuálnímu stavu nabízení a tedy uchovávat svůj stav, který se načte při opětovném zobrazení dialogu. Změnou stavu nabízení však není potvrzení dialogu, ale teprve až kladná odpověď serveru. Pouze tím je zajištěno, že nemůže dojít k situaci, kdy uživatel zažádá o nabízení anotací, server však z důvodu neočekávané chyby nabídky nepošle, ale dialog *Suggest Annotations* uloží a zobrazí stav, že nabídky jsou zasílány.

Na obrázku 5.4 je zobrazen dialog *Suggest Annotations*, který uživateli umožňuje zažádat o nabízení anotací nebo žádost zrušit. K dispozici je výběr typu, díky kterému mohou být nabízeny pouze anotace tohoto typu.



Obrázek 5.4: Dialog pro žádost o nabízení anotací

5.3.1 Zobrazení nabídnutých anotací

Nabídnuté anotace jsou uživateli zobrazeny jak v textu dokumentu, tak v samostatném dialogu. Ve skutečnosti jde o dvě různé instance nabídek, neboť je třeba zajistit jejich odlišné chování. S tím vznikl problém zajištění konzistence na obou místech zobrazení, kterým se bylo třeba zabývat. Nabídnuté anotace, které se zobrazují v dialogu *Suggested Annotations*, jsou při každé změně (potvrzení či odmítnutí nabídky uživatelem nebo zaslání nových či odstranění stávajících nabídek serverem) smazány a vytvořeny dle aktuálního stavu nabídek, které uchovává třída *AEd.entities.SuggestionsManager*. Tím je zajištěno, že dialog *Suggested Annotations* vždy zobrazuje aktuální nabídky, což je patrné také z obrázku 5.5.

Obrázek 5.5: Zobrazení nabídnutých anotací

Při umístění myši nad některou z nabídek v dialogu *Suggested Annotations* bylo nezbytné zobrazit dialog daného fragmentu, aby uživatel měl přehled o tom, ke které části textu daná nabídka patří. Při vytváření všech nabídnutých anotací v tomto dialogu bylo tedy zapotřebí v cyklu *for* vytvořit *handler* ke každé nabídce, který by zobrazil dialog daného fragmentu. K tomu bylo nutné využít volání pomocné metody, která provede kopii iterační proměnné a zajistí tak správné chování [24].

Kapitola 6

Testování

V této kapitole se zabývám testováním navržených oprav a vylepšení editoru anotací. Její úvod je věnován prostředí, ve kterém bylo testování prováděno, a nástrojům, které jsem k testování využil. Zamýšlím se zde také nad možností ladění v jiných prostředích a popisuji testovací data, nad kterými jsem testy prováděl. V další části kapitoly se pak věnuji vybraným nedostatkům, které nebylo možné odhalit v průběhu návrhu, jejich příčinám, a také řešení.

6.1 Testovací prostředí

Jazyk JavaScript je považován za jeden z nejnáročnějších jazyků, co se týče jeho ladění. To je způsobeno nejen jeho dynamickými vlastnostmi, ale také tím, že mnohá léta neexistovaly kvalitní ladící nástroje. Poté, co jazyk JavaScript začal nabízet příkazy *try-catch* a *throw*, které vývojářům umožnily reagovat na chyby již v průběhu jejich vzniku, se začaly objevovat také vývojové nástroje pro různé webové prohlížeče. V současné době jsou tyto nástroje k dispozici pro většinu nejpoužívanějších prohlížečů. [40]

6.1.1 Použité nástroje

Vývoj editoru anotací probíhal ve webovém prohlížeči Firefox, a tak byly použity nástroje, které jsou pro tento prohlížeč k dispozici. Firefox chyby vzniklé při interpretaci kódu vypisuje do chybové konzole, kde lze vždy nalézt adresu URL a čísla řádku, na kterém k chybě došlo. Jazyk JavaScript disponuje funkcí *alert()*, která je často využívána při ladění kódu a odhalování chyb, nenabízí však takové možnosti jako nástroj Firebug.

Firebug [15] je v současné době jeden z nejoblíbenějších vývojových nástrojů pro webový prohlížeč Firefox, který umožňuje sledování a editaci HTML, CSS a JavaScriptu. Díky tomuto nástroji je možné odhalit chyby webových aplikací, s Firebugem lze provádět krokování kódu, vytváření zarážek či sledování hodnot zvolených proměnných. Firebug navíc ke všem načteným stránkám přidává globální objekt nazvaný *console*, který obsahuje řadu metod sloužících pro odhalení chyb. Metoda *console.log()*, která byla nejčastěji využita při vývoji editoru anotací, slouží pro výpis textu či obsahu proměnných do konzole nástroje Firebug. Tato metoda může obsahovat libovolný počet argumentů.

6.1.2 Inicializační skript databáze

Editor anotací po celou dobu vývoje i testování komunikoval se serverem, jehož obsah databáze byl vždy stejný. Měnil se pouze vytvářením typů a anotací přímo v editoru anotací, v případě neočekávaného chování byl znovu obnoven původní stav databáze. K naplnění databáze zkušebními daty posloužil dostupný inicializační skript, ve kterém byly v průběhu vývoje opravovány drobné nedostatky týkající se atributů typů. S takovou databází obsahující minimální data pak bylo možné při vývoji pracovat, a tak například při vytváření a editaci anotací, při manipulaci s atributy typů nebo při potvrzování či odmítání nabídnutých anotací zhodnotit aktuální stav databáze a tedy správnost prováděných operací.

6.1.3 Testovací dokument

Editoru anotací bude v praxi vždy předána adresa dokumentu, který je anotován, a to prostřednictvím daného CMS. V současné době je tato adresa vložena do webové stránky, která dále obsahuje TinyMCE editor s možností spustit editor anotací, napevno. Napevno je také vložen testovací obsah editoru TinyMCE.

6.1.4 Alternativní testování

Editor anotací se prozatím vyvíjí a testuje ve webovém prohlížeči Firefox, implementace pro WYSIWYG textové editory je však nezávislá na prohlížeči, a tak bude nezbytné v budoucnu odladit editor anotací také pro další prohlížeče. Stejně jako ve Firefoxu jsou i zde k dispozici nástroje, které vývoj webových aplikací usnadňují.

Internet Explorer umožňuje ladění skriptů napsaných v jazyce JavaScript pomocí integrovaného vývojového nástroje. Ten nabízí chybovou konzoli, zobrazuje místa kódů, na kterých došlo k chybám, umožňuje krokování a další. Podobnou funkcionalitou dnes disponuje také ladící nástroj prohlížeče Opera nazvaný Dragonfly, přičemž v obou zmíněných prohlížečích je třeba nástroj aktivovat. Ani Google Chrome však nezůstává pozadu a rovněž přináší ladící nástroje určené pro vývojáře webových aplikací. Editor anotací tedy bude možné otestovat také v těchto dalších, zmíněných prostředích.

6.2 Odhalené nedostatky

Díky zmíněným ladícím nástrojům a testovacím datům bylo možné odhalit chyby, které se projevily až v průběhu implementace a bylo nezbytné je dodatečně odstranit. Často se jednalo o nedostatky vzniklé při řešení některého z navržených vylepšení, které však nebylo funkční pro stávající implementaci editoru anotací. Příkladem může být kolize manipulace s atributy typů při vytváření nové a při editaci existující anotace. Za účelem testování byly také doplněny některé připravené, avšak chybějící části editoru anotací.

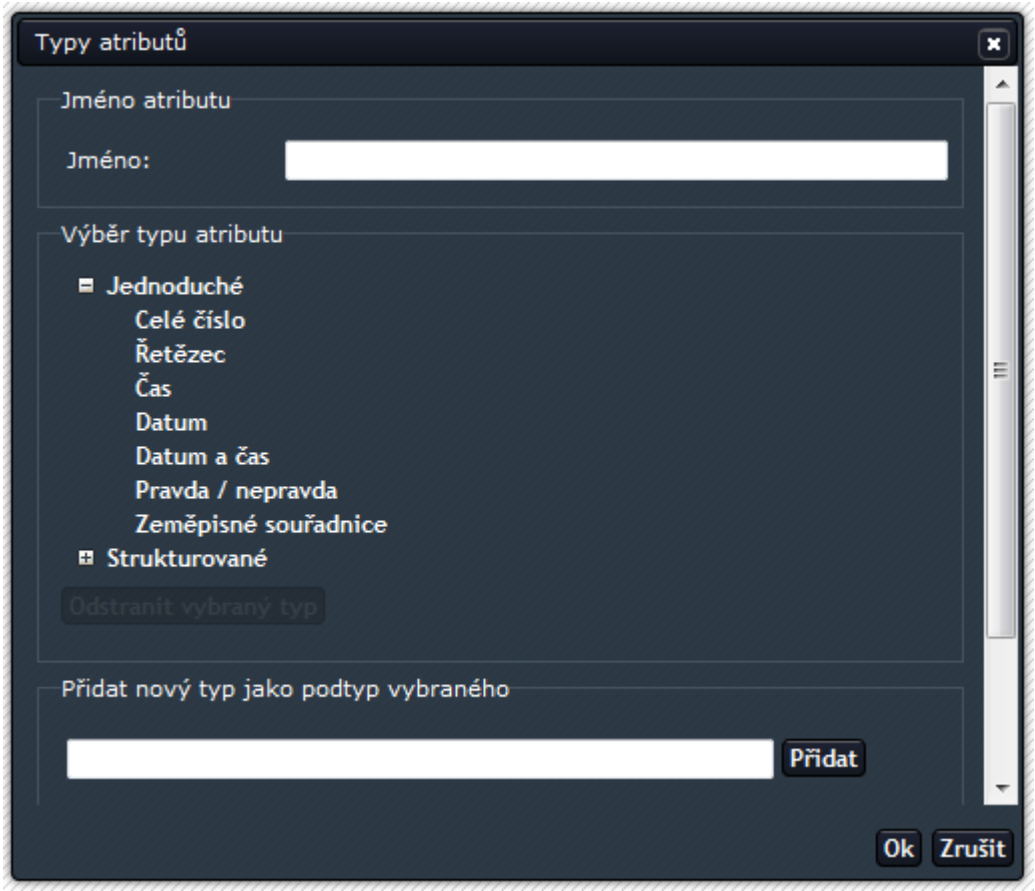
6.2.1 Lokalizace jednoduchých typů

Aby bylo možné odhalit chyby editoru anotací spojené s jeho lokalizací, bylo nezbytné vytvořit lokalizaci pro zvolenou zemi. Za tuto zemi jsem vybrat Českou republiku a provedl překlad editoru anotací z anglického do českého jazyka. Tento jazyk na rozdíl od angličtiny nabízí diakritiku, a tak díky němu lze dobře otestovat chování editoru anotací po změně výchozího jazyka.

Nedostatek, který jsem v této souvislosti objevil, se týkal lokalizace jednoduchých typů, kterou editor anotací umožňuje. Pokud uživatel zadává typ anotace či atributu, typy jsou mu nabízeny dle napsaného textu. Typy, které ve svém názvu obsahují diakritiku, však nebyly vyhledány správně. Proto bylo nezbytné z lokalizovaného názvu typu a textu, který zadává uživatel, diakritiku odstranit a teprve poté dané texty porovnat. To zajistilo správné chování, kdy například při zadání slova *řet* je uživateli nabídnut typ *Řetězec*. Protože v jazyce JavaScript neexistuje funkce, která by z daného řetězce odstranila diakritiku, byla implementována vlastní metoda, která písmena s diakritikou vymění za stejná písmena bez diakritiky.

Závažnější problém, který se vyskytl s lokalizací jednoduchých typů, byla absence skutečného jména typu. To bylo důležité zachovat, neboť se od něj odvíjí například povolené hodnoty atributu daného typu a obecně editor anotací pracuje výhradně se skutečnými názvy typů. Lokalizace si proto vyžádala ke každému napovědněmu typu přidat vlastnost, která by nesla skutečný název - například typ *String*, který uživatel po lokalizaci nalezne jako *Řetězec*. Ve chvíli, kdy pak z nabídnutých typů uživatel vybere typ *Řetězec*, načte se skutečný název tohoto typu, tedy *String* a s tím se pak dále pracuje.

Na obrázku 6.1 je zobrazen dialog pro vytvoření nového atributu s lokalizací jednoduchých typů atributů.



Obrázek 6.1: Lokalizace jednoduchých typů atributů

6.2.2 Reakce tlačítek na stisk klávesy

HTML elementy reagují na stisk klávesy *Enter*, což je v případě editoru anotací nežádoucí. Pokud uživatel provádí výběr typu z nabídnutých typů dle zadaného textu a k potvrzení použije právě klávesu *Enter*, dojde s výběrem typu také ke stisku některého z tlačítek v daném dialogu. Tato chyba byla odhalena v průběhu testování a byla odstraněna tak, že všem elementům v HTML dokumentech příslušných dialogů byl přidán atribut *type* s hodnotou *button*. Tím došlo k zamezení reakcí tlačítek na stisk potvrzovací klávesy.

Kapitola 7

Závěr

V rámci této bakalářské práce jsem provedl analýzu dostupného editoru anotací, nastudoval technologie, které editor anotací využívá, a navrhl jeho opravy a vylepšení. Tato vylepšení jsem v průběhu letního semestru implementoval a provedl testování a opravy nalezených problémů.

Návrh vylepšení editoru anotací byl zaměřen na nejdůležitější nedostatky, které jsem objevil v dostupném editoru anotací. Jednalo se o manipulaci s atributy typů, kterou editor anotací žádným způsobem neumožňoval, nevyhovující nastavení editoru anotací a absenci nabízení anotací. Díky implementaci těchto vylepšení nyní editor anotací poskytuje nejdůležitější funkcionalitu, která je uvedena v jeho specifikaci.

Vylepšený editor anotací, který vznikl v rámci této bakalářské práce, se nyní bude využívat v komponentě pro sémantické anotování v rámci 4. pracovního balíčku evropského projektu Decipher (EU-7FP-IST, 270001, 7E11023).

Editor anotací pro WYSIWYG textové editory je jedním ze dvou klientů systému pro kolaborativní poloautomatické strukturování znalostí v reálném čase, který již nabízí minimální specifikovanou funkcionalitu nutnou pro jeho využití. Druhým klientem je doplněk pro webový prohlížeč Firefox, jehož implementace sice obsahuje větší část funkcionality definované ve 4A Frameworku, ale v některých částech se mírně odchyluje od specifikace a např. práce s atributy typů anotací a samotných anotací je zde méně propracovaná (ošetření změn typů atributů apod.).

Editor anotací byl navržen tak, aby jej do budoucna bylo možné dále vyvíjet. V současné době editor anotací neumožňuje práci s dokumentem, který byl během anotování editován, bude proto nezbytné zajistit zasílání změn provedených v daném dokumentu serveru. Editor anotací by měl rovněž umožňovat vytvoření seznamu vnořených anotací či odkazů na existující anotace v jediném strukturovaném atributu. I tuto funkcionalitu bude možné implementovat v průběhu dalšího vývoje, na kterém budu pokračovat v rámci výzkumu ve výzkumné skupině NLP.

Literatura

[1] CKSource - Frederico Knabbe: *CKEditor* [online]. Dostupné z: http://ckeditor.com/, [cit. 2012-04-11].

[2] Dytrych, J.: *Webové služby a komponentní technologie (pojednání k tématu disertační práce)*. Brno University of Technology, Faculty of Information Technology, 2011.

[3] Dytrych, J.: *4A Framework - Annotations Anywhere, Annotations Anytime* [online]. Poslední modifikace: 2011-04-28 [cit. 2012-04-11].

[4] Garrett, J. J.: *Ajax: A New Approach to Web Applications* [online]. Dostupné z: http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications, Poslední modifikace: 2005-02-18 [cit. 2012-04-09].

[5] Gentics Software GmbH: *Aloha Editor - HTML5 WYSIWYG Editor* [online]. Dostupné z: http://aloha-editor.org/, [cit. 2012-04-11].

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

[7] Hruška, T.: *Internetové aplikace (WAP) VI. část Programování klienta (JavaScript) (Studijní opora)*. Brno, 2007.

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

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

[10] JavaScript Kit: *Cut & Paste Validate Date field script* [online]. Dostupné z: http://www.javascriptkit.com/script/script2/validatedate.shtml, [cit. 2012-04-18].

[11] Kleban, M.: *Editor anotací pro WYSIWYG textové editory napsané v jazyce JavaScript, diplomová práce*. Brno, FIT VUT, 2011.

[12] Kolektiv autorů: *Hypertext Transfer Protocol – HTTP/1.1 (RFC2616)* [online]. Dostupné z: http://tools.ietf.org/html/rfc2616, Poslední modifikace: červen 1999 [cit. 2012-04-07].

[13] Lie, H. W.: *Håkon Wium Lie* [online]. Dostupné z: http://people.opera.com/howcome/, [cit. 2012-04-08].

[14] Moxiecode Systems AB: *TinyMCE - Javascript WYSIWYG Editor* [online]. Dostupné z: http://www.tinymce.com/, [cit. 2012-04-11].

[15] Mozilla: *Firebug - Web Development Evolved* [online]. Dostupné z: https://getfirebug.com/, Poslední modifikace: 2012-04-13 [cit. 2012-04-21].

[16] Network, M. D.: *JavaScript Language Resources* [online]. Dostupné z: https://developer.mozilla.org/en/JavaScript_Language_Resources, Poslední modifikace: 2011-09-16 [cit. 2012-04-09].

[17] Network Working Group: *RFC 3339 - Date and Time on the Internet: Timestamps* [online]. Dostupné z: http://www.ietf.org/rfc/rfc3339.txt, Poslední modifikace: 2002-07-01 [cit. 2012-04-30].

[18] Odvárko, J.: *JSColor - JavaScript / HTML Color Picker* [online]. Dostupné z: http://jscolor.com/, Poslední modifikace: 2012-01-10 [cit. 2012-04-19].

[19] Russell, A.: *Comet: Low Latency Data for the Browser* [online]. Dostupné z: http://infrequently.org/2006/03/comet-low-latency-data-for-the-browser/, Poslední modifikace: 2006-03-03 [cit. 2012-04-09].

[20] Smrž, P., Dytrych, J.: *Towards New Scholarly Communication: A Case Study of the 4A Framework*. Brno University of Technology, Faculty of Information Technology.

[21] Stackoverflow: *endsWith in javascript* [online]. Dostupné z: http://stackoverflow.com/questions/280634/endswith-in-javascript, [cit. 2012-04-18].

[22] Stackoverflow: *How can I convert a string to boolean in JavaScript?* [online]. Dostupné z: http://stackoverflow.com/questions/263965/how-can-i-convert-a-string-to-boolean-in-javascript, [cit. 2012-04-18].

[23] Stackoverflow: *Integer division in JavaScript* [online]. Dostupné z: http://stackoverflow.com/questions/4228356/integer-division-in-javascript, [cit. 2012-04-19].

[24] Stackoverflow: *Assign click handler in for loop in javascript* [online]. Dostupné z: http://stackoverflow.com/questions/8881306/assign-click-handler-in-for-loop-in-javascript, [cit. 2012-04-20].

[25] Steigerwald, D.: *Třídy, dědičnost a OOP v Javascriptu - I* [online]. Dostupné z: http://zdrojak.root.cz/clanky/oop-v-javascriptu-i/, Poslední modifikace: 2010-03-15 [cit. 2012-04-09].

[26] The Tech Terms Computer Dictionary: *WYSIWYG* [online]. Dostupné z: http://www.techterms.com/definition/wysiwyg, Poslední modifikace: 2008-06-11 [cit. 2012-04-11].

[27] W3C: *What is HyperText* [online]. Dostupné z: http://www.w3.org/WhatIs.html, [cit. 2012-04-07].

[28] W3C: *HTML Current Status* [online]. Dostupné z: http://www.w3.org/standards/techs/html#w3c_all, [cit. 2012-04-08].

[29] W3C: *HTML & CSS* [online]. Dostupné z: http://www.w3.org/standards/webdesign/htmlcss, [cit. 2012-04-08].

[30] W3C: *XML Path Language (XPath) 2.0 (Second Edition)* [online]. Dostupné z: http://www.w3.org/TR/xpath20/, Poslední modifikace: 2011-01-03 [cit. 2012-04-08].

[31] W3C: *XQuery 1.0: An XML Query Language (Second Edition)* [online]. Dostupné z: http://www.w3.org/Style/XSL/, Poslední modifikace: 2011-01-03 [cit. 2012-04-13].

[32] W3C: *The Extensible Stylesheet Language Family (XSL)* [online]. Dostupné z: http://www.w3.org/Style/XSL/, Poslední modifikace: 2011-05-05 [cit. 2012-04-13].

[33] W3C: *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification* [online]. Dostupné z: http://www.w3.org/TR/CSS21/, Poslední modifikace: 2011-06-07 [cit. 2012-04-08].

[34] W3C: *HTTP - Hypertext Transfer Protocol* [online]. Dostupné z: http://www.w3.org/Protocols/, Poslední modifikace: 2012-01-04 [cit. 2012-04-07].

[35] W3C: *Extensible Markup Language (XML)* [online]. Dostupné z: http://www.w3.org/XML/, Poslední modifikace: 2012-01-24 [cit. 2012-04-07].

[36] W3C Schools: *JavaScript isFinite() Function* [online]. Dostupné z: http://www.w3schools.com/jsref/jsref_isfinite.asp, [cit. 2012-04-18].

[37] W3C Schools: *JavaScript String Object* [online]. Dostupné z: http://www.w3schools.com/jsref/jsref_obj_string.asp, [cit. 2012-04-18].

[38] wikiHow: *How to Convert Hexadecimal to Binary or Decimal* [online]. Dostupné z: http://www.wikihow.com/Convert-Hexadecimal-to-Binary-or-Decimal, Poslední modifikace: 2012-02-19 [cit. 2012-04-19].

[39] wikiHow: *How to Convert from Decimal to Hexadecimal* [online]. Dostupné z: http://www.wikihow.com/Convert-from-Decimal-to-Hexadecimal, Poslední modifikace: 2012-02-28 [cit. 2012-04-19].

[40] Zakas, N. C.: *JavaScript pro webové vývojáře*. Computer Press, Brno, 2009, ISBN: 978-80-251-2509-0.

[41] Zakhour, S., Hommel, S., Royal, J., Rabinovitch, I., Risser, T., Hoeber, M.: *Java 6 - Výukový kurz*. Computer Press, 2007, ISBN: 978-80-251-1575-6.

Přílohy

Seznam příloh

Příloha A

Obsah přiloženého DVD

Na přiloženém DVD lze nalézt:

- zdrojové kódy vylepšeného editoru anotací
- zdrojové kódy textu bakalářské práce
- text bakalářské práce ve formátu PDF
- inicializační skript databáze serveru
- zdrojové kódy serveru a manuál k jeho instalaci