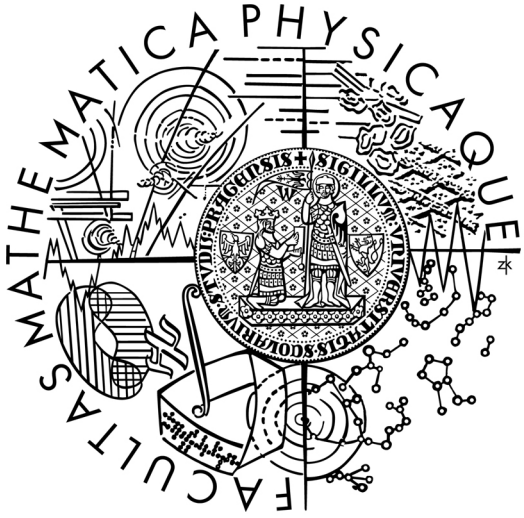
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Jakub Valčík

Editor XML dat

Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. Irena Mlýnková

Studijní program: Informatika, programování

2007

Děkuji RNDr. Ireně Mlýnkové za odborné vedení mé práce, za rady a za čas, který mi během vypracovávání této práce věnovala.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 9.8.2007 Jakub Valčík

Obsah

Název práce: *Editor XML dat*

Autor: *Jakub Valčík*

Katedra (ústav): *Katedra softwarového inženýrství*

Vedoucí diplomové práce: *RNDr. Irena Mlýnková*

e-mail vedoucího: *irena.mlynkova@mff.cuni.cz*

Abstrakt: *Cílem práce je implementace editoru určeného speciálně pro vytváření a úpravu XML dokumentů, DTD a schémat v jazyce XML Schema. Jádrem práce je pak návrh a implementace vlastního editoru. Výsledný program zahrnuje textový editor s podporou highlightingu a kontextových menu s přehledem možných prvků na dané úrovni. Pro automatizované generování XML dokumentu je v programu vytvořen generický editor schopný vytvářet editační okna uzpůsobená pro vytváření elementů, dodržujících strukturu danou schématem. Program umožňuje kontrolu správné strukturovanosti a validity XML dokumentů. Práce obsahuje také teoretická východiska a porovnání s několika podobnými programy.*

Klíčová slova: *XML, editor, XML Schema, DTD*

Title: *XML data editor*

Author: *Jakub Valčík*

Department: *Department of Software Engineering*

Supervisor: *RNDr. Irena Mlýnková*

Supervisor's e-mail address: *irena.mlynkova@mff.cuni.cz*

Abstract: *The aim of the work is to implement an editor especially designated to create and edit XML documents, DTDs and schemas in language XML Schema. The core of the work is design and implementation of the editor. The resulting application includes text editor with support of highlighting and context menus with list of elements and attributes allowed on requested level. Application enables automatically generate XML documents thanks to a generic editor with the ability to create edit forms designed for creating elements in accordance with the structure specified by a schema. It also allows checking XML documents for correct structure and validity according its schema. The work also contains theoretical principles and a comparison with several similar programs*

Keywords: *XML, editor, XML Schema, DTD*

1 Úvod

Nároky na zpracování textu rostou. Roste potřeba jednotlivé texty organizovat do skupin, řadit podle zadaných kritérií, vyhledávat klíčová slova, přenášet texty po celém světě ať už na disketě, CD, pomocí internetu nebo jiného přenosového média, zobrazovat pro čtenáře v co nejlepším možném formátu. Tímto rozhodně nekončí výpis možností jak zpracovávat text. Aby se mohly lépe plnit požadavky stále náročnějšího uživatele, jsou do textu přidány určité řídící znaky. Pomocí nich lze efektivněji zpracovávat daný text. Nejznámějšími řídícími znaky jsou například odřádkování nebo tabulátor. Tyto řídící znaky slouží uživateli pro konečné zobrazení. Do textu se mohou přidat i řídící znaky pro následné řazení či vyhledávání. Řídící znaky mohou určit, co je nadpis v textu a co jsou klíčová slova. Při zpracování takového to textu může být uživateli vygenerována například osnova nebo seznam použitých klíčových slov.

Řídící znaky při běžném zobrazení většinou nejsou vidět. Aby uživatelé mohli pročítat text i s řídícími znaky, mohou být v textu nahrazeny řídícími textovými značkami. Aby se rozeznaly řídící texty od základního textu, používají se různé techniky. Například formát RTF (Rich-Text Format) [1] používá k zadání řídící značky zpětná lomítka a složené závorky. Pro ukončení řídícího znaku je použit oddělovač, kterým je například mezera. Jazyk XML (eXtensible Markup Language) [2] je značkovací jazyk. Do textu jsou přidány řídící značky – XML tagy. Při tvorbě dokumentu ve formátu RTF se musejí používat značky, které určuje specifikace. XML určuje pouze způsob zápisu značek, jejich názvy a význam si definuje uživatel sám. XML je tedy velice flexibilní a v dnešní době stále více populární formát.

1.1 Proč editovat XML

Aby si uživatel mohl vytvořit například vlastní webovou stránku, napíše v jednom z jazyků pro tvorbu internetových stránek kód, který je posléze nahrán na server, a tím zveřejněn. Tímto jazykem může být XHTML (eXtensible HyperText Markup Language) [13]. XHTML je XML, což je textový formát, který může být editován například v NOTEPADu jako běžný text. Ale jelikož to je XML, mohou být využity

sofistikovanější produkty zaměřené na editaci právě XML. Tyto aplikace například zvýrazní a barvami odliší různé části dokumentu pro lepší a snažší orientaci uživatele. Nabízejí možné elementy a atributy, které lze použít na dané úrovni. Automaticky mohou generovat části kódu či uzavírat elementy. Další potřeba editace je pro konfigurační soubory, které musí být editovány pro změnu nastavení aplikací. V počítačovém světě je stále více a více informací „zabalených“ do elementů a atributů ve formátu XML a roste potřeba nejen si tyto informace prohlížet, ať už zpracované nějakým XML procesorem s použitím šablony pro zobrazení, tak v čisté textové podobě, ale také je editovat a měnit.

1.2 Cíl této práce

Cílem této práce je vytvořit nástroj pro prohlížení a editaci XML dokumentů. Jelikož jsou data v XML uzavřena do řídících značek, lze je využít ve prospěch editoru. Editor tak může ulehčit uživateli vytváření dokumentů. Výsledná aplikace ponese název xmlPad.

K XML dokumentu lze připojit schéma, které určuje strukturu řídících značek. Při editaci dokumentu editor načte schéma k dokumentu připojené. Ze znalosti struktury editovaného XML dokumentu určí editor značky, které se mohou vyskytnout na dané úrovni dokumentu a nabídne je uživateli. Díky této nápovědě značek může editor při vytváření dokumentu vést uživatele, který nemusí znát přesnou strukturu. XML značky jsou více druhů a proto je editor bude barevně rozlišovat. Pro uživatele tím bude XML dokument daleko přehlednější a bude se v něm lépe orientovat.

Editor xmlPad obsahuje generický editor. Ten funguje jako průvodce, který provede uživatele vytvářením XML dokumentu. Bude opět využívat nadefinované struktury schématu, které je připojeno k editovanému dokumentu. Generický editor nabízí uživateli značky, které bude automaticky vkládat do textu. Celou strukturu dokumentu tedy vytvoří editor sám, uživatel bude pouze zapisovat data, která editor vloží do výsledného dokumentu mezi vytvořené značky.

Veliká data způsobují řadu problémů. Aplikace, která se neumí vypořádat s velkými dokumenty může mít problém s nedostatkem paměti, velice pomalými reakcemi při

jakýchkoli operacích nebo ani nemusí zvládnout velký dokument otevřít. Odstranit tyto následky práce s velkými soubory je složité. Vyžaduje upravit přístup k editovaným datům, tak aby s nimi editor zvládl pracovat a čas strávený nad prováděním operací byl pro uživatele přijatelný. Při editování rozsáhlých XML dokumentů jsou XML značky spíše přítěží než jak tomu bylo v předchozích odstavcích. Kvůli struktuře XML dokumentu navíc nelze libovolně rozdělit XML dokument na části a vždy editovat jen danou část. Rozdělení do částí musí splňovat určité předpoklady, aby nedošlo k porušení či úplné ztrátě informace, kterou XML dokument nese. Aplikace xmlPad efektivně zpracovává rozumně velké dokumenty a nabízí uživateli editaci takto rozsáhlých XML dokumentů s přijatelnými časovými prodlevami.

1.3 Struktura textu

V této bakalářské práci je ve druhé kapitole popsán formát XML a dva druhy jazyků pro popis schémat, a to XML Schema [4] a DTD (Document Type Definition) [5]. Popis čtyř přístupů k XML dokumentu z pohledu programátora nalezne čtenář ve třetí kapitole. Mezi druhy přístupů k XML zde popsaných je jeden, který byl vyvinut právě pro aplikaci xmlPad za účelem editace velkých souborů. Ve čtvrté kapitole je soupis požadavků na cílovou aplikaci. Pátá kapitola popisuje práci uživatele s aplikací a vysvětluje její funkce na rámcovém příkladu. Další, šestá kapitola popisuje vývoj aplikace a problémy, které vznikly při jejím vývoji. Předposlední, sedmá kapitola analyzuje a porovnává již vytvořené veřejně dostupné nástroje, pomocí kterých lze editovat XML dokument. Závěr shrnuje výslednou aplikaci, dosažení vytyčených cílů a uvádí možnosti rozšíření.

2 XML podrobněji

V této kapitole se čtenář seznámí s formátem XML a se dvěma druhy jazyků pro popis schémat, které se používají pro definování struktury XML dokumentů. Také se dozví v jakých případech se XML využívá.

2.1 XML dokument

Správně strukturovaný [6] XML dokument má na začátku XML deklaraci, ve které je uvedeno o jakou verzi XML se jedná a jaké je použito kódování. Definice kódování může chybět, a v tom případě je použito kódování ISO 10646 [7]. Pro komunikaci se světem se obvykle používá UTF-8 [8] (kompatibilní s ASCII [9], dalším použitelným kódováním). Pro češtinu lze použít ISO-8859-2 [10] nebo Windows-1250 [11]. Po deklaraci je možné nadefinovat strukturu XML dokumentu, o kterou se zajímá další část této kapitoly. Nyní již zbývá samotný obsah, který následuje. Systém XML značek je založen na otevírací značce neboli elementu a na zavíracím elementu. Otevírací značka je tvořena levou špičatou závorkou, názvem elementu, případnými atributy a ukončena pravou špičatou závorkou. Zavírací element vypadá jako otvírací bez atributů, jen je mezi levou špičatou závorkou a názvem elementu lomítko. Atributy elementu jsou vždy jen v otevírací značce. Atribut elementu je tvořen názvem a hodnotou. V každém elementu musí být jména atributů unikátní, nelze tedy v jednom elementu mít více atributů se stejným názvem. V XML je atribut zapsán jako název atributu následovaný rovnítkem a hodnotou atributu v uvozovkách. Mezi otevíracím a uzavíracím elementem jsou buď další elementy, nebo text (obr. 2.1).

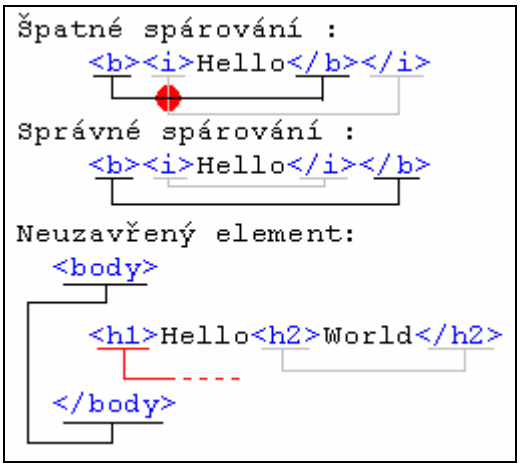
Mezi další řídící značky XML patří komentář, do kterého může být uzavřen libovolný text i značky. Jediným omezením v obsahu komentáře je zákaz výskytu dvou pomlček za sebou. XML procesor při své analýze pozná komentovaná místa, přeskočí je a nijak nezpracovává. V XML dokumentu se mohou vyskytnout i instrukce pro nějakou aplikaci. Tyto instrukce jsou uvnitř značky, které se říká instrukce pro zpracování. Jako poslední je sekce CDATA. Tato sekce slouží pro vložení jakéhokoliv textu do těla nějakého elementu. Aby mohl XML procesor rozpoznat začátky elementů, je zakázáno používat v běžném textu otevírací špičatou

závorku. Místo ní se použije náhrada ve tvaru „<“, této náhradě se říká entita. Při zpracování XML procesor vyhledává entity a nahrazuje je. Při zobrazení se tedy entita „<“ nahradí znakem „<“. Entity mají tvar ampersand, jméno entity a jsou ukončeny středníkem. Ampersand má tedy v xml speciální význam, proto když ho bude chtít uživatel napsat do svého XML dokumentu musí ho opsat také entitou, která je pro ampersand „&”. Aby v textu obsahujícím velké množství těchto znaků nemusel uživatel vše nahrazovat příslušnými entitami, může využít sekci CDATA. XML procesor při zpracovávání dokumentu v sekci CDATA nepovažuje žádný znak za speciální a uživatel může používat ampersand nebo levou špičatou závorku jak se mu zlíbí. V těle sekce CDATA je pouze jedno omezení, nesmí se zde vyskytnout řetězec „]]“. Tento řetězec totiž sekci CDATA ukončuje.

```
<?xml version="1.0" encoding="utf-8"?>
<element atribut="hodnota atributu">
  <!-- toto je komentář <element> -->
  <element bez="obsahu"/>
  <sekce cdata="true">
    <![CDATA[ zde si můžu psát co chci,
       dokonce i < > nebo & ]]>
  </sekce>
</element>
```

Obrázek 2.1: Ukázka XML dokumentu

Celý obsah správně zformovaného dokumentu musí být uzavřen do jednoho elementu, kterému se říká kořenový. Uvnitř tohoto elementu se dále vyskytují další elementy, komentáře, texty, instrukce pro zpracování a sekce CDATA. XML rozlišuje velká a malá písmena, takže záleží na jejich psaní v názvech otevíracích a uzavíracích elementů. Všechny elementy musejí být správně spárovány a musejí být všechny uzavřeny (obr. 2.2). Například při tvorbě webových stránek má uživatel na výběr jaké normě bude výsledný kód odpovídat. Může si vybrat normu HTML (HyperText Markup Languge) [12] nebo normu XHTML. Norma XHTML je založena na formátu XML a výsledná webová stránka tak musí splňovat správnou strukturu XML dokumentu. Norma HTML není odvozena od XML a uživatel při použití této normy nemusí například uzavírat elementy.



Obrázek 2.2: Párování tagů

2.2 Schéma pro XML

Pomocí schémat je možné určovat strukturu XML dokumentu. Ve schématu se nadefinují elementy a atributy, které jsou pak použity v XML dokumentu. Podle vyjadřovací schopnosti jazyka pro popis schémat je možné nadefinovat různě složité vazby. Obecně lze nadefinovat název elementu a jaké elementy má v sobě uzavírat nebo má-li mít textový popřípadě smíšený obsah. Dále uživatel může například určit jaké atributy musí element obsahovat a případně nadefinovat určité restrikce na hodnoty atributů.

2.2.1 DTD

Asi nejznámější jazyk pro popis schématu je DTD. Má odlišnou strukturu od XML, proto pro jeho analyzování musí být použity speciální procesory. V DTD se nejprve určí kořenový element, který musí být pouze jeden. Dále se nadefinují elementy, které tento element uzavírá v sobě (potomci elementu).

Pro definování přípustného výskytu potomků elementu jsou v DTD používány operátory plus, hvězdička, svislá čára, otazník a čárka. Díky těmto operátorům se nadefinuje struktura potomků všech elementů použitých v dokumentu. Lze vyjádřit pouze vztah rodič – potomek a nelze nadefinovat dva elementy se stejným názvem, ale jiným obsahem. Čárka mezi elementy značí sekvenci, tzn. elementy musejí být v napsaném pořadí za sebou. Plus, hvězdička a otazník slouží k určení počtu elementů. Otazník znamená žádný nebo jeden element. Plus znamená alespoň jeden

element, ale může jich být více. Hvězdička značí libovolný počet elementů. Svislá čára určuje volbu elementu, uživatel si může při tvorbě XML dokumentu vybrat mezi elementy, které tato svislá čára odděluje.

Ke každému elementu mohou být v DTD nadefinovány atributy, jejich povinost výskytu a jejich datový typ. Při definování atributu se nejprve napíše, ke kterému elementu atribut patří, poté jeho název a případně povinnost výskytu. V definici atributu lze zapsat zda je striktně vyžadován pomocí slova #REQUIRED. Uživatel také může nadefinovat obsah hodnoty elementu, tedy pomocí výčtu hodnot se uvede jaké hodnoty jsou přípustné pro daný atribut. Také lze nadefinovat implicitní hodnotu atributu. Na obr. 2.3 vidíme ukázku DTD. Uvedené schéma říká, že XML dokument musí obsahovat element zaměstnanci. Tento element má jediný druh potomka, kterým je osoba. Plus u elementu osoba znamená, že se může libovolněkrát opakovat, ale musí být v XML dokumentu alespoň jednou. Podobně je definován i obsah elementu osoba. Dále je v ukázce příklad definice atributů elementu osoba. Celkem může mít element osoba až tři atributy a to id, poznámka a dovolená. Atribut id je povinný a jeho obsah musí být jedinečný identifikátor v rámci dokumentu. Atribut poznámka nemá implicitní hodnotu a jeho obsah je řetězec. Poslední atribut je dovolená, jeho hodnota je omezena výčtem na hodnoty ano a ne, implicitně má tento atribut hodnotu ne. Dále jsou zde určeny datové typy zbývajících elementů jméno, email a telefon. Datový typ je zde libovolný text.

```
<!ELEMENT zaměstnanci (osoba)+>
<!ELEMENT osoba (jméno, email*, telefon?)>
    <!ATTLIST osoba id ID #REQUIRED>
    <!ATTLIST osoba poznámka CDATA #IMPLIED>
    <!ATTLIST osoba dovolená (ano|ne) "ne">
<!ELEMENT jméno (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT telefon (#PCDATA)>
```

Obrázek 2.3: Ukázka DTD

2.2.2 XML Schema

Druhým jazykem popisujícím schéma pro XML dokumenty, o které se bude tato práce zajímat, je XML Schema. S rostoucími požadavky na přesnost struktury XML dokumentu přestává stačit vyjadřovací schopnost DTD. Proto vzniká nový jazyk pro

popis struktury XML dokumentu XML Schema. Jazyk XML Schema využívá XML jako formát pro zápis popisovaného schématu. Narozdíl od DTD tedy není potřeba konstruovat speciální analyzátory pro další jazyk. XML Schema má silnou podpora datových typů. A jak je uvedeno ve studijních materiálech [22] lze vyjádřit přesný počet elementů v rámci nadelementu. Uživatel může nadefinovat elementy se stejnými názvy, ale různými obsahy. Také může snadno vyjádřit libovolné pořadí elementů v rámci nadelementu a při modelování je možné opakovaně využívat již definované prvky. XML Schema využívá objektově-orientovaného přístupu a v tomto jazyku může být využito dědičnosti, substituovatelnosti apod., což je pro modelování přirozené. Unikátnost obsahu elementu, hodnoty atributu nebo jejich kombinace lze specifikovat nejen v rámci celého dokumentu, ale také lze oblast omezit pouze na požadovanou část.

Na obr. 2.4. je ukázka schématu v jazyce XML Schema, kde je vidět nadefinovaný element komentar, který má jako svůj obsah libovolný text. Jako další je nadefinovaný typ ObjednavkovyTyp. Tento typ se dále použije jako typ nějakého uživatelem definovaného elementu. Pokud tedy uživatel přiřadí elementu tento typ, znamená to že element v XML dokumentu bude mít jako potomky elementy dorucovaci-adresa, fakturacni-adresa a komentar. V definici elementů dorucovaci-adresa a fakturacni-adresa je vidět, že jsou typu USAddress, tento typ není součástí ukázky. Třetí element je nadefinovaný jako odkaz na globálně definovaný element s názvem komentar a atribut minOccurs říká, že se tento element nemusí v dokumentu objevit. Všechny tři definované elementy jsou uzavřeny v elementu sequence, což značí neměnnou posloupnost definovaných elementů. Sekvence v jazyce XML Schema odpovídá čárce v DTD, stejně tak choice odpovídá svislé čáře. Speciální operátor all nemá svůj protějšek v DTD. V schématu v XML Schema určuje all neuspořádanou sekvenci nebo-li libovolné pořadí elementů. U typu ObjednavkovyTyp je také nadefinován atribut se jménem datum-objednani. Datový typ tohoto atributu je nadefinován jako datum.

```
<xsd:element name="komentar" type="xsd:string"/>
<xsd:complexType name="ObjednavkovyTyp">
  <xsd:sequence>
   <xsd:element name="dorucovaci-adresa" type="USAddress"/>
   <xsd:element name="fakturacni-adresa" type="USAddress"/>
   <xsd:element ref="komentar" minOccurs="0"/>
 </xsd:sequence>
 <xsd:attribute name="datum-objednani" type="xsd:date"/>
</xsd:complexType>
```

Obrázek 2.4: Ukázka XML Schema

2.3 Jmenné prostory

Při tvorbě XML dokumentu, který nese jistý druh informace, si uživatel vytvoří vlastní sadu značek. Uživatel si určí jména jednotlivých elementů a jejich atributů a vyjádří tuto skutečnost pomocí schématu. Jména jsou v této specifikaci jednoznačná. Problém nastává, když bude chtít uživatel použít v jednom dokumentu více sad značek. Může například dojít ke konfliktu jmen elementů. Tento problém řeší jmenné prostory [39].

Jak je uvedeno v prezentaci k přednášce Technologie XML [22], prostory jmen fungují na jednoduchém principu. Každý element a atribut může být přiřazen k prostoru jmen, který je identifikován jednoznačným textovým řetězcem splňujícím určitá pravidla. Pro zkrácení zápisu se pak v XML dokumentu deklarují pro zavedené jmenné prostory krátké prefixy, které se používají pro přiřazení elementu nebo atributu do určitého jmenného prostoru. Například při použití jazyka XML Schema se uživatel nevyhne použití jmenných prostorů. V příkladu na obrázku 2.5 je vidět deklarace prefixu xsd pro jmenný prostor http://www.w3.org/2001/XMLSchema používaný pro značky jazyka XML Schema, tj. všechny značky s prefixem XSD patří do tohoto jmenného prostoru. Na obrázku je také vidět použití prefixu xml, tento prefix je použit pro jmenný prostor jazyka XML http://www.w3.org/XML/1998/namespace, který je automaticky připojen ke každému XML dokumentu. Při deklarování jmenných prostorů se tedy nesmějí používat prefixy xml a xmlns.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

 <xsd:annotation>
  <xsd:documentation xml:lang="cs">
   Schéma objednávky pro Example.com.
   Copyright 2000 Example.com. Všechna práva vyhrazena.
  </xsd:documentation>
 </xsd:annotation>

...

</xsd:schema>
```

Obrázek 2.5 : Ukázka deklarace jmenného prostoru

2.4 Kde a proč se používá XML

Každý, kdo někdy použil internet a prohlédnul si nějakou stránku, se dostal do kontaktu s XML, i když zpracovaným XML procesorem a zobrazeným na obrazovku. Tímto XML procesorem byl v tomto případě webový prohlížeč uživatele. Díky textovému formátu XML lze k jeho přenosu používat protokol HTTP (HyperText Transfer Protocol) [3]. Webové stránky jsou většinou ve formátu HTML. Jak již bylo napsáno HTML není XML. K tvorbě webových stránek se tudíž čím dál více začíná používat norma XHTML. Tato norma vznikla úpravou normy HTML tak aby výsledná webová stránka byla XML dokument. Tedy webová stránka podle normy XHTML musí splňovat definici správně strukturovaného XML dokumentu a struktura této stránky musí odpovídat schématu podle specifikace normy.

Zobrazování XML je založeno na oddělení dat, která jsou v XML dokumentu, od způsobu formátování, který se může měnit a je uložen v dalších souborech. Tedy při zobrazení XML procesor načte data z XML dokumentu a podle zadaného formátování tyto data vykreslí na obrazovku. Toto je velice užitečné pro prohlížení dat na různých zařízeních, pro různá zařízení jiný druh formátování. Díky této variabilitě můžeme plně využít možnosti zařízení, na kterém jsou data zobrazována. Pro transformaci XML dat do uživateli příjemné podoby se používá XSL (eXtensible Stylesheet Language) [15]. Tento formát je založen na XML.

XML se také používá, v dnes tak populárních B2B (Business To Business) aplikacích při výměně dat či volání vzdálených metod. Příkladem této komunikace

jsou webové služby (Web Services) [16], které používají protokol SOAP (Simple Object Access Protocol) [17]. Služba funguje na principu přijmutí požadavku a odeslání odpovědi. Požadavkem je jméno volané metody se zadanými parametry. Server pak spustí zadanou metodu s parametry z požadavku a výsledek vrátí klientovi. Jak požadavek, tak odpověď jsou ve formátu XML, tudíž je umí dobře zpracovat jak počítač, tak je může přečíst i člověk. S webovými službami také souvisí zkratka WSDL (Web Services Description Language) [18]. Pomocí tohoto nástroje je nadefinován seznam metod s názvy, pořadím a typy parametrů, které mohou být na daném serveru volány klientem. WSDL je vlastně konfiguračním souborem pro klienty webové služby, podle kterého se mohou automaticky nastavit a nabídnout uživateli všechny metody, co tato konfigurace obsahuje i s názvy a typy parametrů. WSDL je samozřejmě také XML.

XML lze použít pro konfiguraci nejen klientů webových služeb. Nyní již velká část softwarových produktů používá XML jako své konfigurační soubory, protože oproti ini souborům ve stylu název proměnné, rovnítko a hodnota proměnné je XML daleko více škálovatelné.

XML se také používá pro logování průběhu nějakých událostí. To již není tak časté neboť je zde jedna nevýhoda oproti ukládání logu do čistého textu. Pokud se jedná o textový log, pak se při zápisu pouze otevře soubor a přidá se nakonec požadovaný záznam. U XML nelze zapsat nakonec, protože, jak bylo zmíněno výše, musí být všechny elementy uvnitř kořenového elementu a musí být dodržena struktura celého dokumentu. Tedy při ukládání záznamu se například musí celý XML dokument načíst do paměti, což je náročné na paměť při velkých souborech (což většinou při logování bývají). Po načtení do paměti se přidá uzel záznamu do dokumentu a poté se tento dokument uloží na pevný disk.

XML tedy slouží pro přenos informací a jejich následným zpracováním aplikací a pro konfigurační a logovací soubory. Další oblastí využití XML je pro uchovávání dat, kde XML může sloužit přímo jako databáze. Jako byl pro relační databáze vytvořen jazyk SQL (Structured Query Language) [19], tak pro XML byly vytvořeny jazyky XQuery [20], XPath [21] a mnoho dalších. Pomocí těchto jazyků se lze dotazovat nad XML dokumentem a dobývat data. Nové databázové servery jako je

MS SQL 2005 mají nativní podporu pro XML data, takže je možné i v SQL dotazu přímo používat již zmiňované jazyky pro práci s XML daty.

3 Jak pracovat s XML

Z předchozích kapitol už víme jakou strukturu má XML dokument a kde se používá. V této kapitole budou rozebrány způsoby, jak lze s XML pracovat a také jaké se používají datové modely pro uchování XML v paměti.

3.1 Reader (pull model)

S XML se vyvinuly i způsoby jak s tímto formátem pracovat. Jako první zmíníme ten, který je použit i v aplikaci xmlPad. Je to XML reader. Reader umožňuje procházet obsah dokumentu, a to uzel po uzlu, nikoli bajt po bajtu nebo záznam po záznamu, jak píše D. Esposito v XML efektivní programování pro .NET [23]. Pro readery přestává být XML dokument textovým souborem doplněným značkami a stává se kolekcí uzlů. Uzel je v podání readeru jakýkoliv vyjadřovací prvek ve struktuře XML dokumentu, tedy uzlem je například element, komentář ale i text uvnitř elementu. Tento model přístupu je specifickým rysem platformy .NET [24]. Readery XML nevyžadují, aby bylo do paměti ukládáno více dat než je zapotřebí. Když je otevřen XML dokument, je vrácen jednoduchý logický ukazatel na uzel. Pak lze jednoduše postupovat po uzlech a vyhledat ten, se kterým chceme pracovat. Díky tomu není paměť aplikace zatěžována žádnými nadbytečnými daty, do paměti je načten pouze obsah aktuálního uzlu.

Pro inicializaci XML readeru je třeba předat nastavovací struktury a datový proud s XML daty. Nastavovacími strukturami se readeru řekne jak má zpracovávat XML, například jestli má zpracovávat nebo ignorovat nevýznamné bílé znaky, komentáře, či instrukce pro zpracování. Jakmile je XML reader připravený, začíná běhová smyčka, ve které se posouvá reader pouze směrem kupředu pomocí volání metody Read. Reader se posouvá ne po bajtech, ale po uzlech jak bylo zmíněno výše. Po každém volání metody Read je v readeru načten aktuální zpracovávaný uzel. Programátor se tedy z readeru pomocí „properties“ dozví jaký typ uzlu je právě zpracováván, může získat prefix, lokální jméno nebo jmenný prostor uzlu. Také může získat aktuálně nadeklarované jmenné prostory vzhledem ke zpracovávanému uzlu. Programátor se dále z readeru dozví jestli má element nějaké atributy a pokud ano, pak je může zase dále zpracovat. Po dalším volání metody Read je do readeru

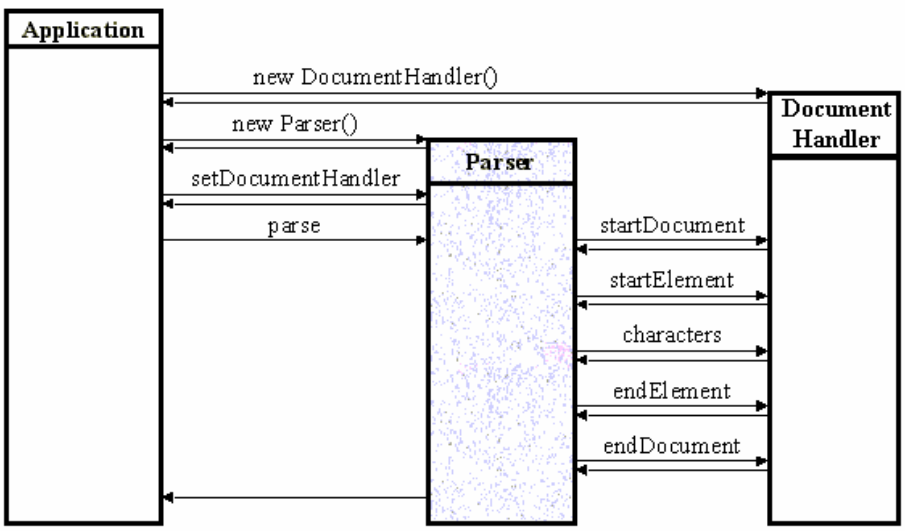
načten nový aktuální uzel. Reader si tedy nepamatuje téměř žádné informace ze zpracování minulých uzlů, díky tomu není zabírána paměť aplikace. Je-li potřeba najít určitý element musí se v nejhorším případě projít celý XML dokument.

3.2 Parser (push model)

Parser SAX (Simple API for XML parsing) [25] řídí přímo průběh procesu čtení dokumentu a odesílá data klientské aplikaci. Proti tomu je XML Reader mnohem pasivnější a ponechává řízení celého procesu na klientské aplikaci.

Pokud chce programátor využít SAX parser pro čtení XML dokumentů, musí nejprve vytvořit třídu, která má rozhraní, se kterým parser umí komunikovat. Tato třída se při inicializaci předá parseru. Pro začátek čtení XML dokumentu parserem slouží metoda parse. Parser postupně čte dokument a vždy, když narazí na nějaký uzel, zavolá příslušnou metodu v programátorem definované třídě, která byla u parseru zaregistrována při inicializaci. Pokud chce parser sdělit zaregistrované třídě něco víc, než jen že narazil na uzel, tak to udělá pomocí parametrů volané metody, které naplní dodatečnými informacemi. Programátor si nadefinuje obsah metod, které reagují na požadované uzly a tím může zpracovat celý dokument. Například pro zpracování začátečního elementu musí být nadefinována metoda startElement. V této metodě jsou parametry určující jmenný prostor, ve kterém se nachází zpracovávaný element, jméno elementu, jméno elementu s prefixem a atributy elementu. Pokud je potřeba najít nějaký uzel uvnitř XML dokumentu, musí se jako u XML readeru v nejhorším případě také projít celý strom.

Na obrázku 3.1 je vidět způsob spolupráce parseru a aplikace při čtení XML dokumentu. V aplikaci se vytvoří třída Document Handler, která má za úkol zpracovávat uzly dokumentu. Dále se vytvoří SAX Parser, ve kterém se zaregistruje třída Document Handler. Čtení se započne předáním řízení aplikace parseru. Parser čte XML dokument a při tom volá metody Document Handleru, které mají v sobě kód pro zpracování uzlů. Po přečtení dokumentu je tok řízení opět vrácen aplikaci.

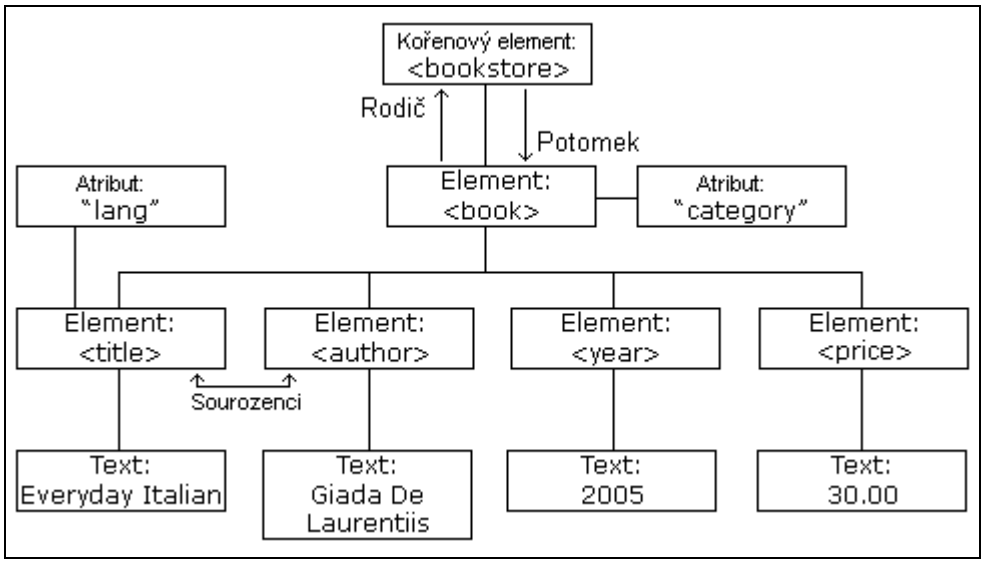


Obrázek 3.1 : Schéma SAX parseru

3.3 XML DOM

XML DOM (Document Object Model) [40] je standardizované rozhraní pro čtení i zápis dat. DOM reprezentuje XML dokument stromovou strukturou – stromem uzlů. Ke každému elementu, jeho atributům i k jeho textovému obsahu lze přistoupit jako k prvkům stromu. Obsah uzlů stromu lze editovat a mazat. Nové uzly mohou být vytvořeny a přidány do struktury.

Všechny uzly mají mezi sebou nějaký vztah. Například na obrázku 3.2 je vidět stromová struktura fragmentu XML dokumentu. Jsou zde celkem čtyři druhy uzlů, element, kořenový element, atribut a text. Mezi kořenovým elementem bookstore a na obrázku hned pod ním elementem book je vidět jeden ze vztahů – vztah rodič – potomek, který je označen šipkami mezi těmito uzly s popisky Rodič a Potomek. Další vztah je typu sourozenci, který je vidět mezi elementy title a autor a je označen šipkou s popiskem Sourozenci. Element book má dále atribut category a celkem osm potomků, kterými jsou title, autor, year a price. Všechny tyto elementy mají textový obsah s informacemi, které jsou vidět na obrázku. Element title má navíc atribut s názvem lang.



Obrázek 3.2 : Stromová struktura XML DOM

XML DOM má jednu značnou nevýhodu. Pro vyhledání jediného uzlu nebo pro změnu jedné vlastnosti je nutné načíst do operační paměti počítače celý dokument. To může spotřebovat značné množství paměti a vytvoření modelu většího dokumentu v paměti může trvat velmi dlouho. Jakmile je však dokument v paměti, může být snadno a rychle načítán upravován a serializován. Nad XML DOM stromem se lze snadno dotazovat pomocí XPath, odmazávat či přidávat celé podstromy uzlů a manipulovat celkově s XML jako se stromem.

3.4 TOM

TOM je zkratka pro Text Object Model. Tento způsob přístupu byl speciálně vyvinut pro aplikaci xmlPad. Umožňuje načtení XML dokumentů do paměti podobně jako XML DOM. XML DOM ovšem načte celý dokument do paměti a je zde uchováván jako strom. Pokud je tedy potřeba tento dokument zpracovávat po částech, lze to jen po podstromech, a to jen u dokumentů, které splňují určitá kritéria. Při zpracování pomocí TOM je XML dokument načten také do paměti a každý uzel je převeden na objekt. Rozdíl od XML DOM je v tom, že TOM neudržuje hierarchii rodič – potomek. Všechny uzly jsou uloženy ve spojovém seznamu a jsou tak všechny na stejné úrovni. Díky tomu, že zde není dodržena hierarchie, může se tento spojový seznam libovolně rozdělit a zpracovávat po částech. Takže při zpracovávání se načte jen potřebná část dokumentu, se kterou uživatel pracuje. Vzhledem k počtu načítaných uzlů pro zobrazení v aplikaci xmlPad jsou v jednom okamžiku v paměti

načteny maximálně dvě části. To se stane při procházení dokumentem na rozhraní dvou částí spojového seznamu.

Na začátku tedy musí aplikace přečíst dokument pomocí SAX parseru nebo XML readeru. Při čtení dokumentu se každý uzel převádí do objektu. Objekty se hromadí do seznamu. Pokud je seznam dostatečně velký, začnou se ukládat objekty do dalšího seznamu, mezi tím je původní seznam uložen na disk. Načtením XML dokumentu do objektů držených v seznamech získáme možnost přistupovat postupně k celému dokumentu. Uživatel se může pohybovat mezi uzly lineárně, tedy se nemůže přesouvat od předka k potomkovi, jediný pohyb je o uzel vpřed a uzel vzad. Při tomto pohybu je tedy načten jen jeden spojový seznam, ostatní jsou serializované na pevném disku a deserializují se jen tehdy, když si uživatel vyžádá uzel z momentálně nenačteného seznamu.

4 Požadavky na program

Základní požadavky na program je možné shrnout do následujících bodů:

- Samotný editor s obvyklými funkcemi
- Kontrola validity a dobré strukturovanosti dokumentů a následné znázornění chyb
- Highlighting
- Kontextové menu
- Generický editor schopný vytvářet editační okna a jejich prvky dle daného DTD nebo schématu v XML Schema
- Podpora velkých souborů

Editor bude uzpůsoben pro editování XML dokumentů, DTD a schémat v jazyce XML Schema. Editovací okno s možností přepínat otevřené dokumenty bude obsahovat obvyklé funkce pro editory

- Editace
- Ukládání, načítání, tisk
- Zpět, vpřed (undo, redo), tj. program si bude udržovat seznam změn, které se na editovaný text aplikují.
- Vyhledávání, nahrazování textu
- Statistiky
- Přepínání mezi otevřenými dokumenty
- Zobrazení navigačního stromu daného dokumentu
- Zpracovávání velkých souborů

XML editory mohou být obecně dvou druhů. Buď jsou elementy v editoru zobrazeny jako objekty a celý dokument se zobrazuje jako strom, nebo funguje v podobě klasického textového editoru. Pro vytvářený editor bude použita druhá varianta, protože umožňuje větší volnost pro uživatele. Program umožní editovat otevřené dokumenty, tzn. po otevření dokumentu může uživatel do dokumentu kamkoliv vpisovat, přepisovat nebo mazat písmena.

Program bude dále obsahovat obvyklé editovací funkce jako jsou Kopírovat, Vystřihnout a Vložit. Kódování, která bude aplikace podporovat budou všechny

nainstalované na aktuálním systému, kde bude aplikace spuštěna. Pro ulehčení práce při vztváření XML dokumentu se při napsaní elementu a uzavření značky ihned napíše uzavírací element. Dokud nebude element správně uzávorkován, nebude editor s dokumentem pracovat jako s XML dokumentem a pochopytelně nebudou fungovat všechny nástroje pro editaci XML dokumentu správně. Tedy highlighting nebude přesně zobrazovat nesprávně uzávorkované části a napovídání s generickým formulářem nebude fungovat.

Pomocí tlačítek undo a redo může uživatel rušit provedené změny, či je zpátky provést. Pro vyhledávání a nahrazování budou využity vestavěné funkce. Tyto funkce budou rozšřeny tak, aby editor uměl vyhledávat podle kontextu. To znamená, že lze zvolit vyhledávání pouze mezi elementy či hodnotami atributů. Statistiky budou obsahovat počty elementů (všech i podle názvu), počty atributů agregovaných podle elementů, ve kterých se vyskytly. Uživatel bude moci editovat najednou více dokumentů a přepínat se mezi nimi, tj. pracovat v prostředí MDI (Multi Document Interface) [26].

Kontrola validity a správné strukturovanosti bude u schémat v XML Schema probíhat podle normy a kontrola XML dokumentu bude probíhat podle příslušných DTD a schémat v XML Schema. Při validaci XML dokumentu se vypíše do speciálního okna hlášení o chybách s případným komentářem a pozicí v XML dokumentu.

Highlighting neboli barevné zvýrazňování slouží k snazší orientaci uživatele. Editor vyhledá různé druhy značek a zvýrazní je barvou, která je přiřazena k dané značce. Tyto barvy bude možné nastavovat v nastavení aplikace. Kontextové menu (našeptávátko) bude při psaní nabízet povolené elementy a atributy, a to buď podle DTD, schématu v XML Schema, nebo podle standardu, pokud bude editovavný dokument XML Schema.

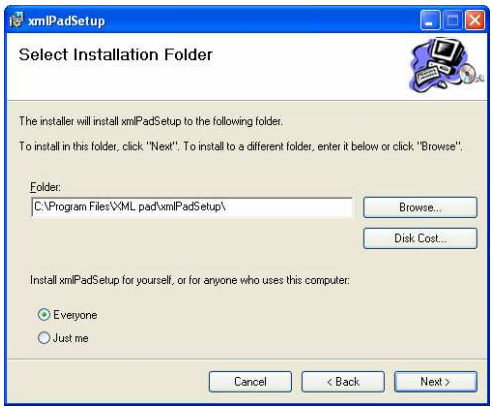
Pomocí generického editoru bude možné vytvářet celé bloky XML dokumentu. Při otevření generického editoru se podle DTD, schématu v XML Schema nebo standardu doplní možné názvy elementů a při editování elementu se objeví atributy se svými hodnotami. Hodnoty elementů a atributů budou nabízeny také podle DTD,

schématu v XML Schema nebo standardu. Uživatel vyplní pole a po potvrzení generický editor do dokumentu vloží dané elementy s jejich atributy.

5 Uživatelská dokumentace

5.1 Instalace

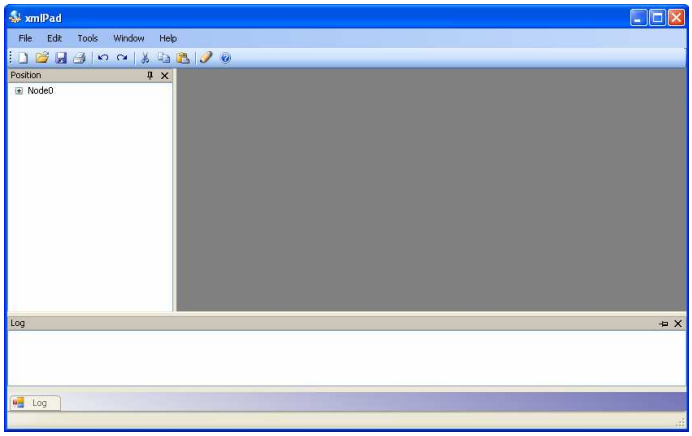
Aplikace je distribuována jako instalační balík. Stačí aby uživatel spustil program setup.exe, který zobrazí průvodce pro instalaci programu, zeptá se uživatele, kam se má aplikace xmlPad nainstalovat a celou aplikaci nainstaluje. Setup také zkontroluje, zda jsou na cílovém počítači potřebné knihovny pro běh aplikace. Pokud ne, pak je nabídnuto uživateli, aby je program automaticky stáhnul z internetu a nainstaloval. Setup při instalaci vytvoří adresářovou strukturu jv\xmlPAD s aplikací xmlPad ve složce pro programy (typicky c:\Program Files). Pro aplikaci xmlPad jsou v adresáři vytvořeny knihovny, spouštěcí soubor a konfigurační soubory. Také jsou do adresáře nahrány složka Samples obsahující příklady a složka Schema s XML schématy. Dále je vytvořen zástupce spustitelného souboru na ploše a v programové nabídce Startu. Setup také automaticky zaregistruje program do nainstalovaných součástí systému. Na obrázku 5.1 je vidět ukázka instalačního okna.



Obrázek 5.1: Instalační okno

5.2 Spuštění

Po instalaci je na cílovém počítači funkční aplikace xmlPad. Uživatel aplikaci spustí buď ze své plochy, nebo z programové nabídky Startu. Po spuštění aplikace se před uživatelem otevře okno xmlPadu, ve kterém je vidět menu a panel s tlačítky pro rychlou volbu (obr. 5.2). Dále je vidět v levé části okno, ve kterém se zobrazuje strom elementů. Ve spodní části je skryté okno pro vypisování logu. Nyní je již vše připraveno pro editaci a vytváření nových XML dokumentů.



Obrázek 5.2: Náhled na hlavní okno

5.3 Podrobný popis pracovního prostředí

V horní části okna se nachází hlavní menu, které obsahuje všechny funkce aplikace. Nástrojová lišta (toolbar) ležící pod menu představuje rychlý přístup k nejpoužívanějším položkám menu. Po najetí myši nad některou z ikon v toolbaru se objeví pro nápovědu název ikony. Název je stejný jako položka v menu, která vykonává stejnou funkci.

Nabídka **File** obsahuje tyto položky:

- *New* – vytvoří nový prázdný XML dokument, do kterého vloží XML deklaraci.
- *Open* – otevře dialog pro otevření souboru.
- *Save* – uloží změny v souboru; jestliže soubor dosud nebyl uložen, zobrazí dialog pro výběr názvu souboru.
- *Save As* – zobrazí dialog pro výběr typu a názvu souboru, do kterého bude diagram uložen.
- *Print* – začne tisknout daný dokument.
- *Print Setup* – nastavení tiskárny.
- *Close – zavře aktivní dokument*
- *Exit – ukončí program*

Nabídka **Edit** obsahuje tyto funkce:

- *Undo* – vrátí zpět poslední změnu
- *Redo* – znovu vykoná akci, která byla jako poslední vrácena zpět a ještě nebyla vykonána znovu
- *Copy, Paste, Cut* – klasické operace pracující se schránkou Windows; akce se vždy týká textu, které jsou označeny.
- *Select All* – označí celý text v aktivním editoru

Nabídka **Tools** obsahuje tyto funkce:

- *Find* – vyhledávání textu, nahrazování textu
- *Convert DTD to XSD* – převede DTD na XML Schema
- *Wizard* – otevře generický editor
- *Insert schemaLocation* – vloží do textu pozici vybraného souboru XML Schema
- *Options* – spuštění nastavení
- *Validate XML* – spustí validaci XML dokumentu
- *Statistic* – provede se statistika XML dokumentu

Nabídka **Windows** obsahuje tyto položky:

- *Close All* – zavře všechny okna s editory
- *Log window*
- *Position window*

Dále jsou v tomto menu zobrazeny všechny otevřené dokumenty a lze se tímto menu mezi nimi přepínat.

Nabídka **Help** obsahuje tyto položky:

- *Help* – otevření uživatelské příručky
- *About* – o programu

V aplikaci se vyskytují plovoucí okna a to okno Position, ve kterém je zobrazen strom elementů. Dále je v aplikaci okno s logem, do něhož jsou vypisovány informace pro uživatele a okno pro vyhledávání a nahrazování. Dalšími okny jsou editovací okna malých a velkých souborů. Také jsou v aplikaci další nástrojová okna, která jsou popsána dále v této kapitole.

5.4 Otevření dokumentu

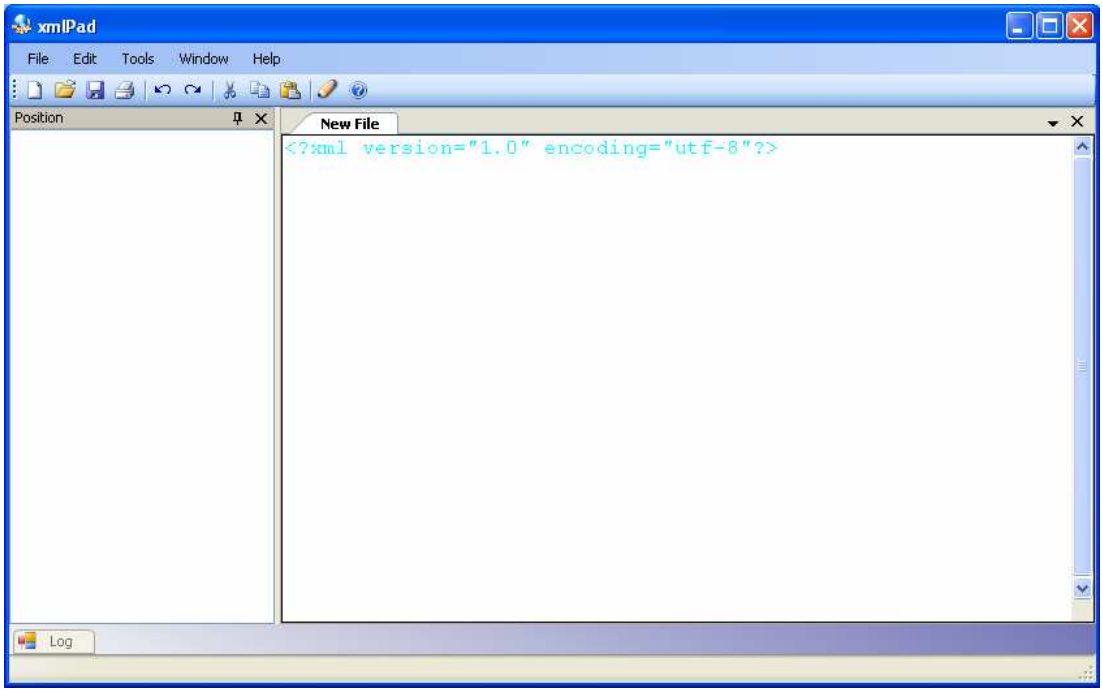
Uživatel může otevřít libovolný XML dokument pro jeho editaci. Otevření se může provést třemi způsoby. Buď přes hlavní menu, kde uživatel vybere položku *File* a poté vybere položku *Open*. Další možnost je pomocí panelu rychlé volby, kde uživatel klikne na tlačítko s obrázkem otevřeného adresáře. Třetí možnost je pomocí klávesové zkratky *Ctrl + O*. Aplikace otevře standardní dialogové okno, ve kterém uživatel zadá cestu k XML dokumentu. Aby mohl uživatel otevřít schéma v jazyku XML Schema, musí nastavit filtr přípon pro XML Schema soubory, to samé pro soubory DTD. Filtr je automaticky nastaven na *.xml a uživateli se zobrazí všechny XML dokumenty v adresáři. Uživatel vybere XML dokument, který chce editovat.

Aplikace tento soubor otevře, načte a vytvoří nové klientské okno uvnitř hlavního. Při otvírání dokumentu je zobrazen čekací formulář a v pozadí mezi tím proběhnou všechny nezbytné úpravy s rozložením komponent na hlavním formuláři. Na čekacím formuláři je zobrazena animace, která upozorňuje uživatele, že aplikace pracuje na pozadí. Do nově vytvořeného okna se zobrazí načtený obsah. Uživatel již může vidět obsah editovaného XML dokumentu. Uživatel může procházet dokument pomocí posuvných lišt vlevo a vespod okna. Může se pohybovat v textu pomocí kurzoru ovládaného šipkami na klávesnici.

5.5 Vytvoření nového XML dokumentu

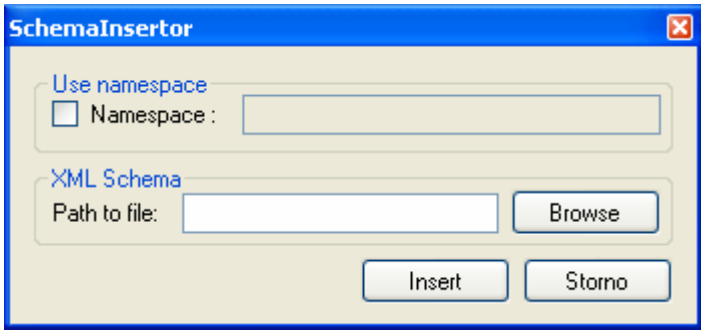
Pro využití všech výhod XML editoru xmlPad je potřeba mít schéma pro vytvářený XML dokument. Tímto schématem je buď DTD nebo v lepším případě XML Schema. Pro ukázku funkčnosti jsou s aplikací distribuovány ukázkové dokumenty. Tyto dokumenty se naleznou v adresáři Samples, který je v kořenovém adresáři aplikace. Nyní uživatel může vytvořit nový XML dokument, který bude odpovídat struktuře nadefinované schématem v jazyku XML Schema v souboru personal.xsd, který je v adresáři Samples. Vytvořit nový dokument lze třemi různými způsoby. První je přes hlavní menu, kde uživatel vyber položku *New*, druhým způsobem je kliknutím na tlačítko panelu rychlé volby s obrázkem prázdného listu a třetím je klávesová zkratka *Ctrl + N*. Aplikace vytvoří nové okno s editovací plochou, ve které je již pro uživatele vepsána XML deklarace (obr. 5.3). Uživatel může na konec

textu připsat kořenový element PurchaseOrder, jakmile uzavře tento element špičatou závorkou, editor automaticky vygeneruje zavírací element.



Obrázek 5.3 : Nový XML dokument

Aby se přiřadilo schéma, podle kterého bude XML dokument vytvářen, k dokumentu je nutné vložit do kořenového elementu určité atributy. Tyto atributy za uživatele do kořenového dokumentu vloží editor. Uživatel musí jen přejet kurzorem do kořenového elementu za jeho název. Poté z hlavního menu vybere položku *Tools*, pod kterou se skrývá další položka pro výběr s názvem *Insert schemaLocation*. Otevře se nástrojové okno (obr. 5.4) a nabídne uživateli určit jmenný prostor a také zadat cestu k souboru s XML Schema. Kliknutím na tlačítko *Browse* se otevře dialogové okno pro otevření souboru. V tomto okně uživatel zadá cestu do kořenového adresáře aplikace, otevře adresář Samples a vybere soubor personal.xsd. Dialogové okno po potvrzení vloží do textu cestu k XML Schema s dalšímu potřebnými atributy a pak tento soubor používá pro určování struktury XML dokumentu.

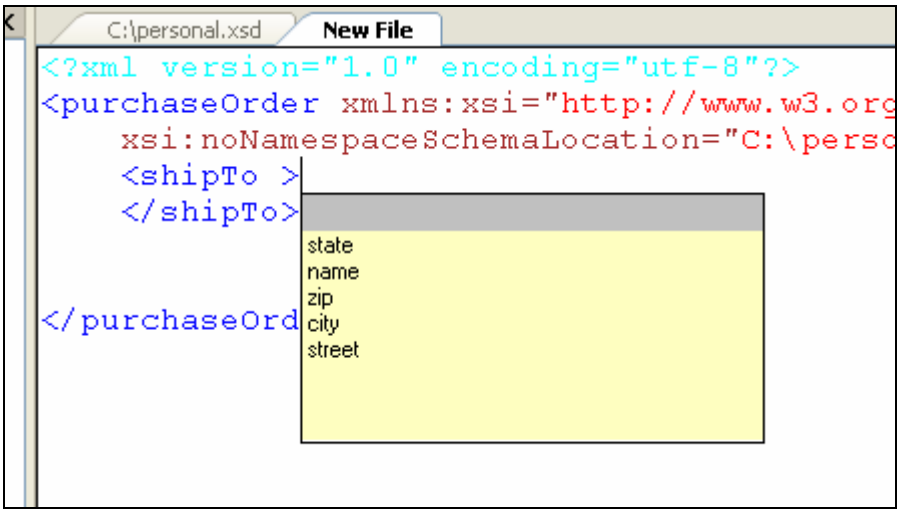


Obrázek 5.4 : Nástrojové okno pro připojení schématu

5.6 Kontextová nabídka

Nyní již uživatel může vytvářet XML dokument s podporou editoru. První ulehčení pro uživatele při vytváření XML dokumentu, který má být strukturovaný podle nějakého schématu, je kontextová nabídka. V této nabídce jsou vždy zobrazeny elementy nebo atributy, které se mohou vyskytnout na pozici kurzoru. Kontextová nabídka se vyvolává sama vždy při napsání úvodní špičaté závorky nebo klávesovou zkratkou *Ctrl + mezerník*. Nabídka se aktualizuje při vybrání elementu na výběr atributů. Po vložení elementu shipTo se nabídka aktualizovala a nabízí atribut editovaného elementu s názvem country. Nabídka se zavře stisknutím klávesy *ESC*. V nabídce si uživatel mezi nabídnutými elementy respektive atributy může vybrat ten požadovaný pomocí šipek a potvrdit klávesou *ENTER* nebo kurzorem myši.

Vrátíme se k vytváření nového XML dokumentu. Prozatím má uživatel vytvořen XML deklaraci kořenový element. Pro pokračování v tvoření XML dokumentu může uživatel posunout kurzor na pozici mezi začáteční a koncovou značku kořenového elementu. Zde může buď napsat levou špičatou závorku, nebo stisknout *Ctrl + mezerník*. Tím se otevře kontextová nabídka, která nabízí jediný element shipTo. Po potvrzení je do editoru vložen začáteční i koncová značka elementu shipTo, kurzor je posunut za název elementu v začáteční značce. V kontextové nabídce je načten atribut, který podle schématu může být vložen, v našem případě atribut country. Při potvrzení tohoto atributu se vloží do elementu shipTo. Jelikož má tento atribut fixní hodnotu je automaticky vložena do hodnoty atributu.

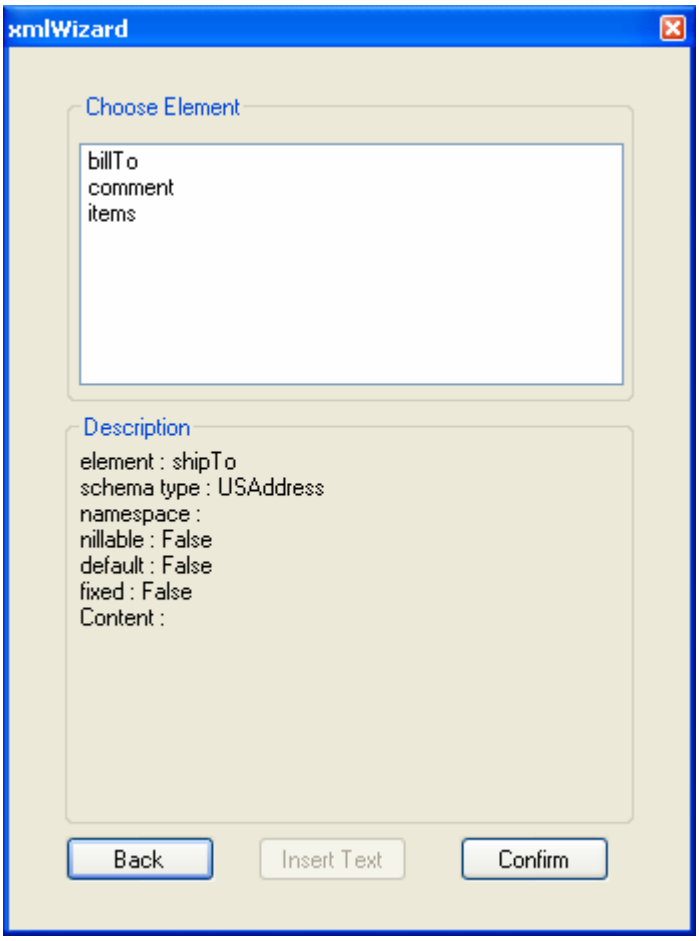


Obrázek 5.5 : Ukázka kontextového menu

Na obrázku 5.5 je vidět výběr z více elementů. Tyto všechny elementy jsou ve schématu uvedeny v neuspořádané množině elementů a může být na tomto místě vložen kterýkoli z nabízených elementů.

5.7 Generický formulář

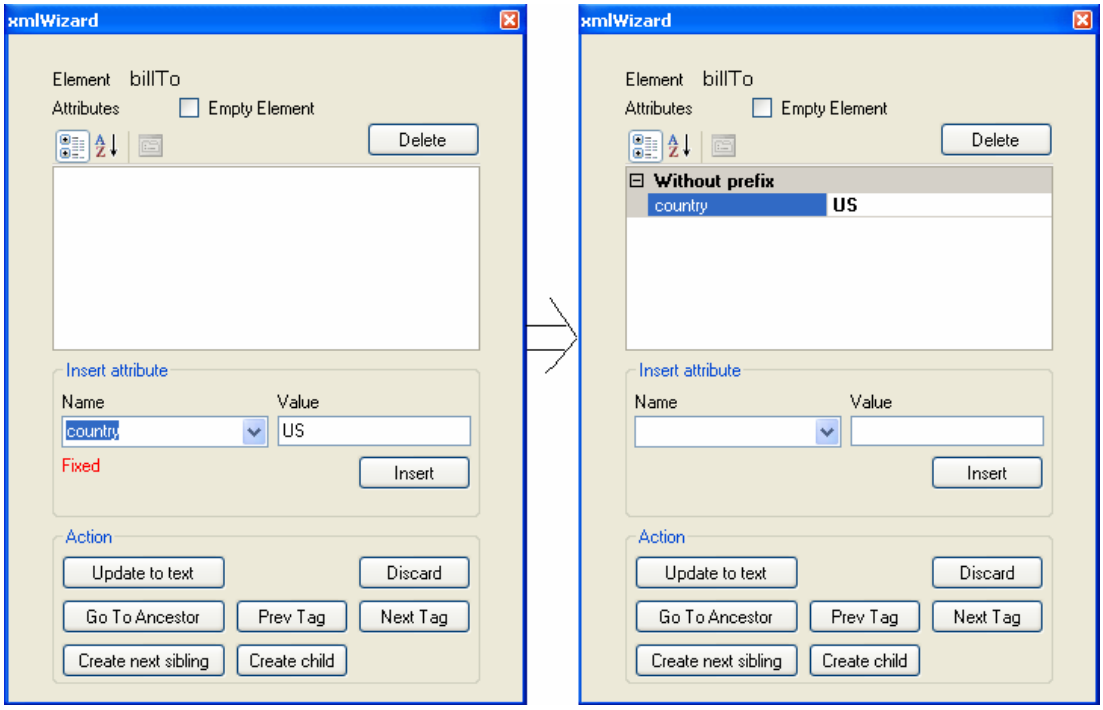
Dalším pomocníkem pro uživatele je generický formulář. Uživatel ho spustí buď přes hlavní menu, kde pod položkou *Tools* vybere položku *Wizard* nebo přes panel rychlé volby, kde je generický formulář spuštěn tlačítkem s obrázkem tužky. Před spuštěním generického formuláře musí stát kurzor někde uvnitř kořenového elementu (tzn. mezi jeho začáteční a koncovou značkou, mimo kořenový element nemá smysl vyvolávat generický formulář). Podle toho kde kurzor stojí je vyvolán příslušný druh generického formuláře. Když je kurzor uvnitř začáteční značky některého elementu je vyvolána obrazovka s editováním tohoto elementu. Pokud je kurzor jinde než v začáteční značce některého z elementů je vyvolána obrazovka s výběrem vložení nových elementů.



Obrázek 5.6 : Generický formulář pro výběr elementu

Pro pokračování v prováděném příkladu uživatel posune kurzor za konečnou značku právě vytvořeného elementu shipTo a jedním ze dvou uvedených způsobů spustí generický formulář. Je otevřeno nové dialogové okno, ve kterém je seznam možných elementů, které mohou být zobrazeny na dané úrovni, při vybrání elementu se pod ním vypíše popis tohoto elementu. Uživatel má na výběr elementy s názvy billTo, comment, items (obr. 5.6). Průvodce zobrazil tyto tři elementy, protože elementy billTo a comment mají ve svém schématu napsáno minOccurs="0", tudíž mohou, ale nemusí být ve výsledném XML dokumentu uvedeny. Uživatel vybere element billTo a potvrdí. Editor vloží nový element a zobrazí nové okno formuláře, které je nyní přizpůsobeno k editování elementu. Zde může uživatel přidávat, mazat a měnit atributy elementu a také funguje jako malý rozcestník. Element billTo má jediný atribut a to country. Na obrázku 5.7 je vidět vybrání tohoto atributu ze seznamu nabízených atributů. Také je vidět, že má tento atribut fixní hodnotu US, která se automaticky vyplnila do textového pole. Druhý stav tohoto okna na obrázku vpravo

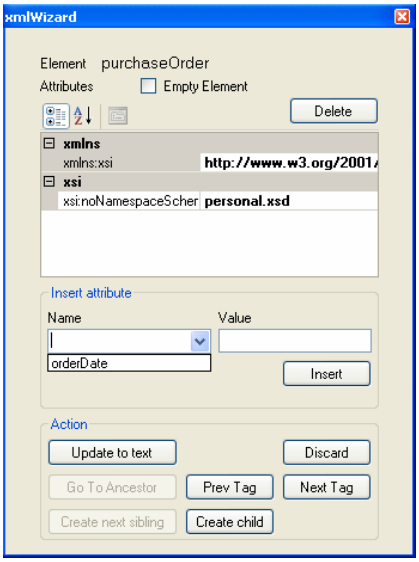
ukazuje editaci elementu již po vložení elementu. Atributy elementu mohou být mazány, stačí označit atribut a poklepat na tlačítko *Delete*.



Obrázek 5.7 : Vložení atributu

Pokud by byl kurzor uvnitř elementu s textovým nebo smíšeným obsahem, pak by průvodce kromě nabídky možných elementů také umožnil stisk tlačítka *Insert Text*. Toto tlačítko zobrazí třetí typ generického formuláře pro vkládání textu. Na obrazovce je jednoduché textové pole, do kterého může uživatel psát. Po vložení textu je uživatel vrácen na obrazovku pro výběr elementů. Tlačítkem *Back* se uživatel dostane na editaci rodičovského elementu.

Na formuláři pro editaci elementu jsou ještě tlačítka pro pohyb po značkách. Pokud uživatel právě edituje element billTo, pak při stisknutí *Go to ancestor* se dostane uživatel do editace rodičovského elementu (v tomto případě kořenového elementu). Atributy jsou rozděleny podle svých prefixů do skupin jak je vidět na obrázku 5.8. Uživatel může vložit nový atribut, který ještě chybí kořenovému elementu purchaseOrder. V části Insert attribute vybere z rozvinovacího menu jediný atribut, který je nabízen a do textového pole vlevo napíše jeho hodnotu. Při stisknutí tlačítka Insert je tento atribut vložen do elementu.



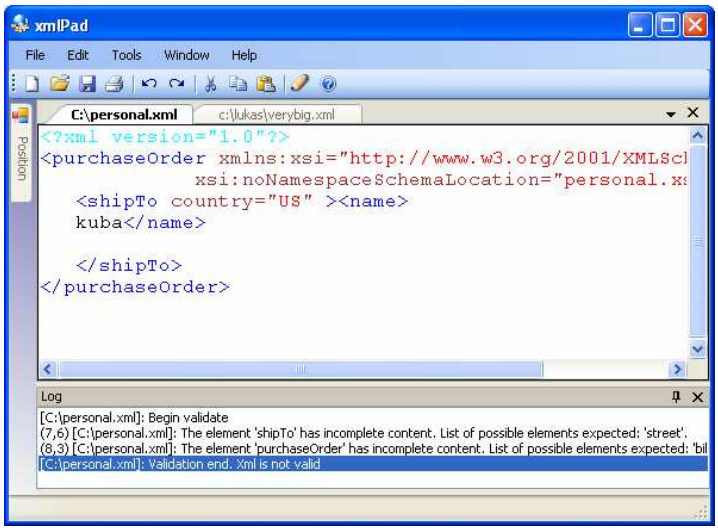
Obrázek 5.8 : Editace elementu

5.8 Vyhledávání

Uživatel může vyhledávat v otevřeném XML dokumentu textový řetězec. Tento textový řetězec může být vyhledáván v celém textu. Uživatel si také může nastavit vyhledávání v určitém kontextu. To znamená, že aplikace rozezná, zda je nalezený text součástí názvu elementu, názvu atributu hodnoty atributu nebo textová hodnota elementu. Uživatel si tedy může vybrat v jakém kontextu má být hledaný řetězec vyhledán. Při hledání lze ještě nastavit, kterým směrem má být řetězec vyhledáván, zda má aplikace rozlišovat velká a malá písmena nebo zda má být řetězec hledán jako celé slovo.

5.9 Validace XML dokumentu

Validace dokumentu určí zda je dokument napsán podle struktury, kterou určuje jeho schéma. Uživatel spustí validaci z hlavního menu. Pod položkou *Tools* je položka *Validate XML*, která spustí validaci dokumentu. Výsledek validace se zobrazí do okna s názvem Log jak je vidět na obrázku 5.7. V logu je vypsána pozice chyby a je popsána její příčina. Pokud chce uživatel přejít ukazatelem na nalezenou chybu, stačí když dvakrát klikne na řádek s chybou. Editor automaticky přemístí kurzor na chybu.



Obrázek 5.9 : Validace

5.10 XML Schema

Editace dokumentu XML Schema uživateli poskytuje stejný komfort jako u editování běžného XML dokumentu. Aplikace umí napovídat elementy a atributy ať už pomocí kontextové nabídky nebo pomocí generického formuláře.

Při otevření souboru s příponou .xsd, je automaticky načteno schéma XML dokumentu podle normy. Editor tedy podle této normy může plnit kontextovou nabídku. Pokud je potřeba editovat DTD, je to možné, ale jelikož to není XML dokument, tak pouze bez napovídání editorem. Nebo je možné převést DTD pomocí nástroje aplikace do XML Schema. Tato funkce je v hlavní nabídce schována pod položkou *Tools* s názvem *Convert DTD to XSD*. Nyní je převedeno DTD do XML Schema a může být dále editováno jak bylo popsáno výše.

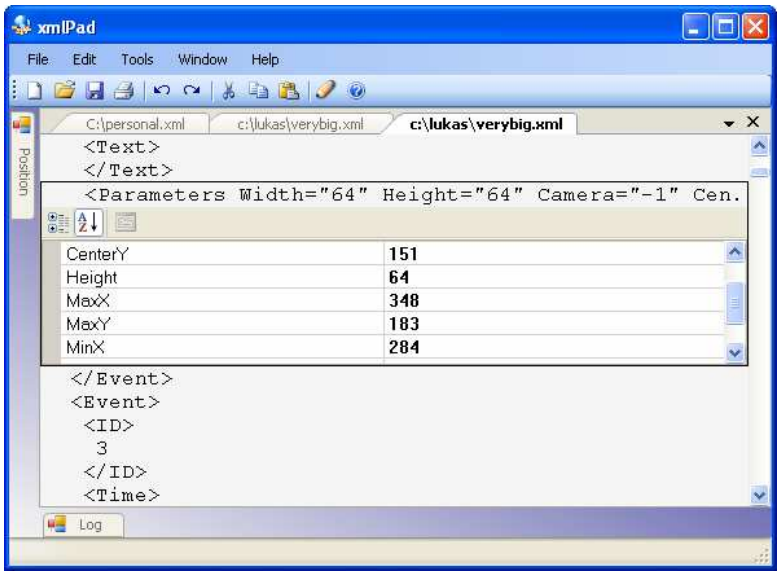
Při změně schématu pro XML dokument, se musí celý dokument uložit, zavřít a znovu načíst. Toto opatření je z důvodu optimalizování načítání schématu k XML dokumentu.

5.11 Zpracování velkých dokumentů

Velkou předností aplikace xmlPad je možnost zpracování velkých dokumentů. Pro velké dokumenty je v aplikaci speciální editor velkých dokumentů. Pokud je

při otvírání soubor větší než hodnota, která je v nastavení aplikace, pak se automaticky otevře XML dokument v editoru pro velké soubory. V tomto editoru lze otvírat jen dokumenty, které jsou správně zformované.

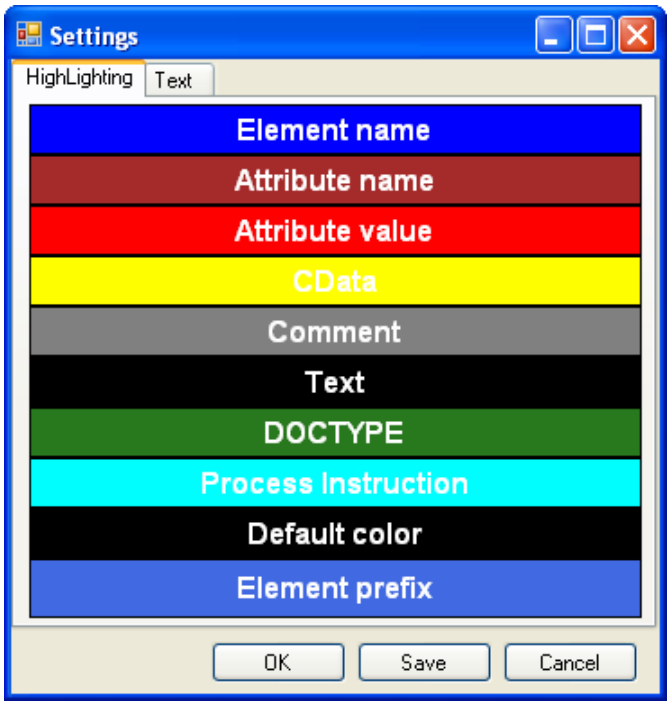
Jelikož je čtení z pevného disku náročné může otevření velkého dokumentu zabrat určitý čas. Po otevření a načtení XML dokumentu je v editoru na každém řádku zobrazen jeden uzel XML dokumentu. V tomto případě je brán uzel z textového pohledu. Tedy uzel odpovídá každé značce (začáteční nebo koncové), textovým uzlům a dalším uzlům XML dokumentu. Pomocí posuvných lišt se lze pohybovat po dokumentu. Pro detailnější pohled na uzel na něj stačí dvojkliknout. Zobrazí se nám detailnější pohled (obr. 5.8), ve kterém lze vybraný uzel editovat. Na obrázku je vidět editování elementu Parameters. Je zde výpis atributů s jejich hodnotami. Ve výpisu lze přímo editovat hodnoty. Pro vymazání nebo vložení atributu slouží kontextové menu, stačí kliknout pravým tlačítkem a je toto menu zobrazeno.



Obrázek 5.10 : Editor velkých souborů

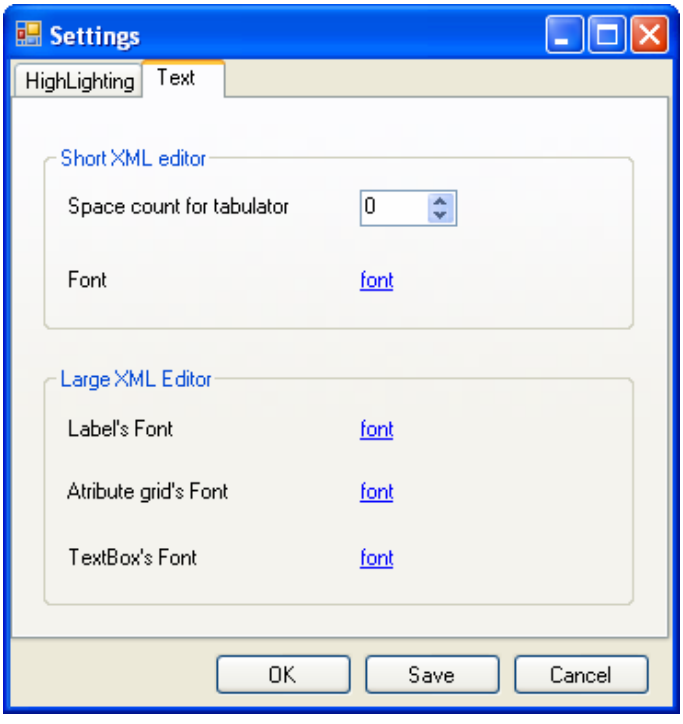
5.12 Nastavení aplikace

Aplikace umožňuje uživateli měnit různá nastavení. Tato nabídka je pod položkou *Tools* a následně položka *Settings*. V nastavení lze měnit barvy, které jsou použity při zvýrazňování XML dokumentu, jak je vidět na obrázku 5.11.



Obrázek 5.11 : Nastavení barev

Dále aplikace umožňuje měnit nastavení fontů (obr. 5.12) jak v editoru pro malé dokumenty, tak v editoru velkých dokumentů. V editoru velkých dokumentů je navíc rozděleno nastavení fontů do více kategorií. V editoru pro malé dokumenty ještě lze nastavit kolika mezerami má být nahrazen tabulátor.



Obrázek 5.12 : Nastavení textu

6 Vývoj aplikace

Tato kapitola přibližuje naprogramované jádro aplikace xmlPad a popisuje principy, na kterých je založena funkčnost programu. Dále zde jsou popsány problémy, které se vyskytly při jejím vývoji. Aplikace je naprogramována v prostředí Visual Studio 2005 [27] v programovacím jazyku C# [43]. Při vývoji byly navíc použity následující volně dostupné externí knihovny.

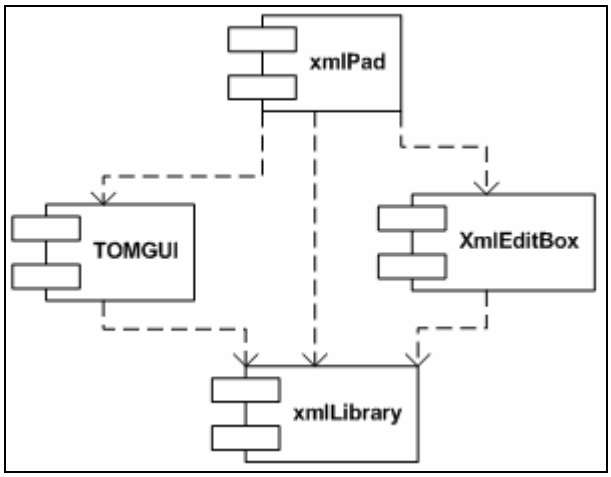
Pro uživatelské prostředí byla použita knihovna WeifenLuo.WinFormsUI.Docking [28], kterou vyvynul Weifen Luo. Tato knihovna umožňuje nadefinovat různé druhy oken, které se zobrazí uživateli. Jedním z možných druhů oken je plovoucí okno, další je postraní okno s možností automatického schovávání. Toto postraní okno může být uživatelem přemísťováno a přichycováno k libovolnému okraji hlavního okna aplikace.

Statistika dokumentu je vytvářena pomocí knihovny XmlStats [29], která analyzuje celý dokument a vypíše jako text. Pro přehlednější zobrazení byla knihovna upravena, aby mohli být jednotlivé statistické údaje z knihovny získány zvlášť a ne v souvislém textu.

Pro editaci dokumentů se schématem v DTD je využito převoditelnosti DTD na XML Schema. Pokud tedy má editovaný dokument schéma v DTD je převedeno na XML Schema a dále se pracuje s dokumentem jakoby měl schéma v XML Schema. K převodu schémat DTD do jazyku XML Schema je využito knihovny DTD2XSD [42]. Pro přístup do vestavěné komponenty Visual Studia property grid je použito knihovny PropertyBag [30]. Tisk je realizován pomocnou knihovnou PrintTextDocument [31].

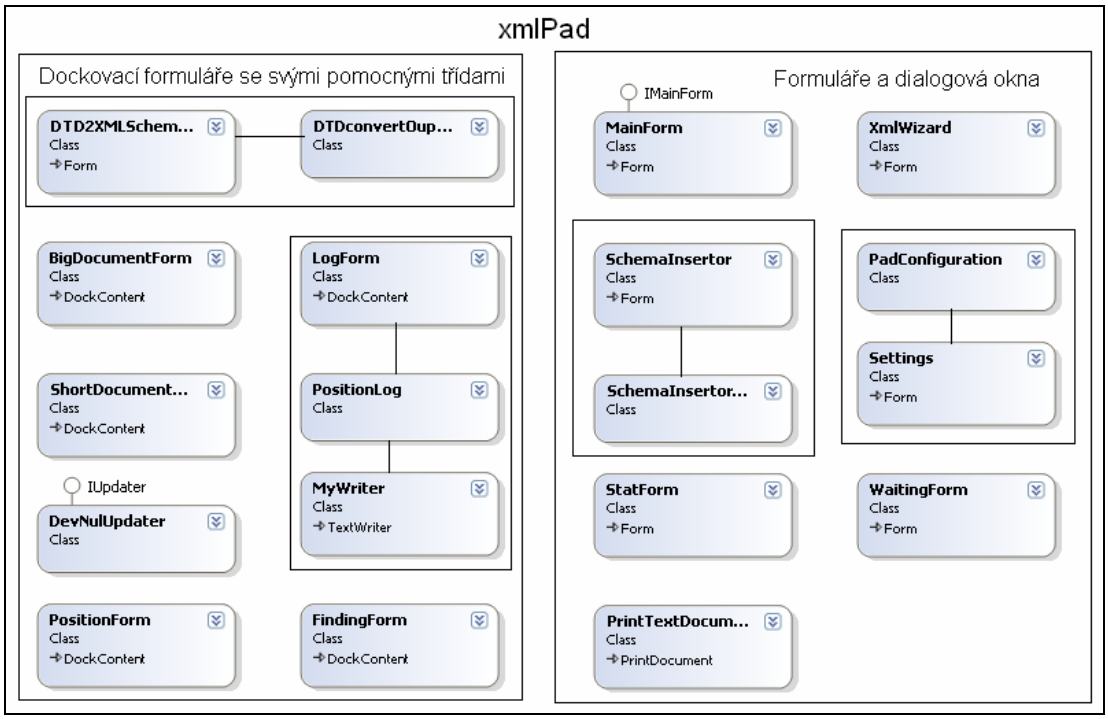
6.1 Rozdělení do komponent

Celý projekt je rozdělen do několika komponent. Komponenta se všemi formuláři se jmenuje xmlPad a slučuje všechny další komponenty. Další komponenta je XmlEditBox, ve které jsou třídy související s editovacím oknem a kontextovým menu, je zde také třída, která zařizuje highlighting. Pro velkou část logiky, se kterou aplikace pracuje, je dále vytvořena komponenta xmlLibrary. Poslední komponentou této aplikace je TOMGUI, která obsahuje komponenty uživatelského prostředí pro práci s velkými soubory. Na obrázku 6.1 je vidět zjednodušené schéma zobrazující závislost komponent.



Obrázek 6.1: Závislost komponent aplikace

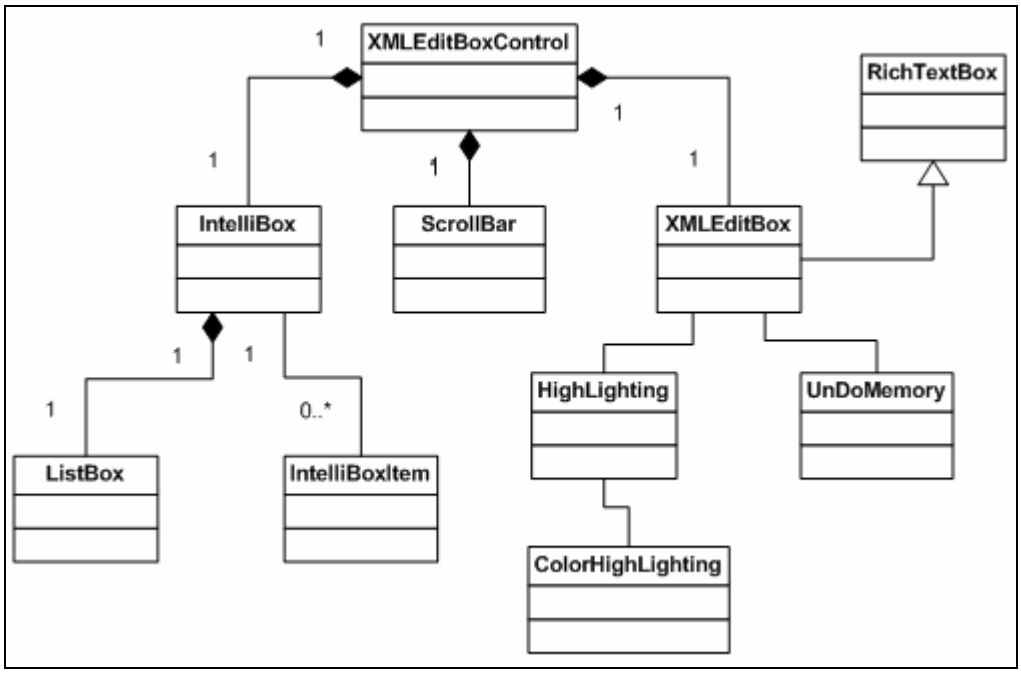
Komponenta xmlPad obsahuje základní formuláře aplikace. Obrázek 6.2 zobrazuje detailnější pohled na komponentu xmlPad. Obrázek je rozdělen do dvou hlavních částí. V levé části jsou formuláře, které se mohou přichytávat ke stranám okna (tzv. dockovací formuláře). Uvnitř této části jsou vidět dva bloky. Tyto bloky zobrazují vzájemnou provázanost pomocné třídy a třídy pro formulář. V pravé části obrázku jsou třídy pro hlavní formulář a pro nástrojová okna. Opět zde jsou bloky slučující spolupracující třídy.



Obrázek 6.2 : komponenta xmlPad

Komponenta XmlEditBox obsahuje třídy pro uživatelské rozhraní. Tyto třídy a jejich provázanost je vidět na obrázku 6.3. Je zde třída XMLEditBox, která je potomkem vizuální komponenty RichTextBox. Tato třída dále využívá třídu HighLighting pro barevné zvýrazňování textu. Třída Highlighting získává aktuální hodnoty barev pro obarvení uzlů XML dokumentu ze třídy ColorHighLighting. Třída XMLEditBox umožňuje návrat do historie úprav textu. K tomu slouží třída UnDoMemory.

Dále je zde vizuální komponenta IntelliBox umožňující zobrazování kontextové nápovědy. Pro zobrazení nabízených elementů či atributů je využito ListBoxu. Komponenta obsahuje seznam nabízených položek. Další součástí tohoto balíku je třída, která sdružuje třídu kontextové nápovědy s XMLEditBoxem. Tato vizuální komponenta je využívána ve výsledné aplikaci.



Obrázek 6.3 : XMLEditBoxControl a přidružené třídy

Komponenta TOMGUI obsahuje komponenty uživatelského rozhraní. Tyto komponenty slouží jako součást editoru velkých XML dokumentů. Pro každý druh uzlu XML dokumentu je v této komponentě jedna vizuální komponenta. Každá komponenta umožňuje editaci uzlu, který zobrazuje. Také je zde třída uchovávající nastavení těchto vizuálních komponent.

Komponenta xmlLibrary obsahuje třídy pro načtení XML dokumentů a práci s nimi. Jednotlivé třídy a algoritmy jsou rozebrány později v této kapitole. Ještě zmíníme, že tato komponenta obsahuje logickou vrstvu pro oba balíky s vizuálními komponentami TOMGUI a XmlEditBox.

6.2 Highlighting

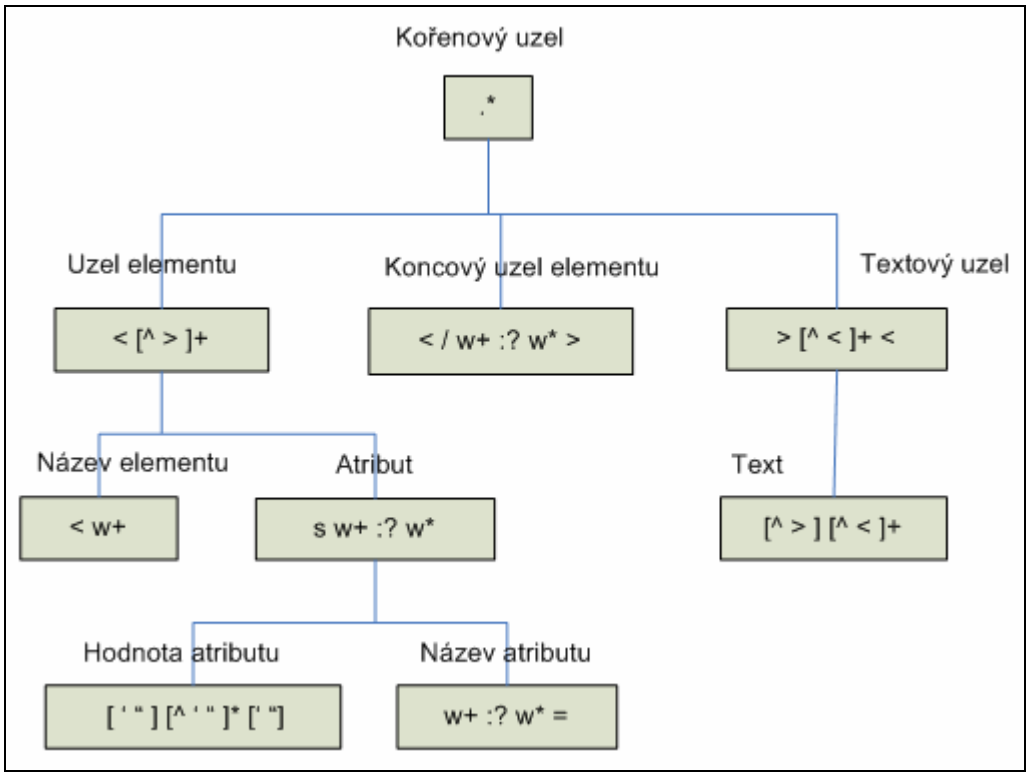
Jako první bude rozebrán způsob barevného zvýrazňování textu. Pro editaci textu je v aplikaci použito vestavěné komponenty programovacího prostředí se jménem RichTextBox. Tato komponenta umožňuje zobrazovat text s různými formátováními částí textu. Při obarvování různých druhů značek XML dokumentu je potřeba najít začátek a konec značky a určit její druh. Jakmile máme tyto pozice pak danému textu podle druhu značky přiřadíme barvu, kterou získáme z konfigurace aplikace.

První způsob kterým může být realizován highlighting je založen na konečném automatu [44]. Tento konečný automat postupně čte znaky v dokumentu a podle stavu, ve kterém se nachází, a přechodové tabulky se posune do dalšího stavu. Pokaždé když je automat v příjmacím stavu je zaznamenána pozice v textu a druh značky, která je přečtena. Výstupem algoritmu využívající tento automat je seznam pozic začátků a konců značek a jim odpovídajících druhů z celého XML dokumentu. Tento seznam je seznamem objektů pro popis značek.

Po zpracování konečným automatem přichází na řadu highlighting. Na začátku obarvování se vypne kreslení komponenty, aby neblikala a nijak se nepohybovala, což by byl pro uživatele nežádoucí efekt. Poté se prochází seznam objektů pro popis značek. Pro každý objekt se označí text vyznačení udanými pozicemi v objektu. Poté je na komponentu RichTextBox zavolána metoda, která změní barvu právě označeného textu. Barvu určuje druh značky, který je také zaznamenán v objektu pro popis značek.

Velká nevýhoda tohoto přístupu je v neustálém označování a barvení označovaného textu, což průběh obarvování zpomaluje. Další nevýhodou je sestrojený konečný automat, který není nejefektivnějším řešením. Při větších objemech textu je editování nereálné a nelze tento způsob použít.

Další způsob analýzy XML dokumentu na druhy značek a jejich pozice je při použití regulárních výrazů [32]. Víme, že regulární výrazy mají stejnou vyjadřovací schopnost jako konečné automaty [45]. Pro vyhledávání značek a rozlišení jejich druhů se použijí regulární výrazy, které se postupně aplikují na celý XML text. Nejprve se vyhledají nadřazené skupiny a poté se pokračuje k podřazeným. Posloupnost aplikování regulárních výrazů odpovídá stromu jak je vidět na obrázku 6.4. Například je nejprve označen regulárním výrazem v kořenu stromu celý XML dokument. Na tento text se dále aplikují regulární výrazy, které najdou další značky jako celý element. Element se dále dělí na jméno elementu a atributy. Atributy se dále dělí na název atributu a na hodnotu atributu. Jakmile je použita celá stromová hierarchie regulárních výrazů dostaneme opět seznam objektů pro popis značek.



Obrázek 6.4 : Stromové uspořádání aplikování regulárních výrazů

Další vylepšení, které může být oproti minulému způsobu použito, je místo označování a přebarvování částí textu, které zpomaluje průběh obarvování, vytvořit RTF dokument. Ve vytvořeném dokumentu je editovaný text s přidanými řídícími značkami RTF dokumentu, které určují barvu. RTF značky jsou přidány do dokumentu na základě analýzy pomocí regulárních výrazů. Celý RTF text pak přiřadíme do komponenty RichTextBox. Přiřazením je RTF dokument parsován komponentou a to zabere při rozsáhlých datech patřičný čas. Proto při použití této metody stále docházelo k velké prodlevě při editaci větších dokumentů a není možné tento způsob použít.

V aplikaci xmlPad je použit právě tento poslední způsob obarvování značek. Analýza XML dokumentu je zjednodušena z vyhledávání značek pomocí regulárních výrazů. Pro analýzu se použije prioritní vyhledávání značek s jejich následným zpracováním. Podle stanovené priority značek, která zapříčiní to že se značky nepomíchají je XML dokument analyzován. Nejprve se tedy analyzují XML deklarace, instrukce pro zpracování, DOCTYPE značky a nakonec elementy. Analýza probíhá na základě hledání počátečních a koncových značek, proto musí být hledání elementu až za všemi ostatními druhy značek. Začátek elementu je levá špičatá závorka, jako

všech jiných jmenovaných druhů značek, proto by došlo ke špatnému přiřazení druhů značek kdyby se hledaly elementy dříve.

Při prohledávání těchto značek se ukládají informace o pozicích začátku a ukončení značky do objektu pro popis značek. Začátek a konec značky určuje interval pozic. Zpracováním tak vzniká množina disjunktních intervalů, která pokrývá daný dokument. Nalezené značky se poté zpracují každá zvlášť. Při tomto zpracování je již známo jaká značka je zpracovávána. Například ve zpracování značky elementu se snadno naleznou atributy a ty se poté rozdělí na názvy atributů a hodnoty atributů. Z tohoto seznamu objektů pro popis značek se zase utvoří RTF dokument, který se předá RichTextBoxu pro vykreslení uživateli.

Poslední vylepšení spočívá v analyzování pouze zobrazené části XML dokumentu a jejím nahrazením v RichTextBoxu vytvořeným RTF dokumentem. Při highlightingu se tedy určí pozice od které je dokument zobrazen uživateli a pozice kde končí zobrazený výřez dokumentu. Tyto pozice se ještě upraví, aby na začátku fragment XML dokumentu byla levá špičatá závorka nebo aby fragment začínal hned za pravou špičatou závorkou. Pro konec fragmentu to je obdobné, tedy fragment musí končit buď v místě pravé špičaté závorky nebo před levou špičatou závorkou. I po této úpravě musí fragment obsahovat zobrazenou část uživateli. Proběhne analýza tohoto fragmentu, ze které je vygenerován seznam objektů pro popis značek. Seznam je použit pro vygenrování RTF dokumentu, kterým je poté nahrazena viditelná část XML dokumentu v komponentě RichTextBox.

Problém u zpracování pouze viditelné části je v komentářích a sekci CDATA. Tyto dva bloky mohou začínat daleko před viditelnou částí, takže ani nelze zjistit z malého okolí viditelné části, zda je nebo není zkoumaný fragment uprostřed komentáře. Proto je v editoru udržována struktura, ve které se zaznamenávají při každé editaci začátky a bloků CDATA a komentářů. To umožňuje určit zda je výřez zobrazený uživateli uvnitř bloku či nikoliv.

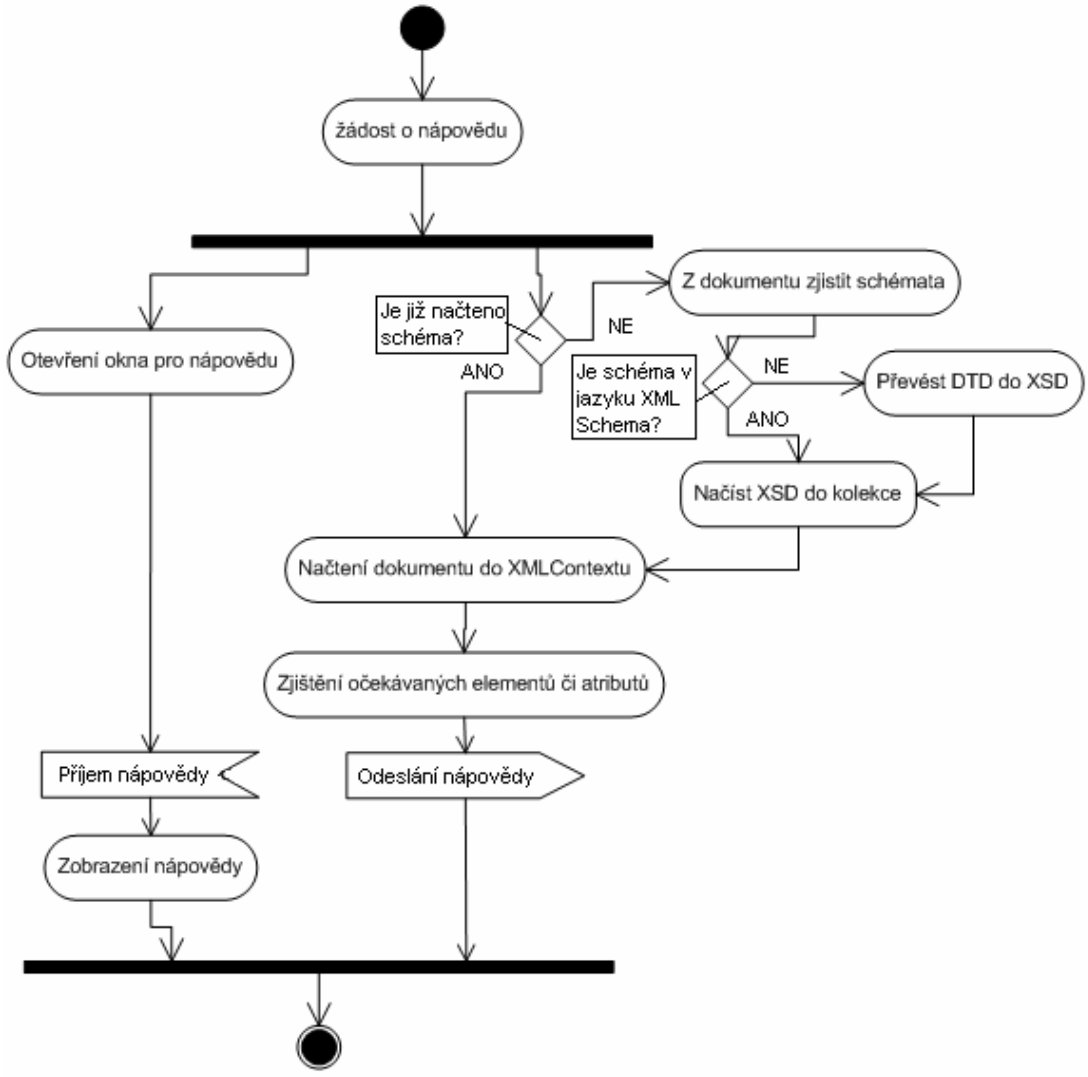
Nakonec bylo třeba vyřešit problém při zjišťování začátku viditelné oblasti. Komponenta RichTextBox neumožňuje přímé zjištění pozice posuvné lišty (scroll bar), a ta musí být zjištěna pomocí poslání zprávy přes win32 API [33]. Problém je

v tom, že komponenta RichTextBox vrací hodnotu z intervalu [0, 65535]. Tudíž dokumenty s větším počtem řádku nebyly zobrazovány korektně. Proto bylo použito vlastní posuvné lišty, která ovládá posun uvnitř RichTextBoxu a ze které lze zjistit hodnotu prvního řádku přesně.

6.3 Kontextové menu

Kontextové menu je tvořeno listboxem, ve kterém se zobrazí elementy, případně atributy, které se mohou vyskytnout na dané úrovni. Celé kontextové menu je vytvořeno jako vizuální komponenta. V editoru je tato komponenta součástí komponenty XMLEditBoxControl jak je vidět v kapitole 6.1 na obrázku 6.3.

Kontextové menu se automaticky otevře při napsání špičaté levé závorky nebo při stisknutí Ctrl + mezerník. Po aktivování kontextové nápovědy se vytvoří zvláštní vlákno, které zařídí naplnění kontextové nápovědy. Nejprve je z XML dokumentu zjištěno jaké schéma je použito a to se načte do kolekce schémat. Pokud je použito schéma DTD, je nejprve pomocí externího nástroje DTD2XSD [30] převedeno do schématu v jazyku XML Schema a poté vloženo do kolekce schémat.

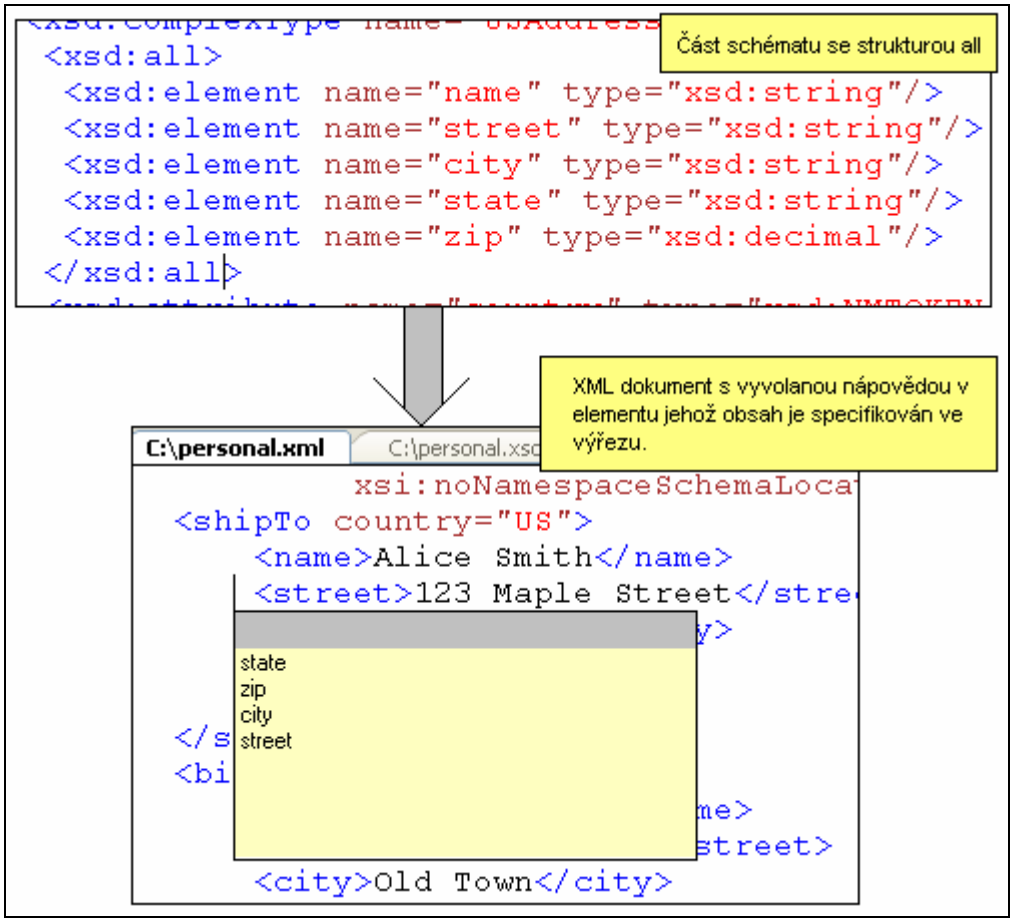


Obrázek 6.5 : Proces načtení nápovědy

Naplnění listboxu názvy elementů a atributů vykonává metoda, která analyzuje dokument. Potom pomocí validátoru zjistí, které elementy nebo atributy se mohou na dané úrovni vyskytnout. Analýza probíhá čtením dokumentu pomocí XML Readeru a přetvářením XML dokumentu do instance třídy XML Context. Do této třídy se ukládají elementy se svými atributy, udržuje se zde hierarchie elementů a buduje se zkrácený DOM strom pouze z elementů ležících na cestě k editovanému elementu. Context se buduje, dokud se XML reader nedostane až na místo, kde se právě nachází kurzor komponenty RichTextBox. V tu chvíli jsou v XML Contextu uloženy elementy se svými atributy, které jsou rodiči a sourozenci elementu nejblíže kurzoru. Poté jsou elementy v XML Contextu předávány třídě XmlSchemaValidator a tou jsou postupně vylisovány. XMLSchemaValidator je tedy v XML dokumentu na pozici, kde je kurzor. Z vnitřního stavu XmlSchemaValidatoru jsou z získány

elementy případně atributy, které jsou dále očekávány. A ty jsou předány kontextovému menu, kde jsou zobrazeny. Celý tento proces zobrazuje obrázek 6.5.

Pro příklad je uvedena ukázka (obr. 6.6) při aktuální pozici kurzoru v elementu shipTo za jeho potomkem, elementem name. Na obrázku je také výřez schématu, který ukazuje datový typ elementu shipTo. Podle tohoto schématu by tedy měl element shipTo obsahovat neuspořádanou pětici elementů, tedy nezáleží na jejich pořadí, ale mohou se zde vyskytnout pouze jednou. Na obrázku je vidět, že kontextová nápověda ukazuje čtyři elementy odpovídající výčtu elementů v definici datového typu shipTo. Chybí pouze nabídka elementu name. Tento element není nabízen, jelikož se kurzor v editoru nachází již za elementem name.



Obrázek 6.6 : Ukázka kontextové nabídky

Na obrázku je také vidět že je nabízen element street, i když je již v elementu shipTo vepsán. Je to proto, že jsou brány v potaz pouze elementy, které jsou před kurzorem. Element street je za kurzorem a není tedy brán v potaz při vyhodnocování nápovědy.

Díky tomu, lze na každé úrovni zjistit, jaké se tam mohou vyskytnou elementy aniž by byly brány v potaz stávající. Další výhodou vynechání elementů za kurzorem je zrychlení analýzy dokumentu.

6.4 Vyhledávání

Pro vyhledávání textu je využito vestavěné metody RichTextBoxu, která vyhledává text. Jako parametr této metody lze nastavit, jestli má při vyhledávání rozlišovat velikost písmen nebo zda má vyhledávat odzadu. Metoda hledání je pouze využita metodou, která v aplikaci zprostředkovává hledání. Nová metoda pro hledání umí rozlišit v jakém je kontextu, tedy zda je nalezený text v názvu elementu, v názvu atributu, v hodnotě atributu nebo v textové hodnotě elementu. Uživatel má tak možnost vybrat, v jakém kontextu má být vyhledávaný text. Vyhledávání v tomto případě funguje tak, že se postupně prohledává dokument a zjistí se v jakém kontextu je nalezený text. Pokud se určí, že je kontext správný, pak to je ohlášeno uživateli, pokud ne, hledání pokračuje.

Kontext nalezeného textu se určuje stejně jako se určuje druh značky při highlightingu. Naleznou se nejbližší špičaté závorky a pokud je text ihned za špičatou závorkou, potom to je název elementu. Pokud je text uvnitř špičatých závorek, ale není ihned za závorkou pak to je buď název atributu, nebo hodnota atributu. To se dále určí podle pozice uvozovek a rovnítka, pokud je text mimo špičaté závorky, pak se jedná o text. Také se z třídy využité pro zaznamenávání komentářů a sekcí CDATA zjistí, zda není nalezený text uvnitř komentáře nebo sekce CDATA.

6.5 Genericky formulář

Generický formulář je pojat jako průvodce pro tvorbu, případně editaci XML dokumentu. Průvodce má tři druhy obrazovek. Z pozice kurzoru v momentě kdy byl spuštěn průvodce se určí která obrazovka průvodce se zobrazí. První je pro výběr elementu a je zobrazena pokud kurzor RichTextBoxu stojí mimo jakýkoliv tag. Na této obrazovce má uživatel možnost vybrat element který bude vložen, případně vložit text, pokud v tomto elementu může být. Nabídka elementů je získána stejným

způsobem jako se získává pro kontextové menu. Algoritmus pro zjištění očekávaných elementů umí zjistit zda může element obsahovat i text. Při vložení elementu je na místo kurzoru vepsán otevírací i zavírací element. Při vložení textu se zobrazí druhá obrazovka.

Na druhé obrazovce je editovací okno pro vložení textu. Lze zde pouze editovat text, který je následně vložen na pozici daného kurzoru. Po vložení se opět zobrazí první obrazovka. Při vložení elementu a jejich textovém vytvoření v RichTextBoxu se posune kurzor na pozici začátečního tagu a zobrazí se třetí obrazovka průvodce, která slouží pro editaci elementu.

Jako vstupní data jsou opět použity nápovědné atributy, které se získají stejně jako v kontextovém menu. Dále je zde využito analýzy dokumentu pro získání pozic začáteční a koncové značky předka, koncové značky právě editovaného elementu a pozice následující a předcházející značky. Termíny následující a předcházející značka jsou brány ve smyslu lineárního čtení od začátku do konce dokumentu. Tedy ne podle stromové hierarchie. Pozice koncové značky předka je použita pro vytvoření dalšího sourozence a pozice koncové značky právě editovaného elementu je použita pro vytvoření potomka. Na této obrazovce je kromě panelu pro pohyb mezi elementy tabulka s použitými atributy a rozvinovací menu s přípustnými atributy. Celá obrazovka je naplněna daty z třídy, která se vytvořila ze získaných dat z XML Contextu a z analýzy XML dokumentu. Obsahuje jméno atributu, prefix a název jmenného prostoru. Dále obsahuje atributy a jejich hodnoty, které jsou použity v daném elementu. Tato třída také uchovává jaké všechny atributy mohou být použity. Pokud má atribut fixní či defaultní hodnotu je to zjištěno z vnitřního stavu XmlSchemaValidatoru při hledání možných atributů. Tato hodnota je pak předvyplněna uživateli.

6.6 Statistika dokumentu a validace

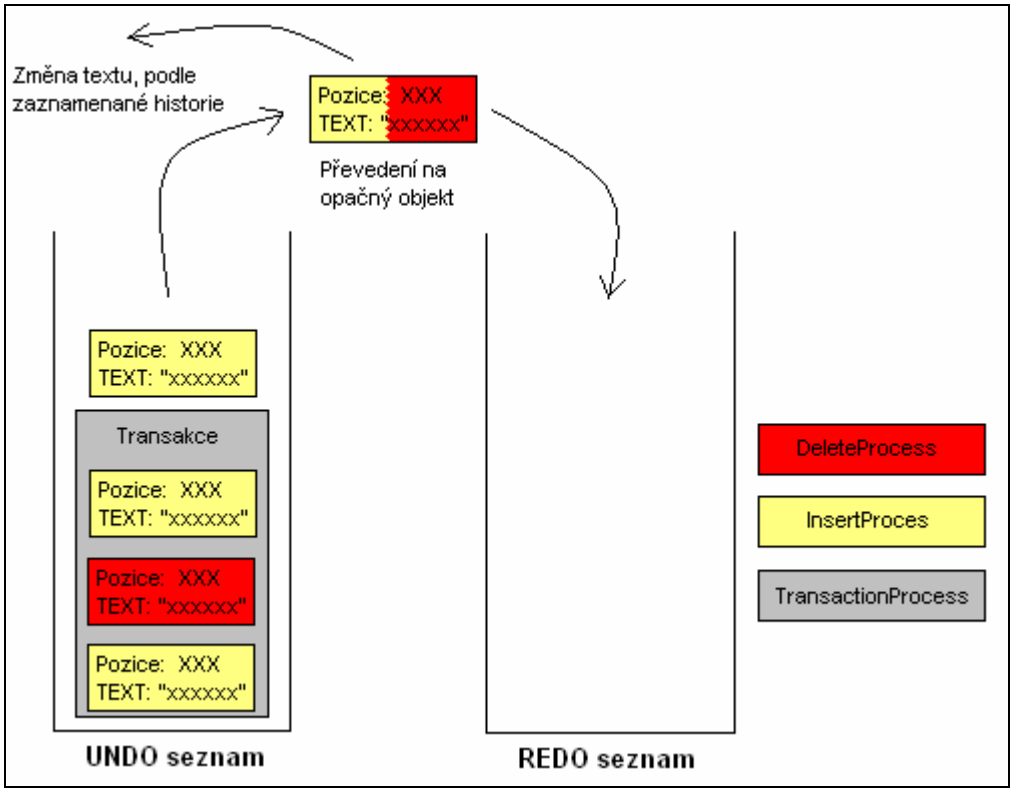
Statistika XML dokumentu je prováděna nástrojem pro provedení statistiky XML Stats. Výstup tohoto nástroje je pouze textový. Proto byla třída modifikována tak, aby z ní mohly být statistické údaje získány strukturovaně. Díky strukturovanému výstupu statistik je lze zobrazit v graficky příjemnější podobě. Pro zobrazení je

využito komponenty Visual Studia PropertyGrid. Pro modifikaci položek PropertyGridu je využito externí knihovny PropertyBag.

Validace dokumentu je zajišťována XML validačním readerem, který vypisuje hlášení o chybách do logu. Validace pro dlouhé čekání při ověřování validity dokumentu běží ve vlastním vlákně. Při nalezení chyby je vypsána pozice chyby a její text. Logovací formulář si pamatuje u každého záznamu pozici chyby. Při dvojkliku na záznam v logovacím formuláři je kurzor přemístěn do správného dokumentu a na správnou pozici tam kde se vyskytla hlášená chyba.

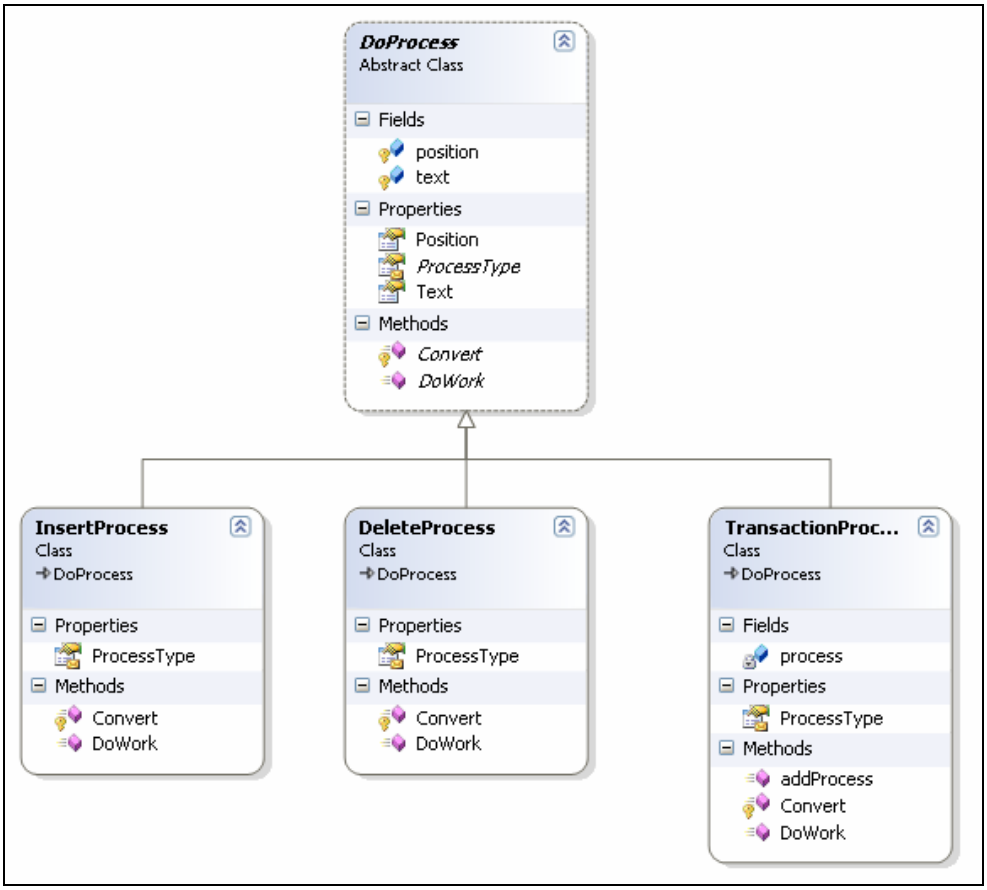
6.7 Zpět a vpřed

Zpět a vpřed neboli undo a redo je zajišťováno pomocí vlastní vytvořené třídy UndoRedoMemory. V této třídě jsou udržovány dva spojové seznamy, ve kterých se uchovávají jednotlivé operace s textem. Jeden spojový seznam pro undo a jeden pro redo, jak je vidět na obrázku 6.7. Seznam pro redo je většinou prázdný.



Obrázek 6.7 : Undo a Redo seznamy

Všechny operace s textem mohou být popsány pomocí vložení a smazání textu. Různé přepisování, připisování a vpisování je realizováno zřetězením těchto operací. Ve spojovém seznamu jsou objekty, které mají společného předka DoProcess. Objekt může být instance jedné ze tří tříd, a to InsertProcess pro vložení, DeleteProcess pro vymazání a TransactionProcess pro zřetězenou operaci přechozích dvou. Instance těchto tříd jsou vytvářeny při editaci XML dokumentu. Při každém vložení či vymazání je vytvořen objekt příslušné třídy a jsou mu předány informace s pozicí, kde změna proběhla, a s textem, kterého se změna týkala.



Obrázek 6.8: Struktura UnDo memory

Při provedení operace zpět je z undo spojového seznamu vytažen objekt s poslední změnou a je na něm spuštěna metoda DoWork. Tato metoda provede buď vymazání textu na uložené pozici a dané délky nebo vložení uloženého textu na danou pozici. Dále se objekt překonvertuje na opačný tedy InsertProcess na DeleteProcess a naopak a je tento objekt uložen do redo spojového seznamu. Pokud je prováděna operace vpřed, pak proces probíhá úplně stejně, jen se objekt DoProcess přesouvá z redo spojového seznamu do undo spojového seznamu. Při vytvoření nového undo

záznamu se spojový seznam redo vymazává. TransactionProcess funguje úplně stejně jako InsertProcess a DeleteProcess jen těchto operací provede více najednou. Na obrázku 6.8 je vidět struktura třídy DoProcess a jejich potomků.

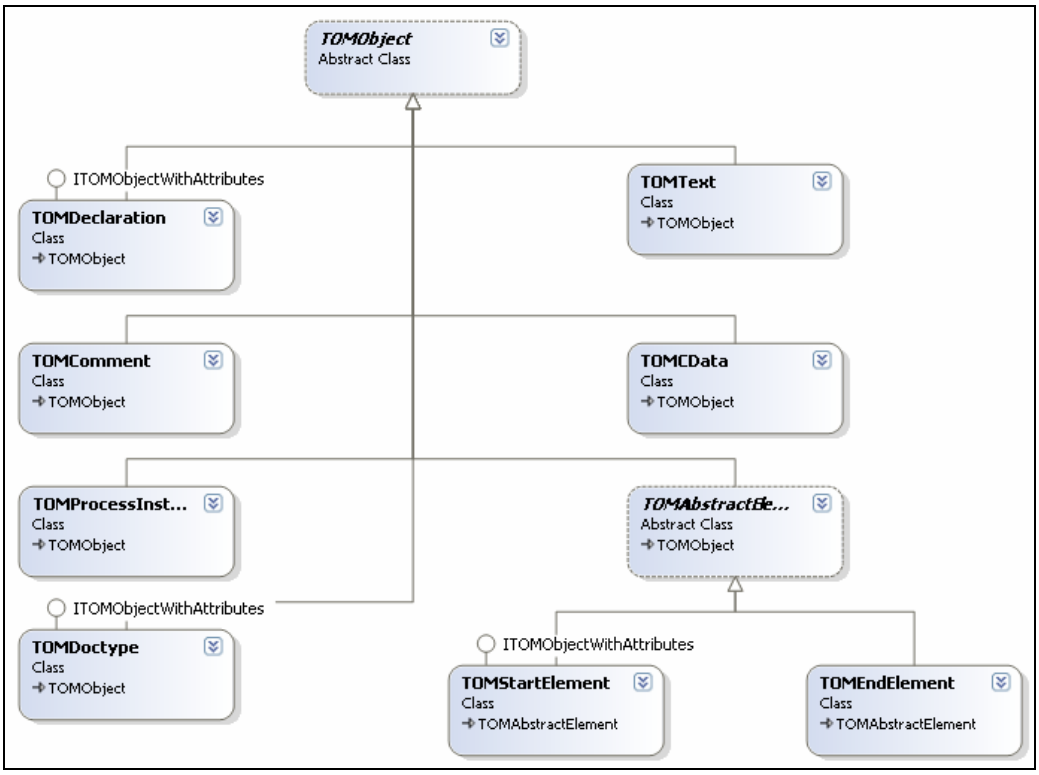
6.8 Zpracování velkých dokumentů

Velký dokument se velice špatně edituje. Především je zde problém s načtením velkého množství dat do paměti. Toto načtení dlouho trvá, protože se data získávají z pevného disku, který má oproti operační paměti milionkrát pomalejší odezvu. Další problém, který se vyskytuje u editace velkých dokumentů je paměť. Dokument může být tak velký, že se do operační paměti nemusí vejít a program může skončit pro nedostatek paměti. S velkým textem má také problém komponenta RichTextBox. Po načtení je totiž text předán RichTextBoxu, který jej zpracovává. Toto zpracování textu je komponentou prováděno neefektivně a zabere příliš času.

Aplikace xmlPad pro editaci velkých dokumentů tedy používá speciálního editoru, kterým není RichTextBox. Byl vytvořen přímo pro účely tohoto XML editoru. Není to typický editor, ale spíše kontejner na další komponenty uživatelského prostředí. Tento kontejner se stará o průběžné načítání uzlů, které se zobrazí. Při pohybu s posuvnou lištou načítá a zobrazuje vždy jen tolik uzlů, kolik je skutečně potřeba.

XML dokument je při otevření analyzován a každý uzel je převeden do nějakého TOM objektu. Například pro komentáře v XML dokumentu je vytvořena instance třídy TOMComment a do ní načten obsah komentáře. Hierarchie tříd je uvedena na obrázku 6.9. Tyto objekty jsou dále ukládány ve spojových seznamech.

Každé třídě z hierarchie TOMObject je přiřazena jedna vizuální komponenta uživatelského rozhraní. Kontejner zobrazuje na každém řádku jednu komponentu uživatelského prostředí, jejíž obsah je vyplněn informacemi z přiřazeného objektu TOMObject. Kontejner, ve kterém je zobrazováno načtené XML, tedy funguje tak, že podle pozice posuvné lišty zažádá objekt třídy, která se stará o načítání XML do TOM objektů, o vydání těchto objektů, ale jen tolika kolik je doopravdy potřeba pro zobrazení. Počet potřebných uzlů pro zobrazení je spočítán z velikosti kontejneru a odpovídá počtu viditelných TOMObjectů do něj vložených.



Obrázek 6.9: Hierarchie třídy TOMObject a potomků

Třída, jež se stará o načítání XML, také rozděluje TOM objekty do spojových seznamů. Pokud je těchto spojových seznamů více, pak se tyto seznamy serializují a uloží na disk, aby zbytečně nezabíraly paměť. Jakmile kontejner požádá o nějaký objekt ze spojového seznamu, který je momentálně serializovaný pak se deserializuje a opět načte do paměti. Nepoužívané spojové seznamy se opět serializují.

7 Srovnání s podobnými programy

XML dokument může být editován jako prostý text, nicméně uživatel bude ochuzen o všechny výhody editování XML dokumentu. V této kapitole jsou analyzovány vybrané editory, se kterými lze XML dokument editovat. Některé zde zmíněné editory neslouží přímo pro editaci XML dokumentu, ale lze s nimi XML dokument editovat.

7.1 Notepad

Notepad [14] nebo-li v české mutaci Poznámkový blok je většinou nainstalován na počítač s instalací operačního systému Microsoft Windows [41]. Pomocí tohoto programu lze editovat jakýkoliv text, který není příliš velký. Jelikož je tento editor zaměřen na editaci prostého textu, neumožňuje žádné zpříjemnění práce uživatele při editaci XML dokumentu. Nepodporuje tedy žádný highlighting ani napovídání možných elementů či atributů. Navíc má aplikace Notepad velký problém při nahrazování řetězce ve velkém textu, kde je větší výskyt nahrazovaného řetězce. Při spuštění funkce nahrazování je velice pomalu procházen text a nahrazován novým řetězcem.

Notepad umožňuje kroky vpřed a vzad, známé jako undo a redo. Navrácení je možné pouze o jeden krok. Uživatel tedy nemá možnost se hlouběji vrátit do historie. Tento editor podporuje funkce jako kopírovat, vložit a vystřihnout. Dále podporuje možnost zalamování řádků na šířku okna editoru. Výhodou programu Notepad je možnost změnit kódování jak při otvírání textu, tak možnost změnit kódování při ukládání dokumentu.

7.2 PSPad

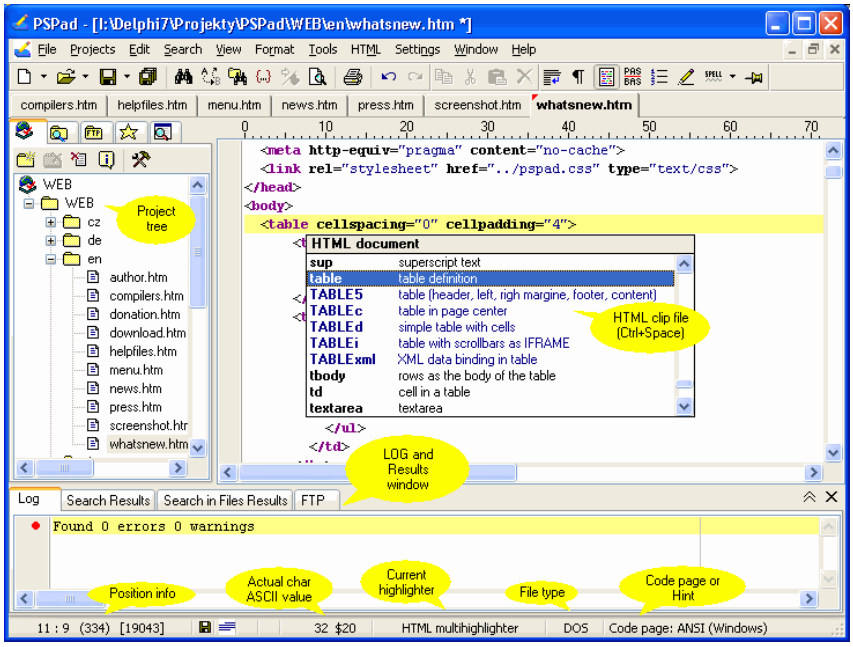
PSPad [35] je víceúčelový editor, ve kterém lze mimo jiné editovat také XML dokumenty. Avšak nejspíš je nejvíce využíván pro editaci webových stránek v jazyku HTML (XHTML). Má všechny základní vlastnosti jiných textových editorů, jako jsou kopírovat, vložit a vystřihnout. Pamatuje si historii úprav textu a

pomocí tlačítek vpřed a vzad lze touto historií procházet. Hloubka historie oproti Notepadu není omezena pouze na jeden krok, uživatel tedy může vrátit více změn.

PSPad podporuje barevné zvýrazňování textu a to nejen XML dokumentu, ale i dalších formátů a jazyků. Dokonce umožňuje uživateli nadefinovat vlastní highlighting. Mezi automaticky vestavěné highlightingy patří zvýrazňování například XML, HTML a programovacích jazyků jako Pascal, C, Java.

Ověřování validity a kontextové menu s nápovědou, které PSPad také obsahuje, není děláno pro obecné XML dokumenty, ale jen pro HTML (XHTML) dokumenty. Pro HTML dokumenty je zde jednoduchý generický formulář. Tento formulář umožňuje vložit element a nadefinovat jeho atributy. PSPad podporuje přepínání znakových sad. V tomto editoru můžete také editovat soubory nacházející se na vzdálených počítačích přes protokol FTP [34].

Na obrázku 7.1 je vidět rozložení oken PSPadu. Vlevo je strom souborů aktuálního projektu. V dolní části je okno s výstupem z Logu a s výsledky hledání. V hlavním editovacím okně je vidět otevřené napovídání pro HTML dokument. Ve stavovém řádku jsou postupně zleva vyobrazeny položky, které ukazují pozici kurzoru v souboru, ASCII hodnotu znaku za kurzorem, aktuální higlighting, typ souboru a znakovou sadu editovaného souboru.



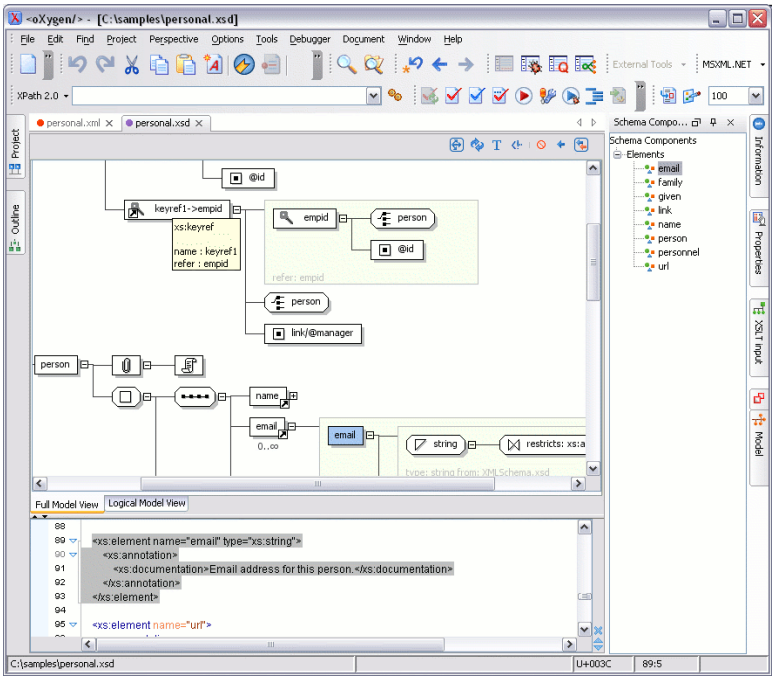
Obrázek 7.1:PSPad

7.3 <oXygen/>

<oXygen/> [36] je velice kvalitní komerční editor XML dokumentů. Tento editor je rozsáhlý. Umí samozřejmě všechny obvyklé funkce jako jsou kopírovat, vložit a vystřihnout. Podporuje pohyb v historii úprav textu směrem vpřed i vzad, a to do hloubky větší než jedna.

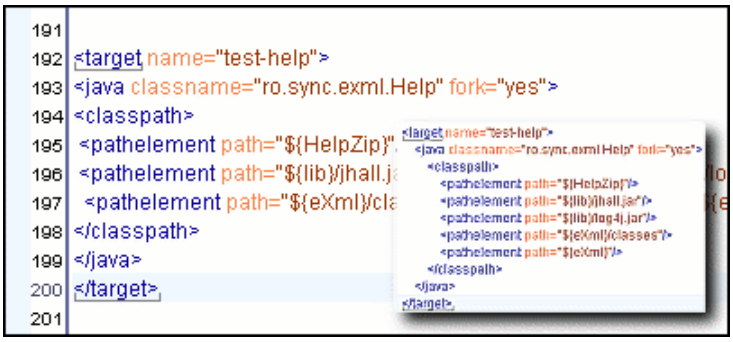
Mezi pokročilejší funkce zaměřené na XML a práci s ním patří následující. Umí se nad dokumentem dotazovat v jazyku XPath 1.0 i XPath 2.0. Pro dotazování nad XML daty má také podporu XQuery. Dotazy XQuery umí krokovat pomocí XQuery debugeru. Umí transformovat XML dokument pomocí XSL šablony. Tuto transformaci, si může uživatel také krokovat a odladit. Obsahuje také navigační strom, pomocí kterého může uživatel přeskakovat mezi elementy.

Program dále umí vygenerovat DTD k danému XML dokumentu. Samozřejmě podporuje ověření validity XML dokumentu proti DTD či schématu v XML Schema. Pro editaci schématu v jazyce XML Schema, editor zobrazí takzvané model view (obr. 7.2), které zobrazí XML dokument ve stromové struktuře, takže je vidět přesné rozložení elementů, atributů a vlastností. Pomocí tohoto model view se může uživatel pohybovat v XML dokumentu.



Obrázek 7.2 : Model view

<oXygen/> umí ověřit validitu XML dokumentu vůči DTD či schématu v jazyku XML Schema. Validace také probíhá rovnou při psaní a podtrhává chybně napsané elementy a atributy nebo pokud v elementu něco chybí, či přebývá. Také zvýrazňuje začáteční a uzavírací element, patřící k sobě, tím, že je oba podtrhne. Na obrázku 7.3 je vidět část editovací oblasti aplikace. Tento editor je velice rozsáhlý a podporuje mnoho dalších funkcí jako je SOAP analyser pro tvorbu a ladění SOAP dotazů, XPath builder pro tvoření dotazů v jazyku XPath, atd. Jediná nevýhoda tohoto programu je, že je komerční. Cena Professional verze je $200.



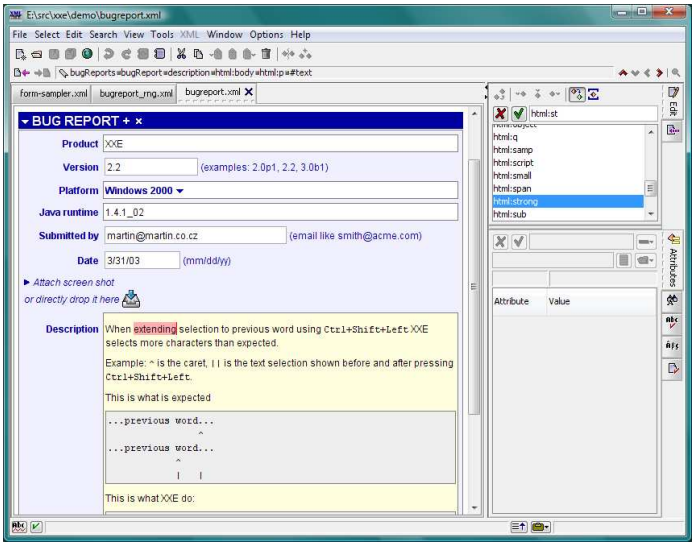
Obrázek 7.3: <oxygen/>

7.4 XMLmind XML Editor

XMLmind [37] editor se od předešlých liší svým přístupem k XML dokumentu. Dokument v tomto editoru je zobrazen jako strom, každý element je zde jako objekt, má svůj obsah a své atributy. Podporuje transformaci s použitím šablon XSL. Výsledek po transformaci je zobrazen jako webová stránka v HTML, takže se uživatel může přepínat mezi různými zobrazeními daného dokumentu.

Tento editor také umí ověřovat validitu dokument vůči DTD nebo schématu v XML Schema. U aktuálně editovaného elementu podporuje zobrazování atributů ve vedlejším okně. Pokud chce uživatel přidávat do dokumentu elementy, musí kliknout na element a poté může vložit nový element před nebo za aktuální element. Následuje výběr elementu, a tím se do těla dokumentu vybraný element vloží. V případě, že chce uživatel editovat atributy, je zde pro tento účel již zmíněné okno atributů, kde si nastaví atributy podle potřeby. Tímto editorem lze editovat pouze XML dokumenty, a to správně uzávorkované. Pokud se pokusíte načíst tímto

editorem soubor, ve kterém někde chybí uzavírací závorka nebo není správně zformovaný, pak je při otevírání nahlášena chyba, kde editor skončil s analýzou. Okno XMLmind editoru je uvedeno na obrázku 7.4.



Obrázek 7.4:XMLmind

8 Závěr

Cílem této práce bylo vytvořit víceúčelový XML editor xmlPad. Vytvořená aplikace umí editovat XML dokumenty a pokud je tento XML dokument malý, pak uživatele nijak neomezuje v psaní textu. Může tedy vytvářet nevalidní, ale i špatně uzávorkovaný XML dokument. To je důležité hlavně pro případy, kdyby uživatel chtěl editovat například dokument, který není XML dokumentem a on jej bude muset na XML dokument upravit.

Další velkou výhodou programu xmlPad je, že dokáže editovat velké soubory. Při načítání takovéhoto dokumentu je určitá prodleva. Nicméně editace již načteného XML dokumentu už funguje v pořádku. Načtení 25MB XML dokumentu na počítači s procesorem Intel Centrino Duo 1,8GHz a 1GB RAM trvá přibližně 15 vteřin. Na tomtéž počítači otevření 100MB XML dokumentu trvá 50 vteřin.

Aplikace tedy umožňuje editaci XML dokumentu jak malých, kde nijak neomezuje uživatele, tak velkých kde na úkor velikosti je omezena volnost uživatele a je nutné otevírat pouze správně strukturované XML dokumenty.

Pro větší přehlednost je v aplikaci xmlPad použito barevného zvýrazňování, takzvaného highlightingu. Uživatel tak rychle a na první pohled pozná kde začíná a končí daný element, jaké má atributy nebo kde je komentovaná část. Všechny barvy použité pro zvýrazňování mohou být uživatelem nastaveny v nastavení aplikace. Dalším pomocníkem při vytváření či editaci dokumentu je kontextová nápověda. Tato nápověda využívá schémat DTD nebo v jazyce XML Schema, aby mohla uživateli napovědět jaké se mohou elementy nebo atributy vyskytnout na aktuální pozici v editoru.

Pomocí generického formuláře je uživatel automaticky veden vytvářením XML dokumentu. Může tak vytvořit XML dokument bez námahy psaní elementů či atributů. Pouze vyplní textové hodnoty a hodnoty atributů, ostatní si pomocí kurzoru vybere a generický formulář vše obstará.

Aplikace xmlPad dále obsahuje funkce pro kopírování, vkládání a vystřihování textu. Umí vyhledávat zadaný řetězec a nejen tak jak je běžně známé, ale umí rozlišit zda je nalezený text v názvu elementu, v názvu atributu, v hodnotě atributu či to je textový obsah elementu. Díky této nadstavbě vyhledávání si může uživatel vybrat a upřesnit, kde má být vyhledávaný řetězec hledán. Pro možnost oprav uživatele jsou dále v aplikaci vytvořeny funkce pro procházení historií úprav a změn v textu. Uživatel může vracet provedené operace a poté je znovu vyvolat.

Pro souhrné informace o XML dokumentu slouží XML statistika, která je uživateli vygenerována. Uživatel může přetvářet schémata v DTD do schémat v jazyce XML Schema. Schémata v XML Schema mohou být editována a kontextový editor zobrazuje nápovědu možných elementů a atributů podle příslušné normy jazyka XML Schema. Tato funkce velice usnadní uživateli, jenž není zcela obeznámen s tímto jazykem, jeho editaci či vytvoření nového schématu.

8.1 Možnosti rozšíření

V dalších verzích tohoto programu by mohla přibýt funkce na vygenerování DTD podle XML dokumentu. Dále by mohl být editor rozšířen o možnost zobrazení a editování XML dokumentu jako DOM stromu. Každý element by byl objekt a celý dokument by byl zobrazen jako strom těchto objektů. Další rozšíření by mohlo být zobrazení XML dokumentu jako HTML stránky, pokud tento dokument bude XHTML nebo pokud budou použity nějaké šablony XSL.

9 Zdroje

[1] RTF specifikace,
http://msdn2.microsoft.com/en-us/library/aa140277(office.10).aspx

[2] XML verze 1.0 specifikace, http://www.w3.org/TR/1998/REC-xml-19980210

[3] HTTP protokol, http://www.w3.org/Protocols/HTTP/HTTP2.html

[4] XML Schema verze 1.1 specifikace http://www.w3.org/TR/xmlschema11-2/

[5] DTD specifikace,
http://www.w3.org/XML/1998/06/xmlspec-report-19980910.htm

[6] Mlýnková, I. - Pokorný, J. - Richta, K. - Toman, K. - Toman, V.: Technologie XML. Skripta. Karlova Univerzita, Praha, Česká republika, září 2006

[7] ISO 10646, http://cs.wikipedia.org/wiki/ISO_10646

[8] UTF-8, http://www.rfc-editor.org/rfc/rfc3629.txt

[9] ASCII, http://cs.wikipedia.org/wiki/ASCII

[10] ISO-8859-2, http://cs.wikipedia.org/wiki/ISO_8859-2

[11] Windows-1250, http://cs.wikipedia.org/wiki/Windows-1250

[12] HTML verze 4.01, http://www.w3.org/TR/html401/

[13] XHTML verze 1.0, http://www.w3.org/TR/xhtml1/

[14] Notepad,

[15] XSL, http://www.w3.org/TR/xsl/

[16] WEB SERVICES, http://www.w3.org/TR/ws-arch/

[17] SOAP, http://www.w3.org/TR/soap/

[18] WSDL, http://www.w3.org/TR/wsdl

[19] SQL, http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt

[20] XQuery verze 1.0, http://www.w3.org/TR/XQuery/

[21] XPath, http://www.w3.org/TR/xpath

[22] Prezentace k přednáškám Technologie XML,
http://kocour.ms.mff.cuni.cz/~mlynkova/prg036/

[23] Esposito D.: *XML – efektivní programování pro .NET*, nakladatelství GRADA, 2004, ISBN: 80-247-0775-6

[24] .NET, http://www.microsoft.com/net/default.aspx

[25] SAX, http://xmleverywhere.com/wrox/3110/31100602.htm

[26] MDI, http://en.wikipedia.org/wiki/Multiple_document_interface

[27] VS 2005, http://msdn2.microsoft.com/en-us/vstudio/default.aspx

[28] DockManager 1.0.0.0, Weifen Luo, http://www.codeproject.com/cs/miscctrl/DockManager.asp?df=100&forumid=16526&exp=0&select=1666962

[29] XML stat, Chris Lovett, Andreas Lang http://www.lovettsoftware.com/tools/xmlstats/readme.htm

[30] Property bag, Tony Allowatt, 2002, http://www.codeproject.com/cs/miscctrl/bending_property.asp

[31] Print text document, Alberto Ferrazzoli, 2004 http://www.codeproject.com/csharp/printtextdocument.asp

[32] Regulární výrazy, http://www.regular-expressions.info/reference.html

[33] Windows API, http://msdn2.microsoft.com/en-us/library/aa383750.aspx

[34] FTP, http://www.ietf.org/rfc/rfc0959.txt

[35] PSPad, http://www.pspad.com/cz/

[36] <oxygen/>, http://www.oxygenxml.com/

[37] XMLmind, http://www.xmlmind.com/xmleditor/

[38] Sells, C. : *C# a WinForms – programování formulářů Windows*, Addison-Wesley, Zoner Press Brno 2005, ISBN : 80-86815-25-0

[39] Jmenné prostory, http://www.w3.org/TR/REC-xml-names/

[40] XML DOM, http://www.w3schools.com/dom/default.asp

[41] Microsoft Windows, http://www.microsoft.com/windows/default.mspx

[42] DTD2XSD, http://www.gotdotnet.com/community/usersamples/details.aspx?sampleguid=54358b80-1324-49e9-821b-a08911356ad7

[43] programovací jazyk C#, http://msdn2.microsoft.com/en-us/vcsharp/default.aspx

[44] přednáška Automaty a gramatiky, http://kti.mff.cuni.cz/~bartak/automaty/prednaska.html

10 Příloha

K bakalářské práci je přiloženo CD s instalačním balíčkem aplikace xmlPad, zdrojovým kódem této aplikace a elektronickou verzí tohoto textu.

Obsahuje adresáře

- Source obsahuje projekt ve Visual Studiu 2005, zdrojové soubory mají příponu *.cs
- Instalace obsahuje instalační balíček
- Text obsahuje tento text v elektronické verzi