VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

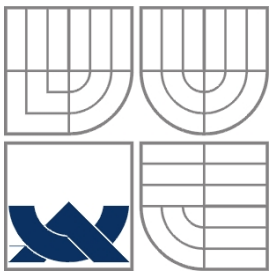
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INTERAKTIVNÍ VIZUALIZACE XML

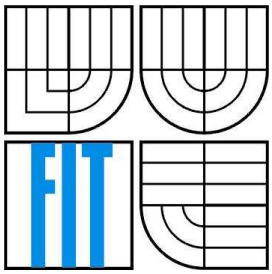
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MARTIN SEKO
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INTERAKTIVNÍ VIZUALIZACE XML

INTERACTIVE VISUALIZATION OF XML

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN SEKO

VEDOUCÍ PRÁCE
SUPERVISOR

ING. PETR CHMELAŘ

BRNO 2009

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2008/2009

Zadání bakalářské práce

Řešitel: **Seko Martin**
Obor: Informační technologie
Téma: **Interaktivní vizualizace XML**
Kategorie: Databáze

Pokyny:

1. Seznamte se s problematikou XML a jeho definicemi.
2. Identifikujte a pokuste se vyřešit problémy interaktivního zobrazení XML dat včetně jejich definice.
3. Vytvořte grafický nástroj pro zobrazení a modelování libovolných datově orientovaných dokumentů XML.
4. Zhodnoťte vlastnosti a případné vylepšení nástroje.

Literatura:

- Quin, L. W3C. *Extensible Markup Language (XML)*. 2006. Dostupný z: http://www.w3.org/XML/.
- Oracle Corporation. *Oracle Technology Network*. 2006. Dostupný z: http://www.oracle.com/technology/.
- Herout, P. *Učebnice jazyka Java.* České Budějovice: Kopp, 2007. 381 s. ISBN 978-80-7232-323-4.
- Jelinek, J. - Slavik, P. XML visualization using tree rewriting**.** *Proceedings of the 20th spring conference on Computer graphics*. ACM, 2004. pp. 65-72. ISBN:1-58113-967-5.

Při obhajobě semestrální části projektu je požadováno:

- 1. a 2. bod zadání.

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

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

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

Vedoucí: **Chmelař Petr, Ing.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2008
Datum odevzdání: 20. května 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Hlavným cieľom tejto bakalárskej práce je riešenie problému interaktívnej vizualizácie dátovo orientovaných XML dokumentov a súčasne implementácia danej aplikácie, ktorá dokáže zobraziť akýkoľvek dokument tohto typu. Aplikácia z otvoreného XML dokumentu generuje schému v štyroch formátoch, a to XML schema, DTD, RNG a RNC. Taktiež dokáže vizualizovat tieto dáta, a to či už formou stromu alebo tabuliek.
Výsledkom tejto bakalárskej práce je aplikácia XMLspark, ktorá je implementovaná ako multiplatformná desktop aplikácia v prostredí Java s grafickým rozhraním Swing.

Abstract

The main objective of this bachelor's thesis is to create an interactive visualization of data-oriented XML documents along with an implementation of such an application which is capable of displaying a document of this datatype. Application can generate schemas from an opened document in four supported languages: XML schema, DTD, RNG and RNC. It can also visually display this data in either tree or table format.
Result of this bachelor's thesis is an application XMLspark, which is implemented as a multiplatform desktop application in the Java environment with the Swing GUI.

Klíčová slova

interaktivní vizualizace XML, generátor tabulek z XML, generátor stromu z XML, generátor XML schéma, XSD, RNG, RNC, DTD, XML, Relax NG, XMLspark

Keywords

interactive visualization of XML, XML table generator, XML tree generator, XML schema generator, XSD, RNG, RNC, DTD, XML, Relax NG, XMLspark

Citácia

Seko Martin: Interaktívna vizualizácia XML, bakalárska práca, Brno, FIT VUT v Brně, 2009

Interaktívna vizualizácia XML

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Petra Chmelaře.
Ďalšie informácie mi poskytli Bc. Ivan Kramárik a Ing. Petr Jambor.
Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

……
Martin Seko
20. mája 2009

Poďakovanie

Týmto by som sa chcel poďakovať predovšetkým svojmu odbornému konzultantovi Ing. Petrovi Chmelařovi za odbornú pomoc pri zostavovaní celej aplikácie a tejto technickej správy. Ďalej by som sa chcel poďakovať Bc. Ivanovi Kramárikovi za poskytnuté testovacie dáta a Bc. Martinovi Gáborovi za testovanie a spätnú väzbu v oblasti grafického rozhrania.

Obsah

1 Úvod

Žijeme v dobe, kde náš svet obklopujú počítače. Či už priamo alebo nepriamo, môžeme povedať, že náš život od nich závisí. Bezpečnostné systémy zabezpečujúce chod elektrární, ktoré dodávajú elektrinu nám domov alebo firmám, ktoré pre nás, ľudí, produkujú buď potraviny alebo iný tovar potrebný pre život. Počítače, ako aj všetka výpočtová technika však majú niečo spoločné – výpočty uľahčujúce každodennú činnosť. Čo je však úzko spojené s počítaním? Načo sú nám výpočty ak ich nemáme z čoho riešiť? Áno, hlavnou alfou a omegou celého nášho sveta sú informácie.

Informácie, ktoré získavame našimi zmyslami – očami, ušami, prstami, chuťou alebo čuchom, môžeme tvrdiť, že aj toto sú informácie ktoré máme k dispozícii. Avšak prenesme sa späť do sveta výpočtovej techniky, kde má slovo informácia v podstate rovnaký význam ale predsa len sa nejako odlišuje.

Dáta – nuly a jednotky, tak ako ich počítač reprezentuje by boli pre nás úplne nepochopiteľné. Preto ľudstvo vytvorilo rôznorodé programy, aby obyčajní užívatelia boli schopní tieto údaje pochopiť a vnímať ich tak ako ich dotyčný človek pred nimi uložil. Existuje nespočetné množstvo typov informácii, ktoré sú zapuzdrené pod jedným rúškom – počítačom.

Vo svojom laptope alebo stolovom počítači určite nájdete nejaké video, textový dokument, hudobnú skladbu, tabuľkový dokument, hru a podobne. Tieto všetky veci, obsahujú dáta, no však bez príslušného programu nie sme schopní vidieť, čo daný súbor obsahuje. A tu prichádzame na koreň problému.

Reprezentácia dát každým jednotlivým programom je zväčša úplne rozdielna. Len si porovnajte textový dokument typu .txt a .doc. V prvom budete vidieť čo je obsahom, no však ak bude textu veľa, budeme sa strácať v štruktúre. Dokument programu MS Word je pekne naformátovaný, dobre čitateľný, no bez tohto programu nemáme šancu daný dokument prečítať. Preto sa snažíme dospieť k jednotnému ukladaniu všetkých informácii, ktoré na počítačoch máme, no nie vždy je to možné. Taktiež vzniká potreba prenášať informácie medzi jednotlivými aplikáciami, no bez konkrétneho prenosového formátu sa znova dostaneme do Pandorinej skrinky.

Jedným z prípadov zjednocovania dát je aj jazyk XML (*eXtensible Markup Language*, slovensky: rozšíriteľný značkovací jazyk). Bol navrhnutý ako univerzálny formát pre uloženie hierarchických a tabuľkových dát. Postupne sa však stáva aj zástupcom, poprípade i náhradou iných jazykov, kvôli jeho jednoznačnosti a prísnej syntaxi. V súčasnosti mnoho aplikácii využíva jazyk tento značkovací jazyk na prenos najmä externých dát ako napríklad konfiguračné súbory, tabuliek s rôznorodými dátami, web stránok, plánmi výroby, záznam cesty zo sledovača GPS, formátovanie dokumentov a podobne.

Vďaka širokému spektru typov informácii, ktoré môže XML v súčasnosti reprezentovať sa dostávame k problémom pri zobrazovaní údajov obsiahnutých v dokumente. Z toho logicky vyplýva,

že aj tvorba aplikácie na zobrazovanie dát v xml súbore bude niesť so sebou množstvo problémov. Nami otvorený dokument je text, či tabuľky obsahujúce vyextrahované informácie z databázy alebo je to konfiguračný súbor nejakého programu?

V tejto práci sa budem zaoberať práve problémom zobrazenia dát v XML dokumentoch pre obyčajných užívateľov, ktorí nemajú príliš veľkú znalosť XML a jeho rozmanitosti. Sústredím sa hlavne na dátovo orientované dokumenty. Tie sa väčšinou dajú zobraziť ako tabuľky, na rozdiel od textovo orientovaných dokumentov, ktoré by sa dali zobraziť ako tabuľky veľmi ťažko, poprípade vôbec.

1.1 Štruktúra práce

V teoretickej časti práce čitateľa bližšie oboznámim so samotným jazykom XML, jeho syntaxou, históriou, využitím a významom. Samozrejme opomeniem schémy XML dokumentov, ktoré sú takmer neoddeliteľnou súčasťou samotného jazyka. Keďže celá aplikácia je implementovaná v programovacom jazyku Java, zmienim sa aj o ňom, jeho výhodách a nevýhodách, potrebných knižniciach pre prácu s XML a tak ďalej.

V nasledujúcej kapitole budem pojednávať o návrhu jednotlivých častí aplikácie. Konkrétne okomentujem zobrazenie dokumentu, potom návrh zobrazenia a generovania schémy daného dokumentu a možnosti zobrazenia výslednou tabuľkovou alebo stromovou reprezentáciou alebo inými prostriedkami.

Implementačná časť práce obsahuje popis funkčnosti jednotlivých častí aplikácie. Popíšem prácu so samotným dokumentom, potom schémou a predstavím implementáciu už samotnej vizualizácie.

Kapitola obsahujúca testovacie výsledky priblíži chyby nájdene počas testovania aplikácie tretími osobami. V závere zhrniem prínos a výsledky tejto práce a v poslednej časti je možno nájsť príručku na ovládanie aplikácie a obsah priloženého CD nosiča.

2 Teória

Táto kapitola pojednáva o základoch jednotlivých použitých technológií v pri implementácii aplikácie. Približuje problematiku jazyka XML, históriu jeho vzniku, syntax a význam. Ďalej pojednáva o reprezentácii schémy dokumentu pomocou rôznych technológii. Zaoberá sa aj samostatným programovacím jazykom Java a jeho prostriedkami na implementáciu práce s XML dokumentmi a schémami.

2.1 XML

V súčasnej dobe sa o XML (*eXtensible Markup Language*) hovorí predovšetkým v súvislosti s web stránkami a považuje sa za nástupcu dnes používaného jazyka HTML (*HyperText Markup Language*). Výhoda XML spočíva hlavne v tom, že autor stránky môže používať vlastné tagy, ktoré dokážu omnoho presnejšie označiť význam reprezentovaných informácii, čo má veľký význam pri vyhľadávaní informácii [1].

Jazyk sám o sebe je určený najmä na výmenu dát medzi aplikáciami a na publikovanie dokumentov. Dokumentmi v tom zmysle rozumieme nielen formátovaný text alebo webovú stránku, ale aj rôzne dátové štruktúry ako napríklad tabuľky, grafy, súbor s popisom správania programu a podobne. Jazyk XML umožňuje popísať štruktúru dokumentu z hľadiska vecného obsahu jednotlivých častí, nezaoberá sa sám od seba vzhľadom dokumentu alebo jeho častí. Tento jazyk je založený na jednoduchom texte a v prípade potreby je spracovateľný ľubovoľným textovým editorom bez sebe menších problémov.

XML hneď od začiatku počítal s potrebami aj iných jazykov ako je angličtina, preto sa ako znaková sada implicitne používa ISO 10646 (tzv. Unicode). Špecifikácia XML konzorcia W3C [12] je zdarma prístupná každému. Hocikto teda môže bez problémov do svojej aplikácie implementovať podporu XML. To je veľký rozdiel oproti firemným formátom, ku ktorým nie je takmer žiadna dokumentácia a navyše sa jedná o veľmi zložité, často binárne, formáty na rozdiel od XML [2].

2.1.1 História

História jazyka XML siaha až do šesťdesiatych rokov dvadsiateho storočia, kedy firma IBM riešila problém s ukladaním veľkého množstva právnych dokumentov. Bolo treba vymyslieť formát , ktorý by mal dlhú životnosť a nebol by závislý na používaných programoch atd. Výsledkom bol všeobecný značkovací jazyk, ktorý sa v praktickom nasadení osvedčil. Jazyk sa postupne rozširoval, až sa z neho v roku 1986 stal jazyk SGML (Standard Generalized Markup Language), ktorý bol prijatý za normu ISO. SGM bol veľmi flexibilný, aby vyhovel rôznym požiadavkom [5].

Prvou masovo použitou aplikáciou SGML bol jazyk HTML. Spočiatku mal HTML malý počet tagov, no výrobcovia prehliadačov si ho postupne prispôsobovali a rozširovali, aby vyhoveli užívateľom. Z toho postupne vznikali problémy s kompatibilitou, kvôli rôznorodosti špecifických zápisov pre každý prehliadač. Konzorcium W3C [12] sa snažilo takémuto vývoju zabrániť, postupným rozširovaním štandardu, ale čoskoro to začalo byť neúnosné, a dospelo sa k záveru, že je nutné vytvoriť novú technológiu.

Výsledkom niekoľko ročného snaženia predných odborníkov bol jazyk XML, pričom jeho finálna verzia bola publikovaná v roku 1998. Jazyk XML vznikol ako zjednodušená podmnožina SGML. Tým sa odstránil najväčší problém SGML, a to prílišná zložitosť. Flexibilita tak ostala zachovaná a XML sa stáva jedným z popredných značkovacích jazykov [3].

2.1.2 Syntax XML

Na rozdiel od HTML, syntax jazyka XML musí splňovať omnoho prísnejšie pravidlá. Pokiaľ dokument splňuje všetky tieto pravidlá hovoríme, že daný dokument je správne štruktúrovaný (*well-formed*). S takýmto dokumentom sú potom aplikácie schopné pracovať a pomocou tzv. *parseru* sprístupniť aplikácii [1]. Aby bol dokument validný v XML jazyku musí mať minimálne tieto vlastnosti:

- Musí mať práve jeden koreňový (root) element.
- Neprázdne elementy musia byť ohraničené štartovacou `<tag>` a koncovou `</tag>` značkou. Prázdne elementy môžu byť označené značkou „prázdny element“ `<tag/>`.
- Všetky hodnoty atribútov musia byť uzavreté v úvodzovkách, a to buď v jednoduchých (‘) alebo dvojitých (“). Tieto úvodzovky nemožno kombinovať.
- Elementy sa môžu zanorovať, nesmú sa však prekrývať, t.j. každý element musí byť celý obsiahnutý v inom elemente. Napríklad:

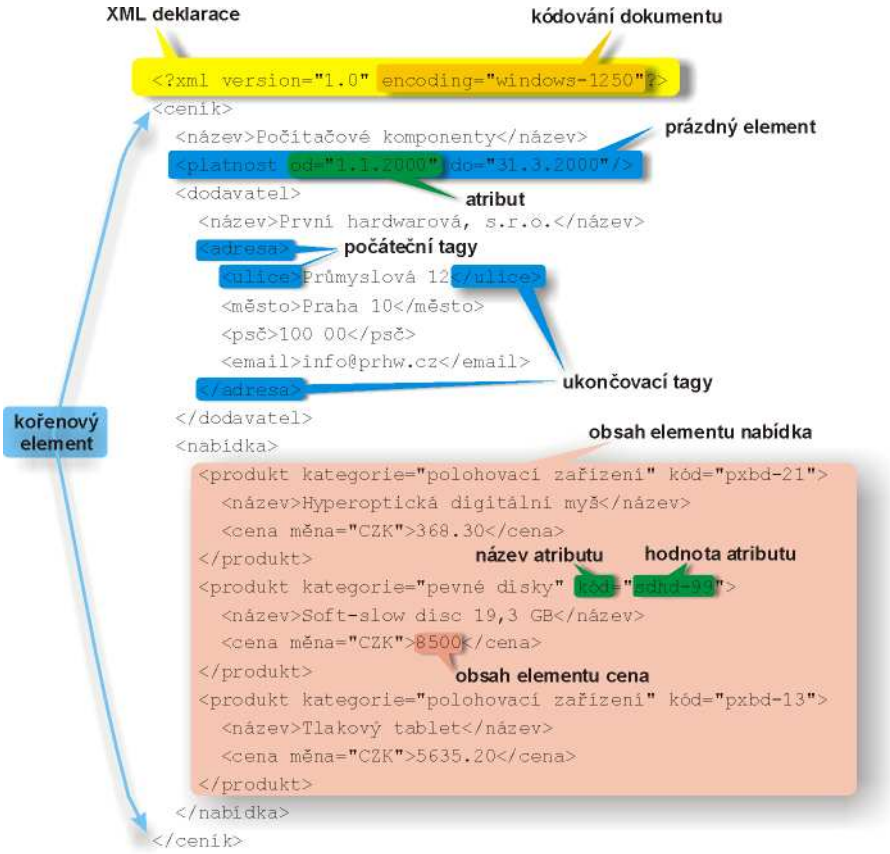
```
<element1>
   <element2>
   </element2>
</element1>
```

Mená elementov rozlišujú malé a veľké písmená abecedy, jazyk XML je teda *case-sensitive*. Dokument XML je text, vždy kódovaný podľa normy ISO 10646 (Unicode), v Českej republike zväčša ako UTF-8, ale sú povolené aj iné kódovania. Informácie o kódovaní dokumentu sú vždy obsiahnuté hneď na začiatku, spolu s verziou XML [2]. Na začiatok dokumentu môžeme poprípade vložiť náš komentár alebo určiť typ dokumentu. Tento typ určíme v *deklarácii typu dokumentu* – `DOCTYPE` (viď obr. 2.1). Typom môže byt napríklad webová stránka, list známemu, databáza kníh a podobne.

Definícia typu dokumentu je potom uložená v osobitnom súbore, v našom prípade je to súbor dopis.dtd, ktorý slúži v podstate ako schéma (viď kapitola 2.1.4) pre daný dokument, určujúca celkovú štruktúru, elementy a atribúty, ktoré môžeme alebo musíme v danom dokumente použiť [1].

```
<?xml version="1.0"?>
<!DOCTYPE dopis SYSTEM "dopis.dtd">
<dopis>
  <adresát>
    <jméno>Jan Novák</jméno>
    <adresa>...</adresa>
    ...
  </adresát>
  <předmět>Pozvánka na besedu o XML</předmět>
  <tělo>
    <para>Vážený pane,</para>
    <para>rád bych Vás jménem naší...</para>
    ...
  </tělo>
</dopis>
```

Obrázok 2.1: Dokument s deklaráciou typu dokumentu (prevzaté z [1])



Obrázok 2.2: Ukážka a popis syntaxe jazyka XML (prevzaté z [4])

2.1.3 Význam a využitie XML

XML slúži na identifikáciu – popis obsahu (vrátane jeho štruktúry) dokumentu. Význam XML vyplýva už samotného zadania konzorcia W3C [12], ktoré toto konzorcium dalo skupine vývojárov jazyka. Jeho vlastnosťami sú teda:

- XML je využiteľné v Internete
- podporuje širokú škálu aplikácii
- umožňuje jednoduché písanie programov, ktoré pracujú s dokumentmi napísaných v XML
- dokumenty napísané v jazyku XML sú zrozumiteľné a čitateľné aj pre ľudí
- tvorba samotného dokumentu typu XML je jednoduchá.

Prínos uplatnenia XML je jasný už od jeho vzniku. Informácie obsiahnuté v dokumente sú čitateľné natrvalo a sú nezávislé na dátovom formáte. Revolúciu priniesol aj do vyhľadávania a to tým, že je informácie možno hľadať nie len podľa obsahu ale aj na základe ich významu. Nakoľko dáta možno nielen vyhľadávať ale aj vytvárať, XML má preto obrovský význam pre *publishing* – vystavovanie dokumentov podľa typov formátov (XSL, DTD atd.) [6]. Aplikovanie XML do súčasných technológii naberá pomerne rýchly spád. Dobrými príkladmi terajšieho využitia jazyka XML sú:

- XHTML – nástupca jazyka HTML
- RSS – rodina XML formátov slúžiacich na čítanie noviniek (*news*) na webových stránkach
- SMIL (*Synchronized Multimedia Integration Language*) – popis multimédií pomocou XML
- SVG (*Scalable Vector Graphics*) – jazyk na popis dvojrozmernej vektorovej grafiky
- DocBook – sada definícii dokumentov a štýlov pre publikačnú činnosť
- Jabber – protokol na *Instant messaging*
- SOAP – protokol na komunikáciu medzi webovými službami [2].

2.1.4 Schéma XML dokumentu

Napriek tomu, že XML dokumenty môžu mať ľubovoľnú štruktúru, veľmi skoro zisťujeme ako rýchlo by sme dospeli do pôvodného stavu – chaosu. Preto vzniká potreba určiť nejakú, predom definovanú, podobu nášmu dokumentu. Táto podoba sa nazýva schéma XML dokumentu. Určujeme ňou ako sa môžu jednotlivé elementy nazývať, ich postupnosť, obsah a taktiež to, či sú povinné alebo sa nemusia v dokumente vôbec vyskytnúť. Schéma dokumentu XML teda plní podobnú funkciu ako schéma databázy vo svete relačných databázových systémov [7].

Schémy dokumentov nie sú nič iné len znova ďalšie značkovacie jazyky definujúce formát akým má byť dokument napísaný. Niektoré jazyky na popis schémy dokážu navyše určovať aj dátový typ obsahu elementov alebo atribútov ako napríklad číslo, reťazec, dátum a podobne.

Primárne sa schémy využívajú predovšetkým na *formálnu definíciu* značkovacích jazykov založených na XML. Výhoda formalizovanej definície spočíva v tom, že všetci, čo chcú dokumenty písať alebo spracovávať vedia ako budú súbory vyzerať. Keďže vieme aký má dokument formát, dokážeme ho aj validovať, teda zistíme či je správne napísaný alebo nie. Počas validácie samozrejme môžeme kontrolovať aj typy elementov, pokiaľ to však schéma obsahuje. Historicky najstarším jazykom na popis XML dokumentu je DTD (*Document Type Definition*). Neskôr sa objavili aj ďalšie jazyky ako napríklad XML schema alebo Relax NG.

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
   <book id="bk101">
      <author>Gambardella, Matthew</author>
      <title>XML Developer's Guide</title>
      <genre>Computer</genre>
      <price>44.95</price>
      <publish_date>2000-10-01</publish_date>
      <description>An in-depth look at creating applications
      with XML.</description>
   </book>
   <book id="bk102">
      <author>Ralls, Kim</author>
      <title>Midnight Rain</title>
      <genre>Fantasy</genre>
      <price>5.95</price>
      <publish_date>2000-12-16</publish_date>
      <description>A former architect battles corporate zombies,
      an evil sorceress, and her own childhood to become queen
      of the world.</description>
   </book>
</catalog>
```

Obrázok 2.3: XML dokument catalog.xml

2.1.4.1 DTD

Priamo špecifikácia jazyka XML popisuje DTD, čo je vlastne jazyk na popis schémy dokumentu pochádzajúci ešte z jazyka SGML. DTD vďaka tomu podporuje veľké množstvo aplikácii no však má veľa nevýhod. Prvou jeho nevýhodou je neschopnosť podpory menných priestorov (*namespaces*). Menné priestory sú mechanizmus, ktorý umožňuje v jednom dokumente kombinovať viacej značiek, napríklad do webovej stránky môžeme vložiť vektorový obrázok v SVG a podobne. Druhým nedostatkom je neschopnosť určiť dátový typ [7]. Tretím dôvodom bola syntax DTD, ktorá sa nikde v XML nepoužíva a teda zápis niektorých schém bolo až príliš ťažkopádne. Príklad DTD môžeme vidieť na obrázku 2.4. DTD súbory majú logicky príponu .dtd.

```
<?xml encoding="UTF-8"?>
<!ELEMENT catalog (book)+>
<!ELEMENT book (author,title,genre,price,publish_date,description)>
<!ATTLIST book
          id CDATA #REQUIRED>
<!ELEMENT author (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT genre (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT publish_date (#PCDATA)>
<!ELEMENT description (#PCDATA)>
```

Obrázok 2.4: DTD dokumentu catalog.xml

2.1.4.2 XML schema

Jazyk XML schema vznikol z XML-Data a XDR, ktorý od roku 2001 konzorcium W3C [12] odporúča na tvorbu schém. Je to dnes najpoužívanejší jazyk po DTD [7]. Definuje presne miesta v dokumente, na ktorých sa môžu vyskytovať jednotlivé elementy. Definuje atribúty, poradie a počet elementov, dátové typy jednotlivých elementov a ich atribútov a podobne [8].

XML schema samozrejme je úplne postačujúci no však chýba mu jednoduchosť. Má celkovo zložitú špecifikáciu a tzv. temné zákutia. Už len z pohľadu na XML schéma je jasne viditeľný formát ako ma sám jazyk XML (viď obrázok 2.5). Súbory obsahujúce schéma dokumentu v jazyku XML schema majú koncovku .xsd.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="catalog">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="book" maxOccurs="unbounded"/>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="author" type="xs:string"/>
            <xs:element name="title" type="xs:string"/>
            <xs:element name="genre" type="xs:string"/>
            <xs:element name="price" type="xs:float"/>
            <xs:element name="publish_date" type="xs:date"/>
            <xs:element name="description" type="xs:string"/>
          </xs:sequence>
          <xs:attribute name="id" type:"xs:string" use="required"/>
        </xs:complexType>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Obrázok 2.5: XML schema dokumentu catalog.xml

2.1.4.3 Relax NG

Pretože ľudia nie sú nikdy spokojní a jazyk XML schema je celkom ťažké sa naučiť, vznikol elegantný jazyk Relax NG (*REgular LAnguage for XML Next Generation*). Relax NG (*RNG*) schéma špecifikuje vzor a obsah XML dokumentu. Samotný súbor obsahujúci schému v tomto jazyku je validný XML dokument. Jazyk okrem schémy v XML formáte poskytuje aj jednoduchý (tzv. kompaktný – *RNC*) formát, ktorý pripomína DTD. Od roku 2003 je Relax NG normovaný podľa ISO 19757 [11].

Keďže Relax NG je založený na vzoroch, celé schéma je vzorom (*pattern*) dokumentu. Vzor sa pritom skladá z vzorov elementov, atribútov a textových uzlov. Tieto vzory môžu byť ľubovoľne kombinované do usporiadaných alebo neusporiadaných skupín, môžeme určiť či sa opakujú alebo nie, poprípade ich voliteľnosť [10].

Najväčšou výhodou Relax NG je kompaktná syntax. Tá umožňuje zapísať schéma v textovej podobe, pripomínajúcej DTD. Rozdiel medzi oboma formátmi – RNG a RNC je možno vidieť na obrázkoch 2.6 a 2.7.

```
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
  <start>
    <element name="catalog">
      <oneOrMore>
        <element name="book">
          <group>
            <attribute name="id">
              <text/>
            </attribute>
            <group>
              <element name="author">
                <text/>
              </element>
              <element name="title">
                <text/>
              </element>
              <element name="genre">
                <text/>
              </element>
              <element name="price">
                <text/>
              </element>
              <element name="publish_date">
                <text/>
              </element>
              <element name="description">
                <text/>
              </element>
            </group>
          </group>
        </element>
      </oneOrMore>
    </element>
  </start>
</grammar>
```

Obrázok 2.6: RNG schéma ignorujúce dátové typy dokumentu catalog.xml

```
start =
  element catalog {
    element book {
      attribute id { xsd:string },
      (element author { xsd:string },
       element title { xsd:string },
       element genre { xsd:string },
       element price { xsd:float },
       element publish_date { xsd:date },
       element description { xsd:string })
    }+
  }
```

Obrázok 2.7: RNC schéma zahrňujúce definície dátových typov dokumentu catalog.xml

Vzory opakovania (oneOrMore, zeroOrMore) veľmi jednoducho odhalia, ktoré elementy sa v dokumente opakujú (viď obrázok 2.6). Je nutné podotknúť, že pri práci v RNG alebo RNC je možno použiť rozmanité dátové typy (viď obrázok 2.7), a tým pádom aj uskutočňovať spätnú kontrolu typov pri validácii. Dátové typy v Relax NG poskytujú rôzne možnosti aj ich kombinácii. Napríklad, máme element cena a vieme, že táto pozostáva z číselnej hodnoty a meny. Preto jednoducho vytvoríme dátový typ, ktorý bude obsahovať číslo (napr. *decimal*) a za ním bude označenie meny ako možnosť výberu (*choice*) medzi CZK alebo EUR. Pokiaľ vynecháme dátové typy všetky elementy a atribúty budú automaticky typu *text* ako na obrázku 2.6. Formát súborov jazyka Relax NG je pochopiteľne .rng alebo .rnc pre kompaktnú schému. Úplný popis syntaxi tohto jazyka je možno nájsť na [9].

2.2 Programovací jazyk Java

Snáď všetci poznáme známu šálku kávy svietiacu zväčša v taskbare. Java je objektovo orientovaný programovací jazyk, ktorý vyvinula firma Sun Microsystems a prestavila ho 23. mája 1995. Java je jeden z najpoužívanejších jazykov na svete. Jeho syntax je zjednodušenou verziou syntaxe jazyka C a C++.

Java je obľúbená hlavne kvôli svojej intuitívnosti a robustnosti. To ju predurčuje na písanie veľmi spoľahlivých a elegantných aplikácii. Taktiež má dobre realizovanú nielen správu pamäti ale aj bezpečnosť, ktorú užívateľ ocení najmä pri tvorbe sieťových a viacvláknových aplikácii.

Nakoľko sa jedná o interpretovaný jazyk, prekvapivo nemá až taký výrazný vplyv na výkon. Jedným z najcennejším znakom Javy je však jej prenositeľnosť a nezávislosť na architektúre počítača. Avšak existujú rôzne platformy, a tu si treba dávať pozor, pretože nižšia platforma nemusí podporovať všetky funkcie vyššej a tým pádom nám program nemusí fungovať. Funkčnosť Javy je takmer rovnaká, či už je to Microsoft Windows, Unix alebo OS X. Túto prenositeľnosť umožňuje vlastný interpret Javy, tzv. virtuálny stroj – *Java Virtual Machine* (JVM).

Zdrojový kód sa najprv preloží a až potom spustí preto je štart programov písaných v Jave pomalší oproti jazykom, ktoré kompilujú staticky ako napríklad C++. Existujú však technológie na zefektívnenie prekladu a urýchlenie tak celého procesu.

Ďalšou nevýhodou je vyššia pamäťová náročnosť programov, čo sa prejavuje hlavne u jednoduchých aplikácii, pretože je treba mať v pamäti celé prostredie, aby bol program schopný fungovať [13].

2.2.1 Java Virtual Machine

Java Virtual Machine (JVM) je sada počítačových programov a dátových štruktúr, ktoré využíva modul virtuálneho stroja, aby mohol spustiť počítačový program a skripty v jazyku Java. Je to interpret ktorý umožňuje preklad a spustenie programu. Vďaka tomu, že JVM je k dispozícii v mnohých platformách je možné aplikáciu v Jave vytvoriť len raz a potom ju spustiť na ktorejkoľvek platforme, kde pracuje JVM.

JVM sa dodáva spolu s určitým štandardnými knižnicami, ktoré sa nazývajú Java API (Application Programming Interface). Samotná aplikácia sa môže skladať z rôznych typov súborov a preto je možné ich zbaliť do jedného súboru typu .jar. Vďaka .jar súborom sme potom schopní jednoducho distribuovať aj rozsiahle aplikácie [14].

Virtuálny stroj Javy je štandardne nastavený tak, aby povolil malé množstvo pamäti. Avšak pri práci s veľkými súbormi typu XML je potrebné povoliť väčšie množstvo, aby nám program neskončil kvôli nedostatku pamäti. Jednoducho nám stačí JVM spustiť s parametrom na zvýšenie *Java Heap Space*, napríklad: –Xmx1000m. Tento parameter povolí aplikácii využiť maximálne 1000MB pre svoje potreby – naplnenie dátových štruktúr a podobne.

2.2.2 Swing

Swing je súbor jednoduchých komponentov na tvorbu grafického užívateľského rozhrania (GUI) v jazyku Java. Swing vznikol kvôli potrebe viac sofistikovanejších a krajších riešení grafických komponentov než poskytovalo predošlé AWT (Abstract Window Toolkit). Patrí k štandardnému vybaveniu Javy už od verzie 1.2 (JSE 1.2).

Swing je nezávislý na platforme a podporuje mechanizmus *look and feel*, ktorý umožňuje meniť vzhľad jednotlivých komponentov v aplikácii, bez toho aby sa robili razantné zmeny v samotnom zdrojovom kóde.

GUI implementované s pomocou Swingu sa ľahko až intuitívne konfiguruje a prispôsobuje. Bez problémov zmeníme jeho farbu, ohraničenie, názov, chovanie a podobne.

2.2.3 Vývojové prostredia

Existuje obrovské množstvo vývojových prostredí, ktoré sú schopné vyvíjať programy v jazyku Java. Jedným z najvýznamnejších je NetBeans IDE, priamo od Sun Microsystems, ktoré podporuje aj veľké množstvo iných programovacích jazykov, napríklad C++ a Ruby.

Ďalšie nemenej známe vývojové prostredie je Eclipse, no nesmieme však zabudnúť na voľne šíriteľný JDeveloper od firmy Oracle alebo BlueJ, taktiež voľne šíriteľné vývojové prostredie pre jazyk Java.

2.2.4 Java a XML

Samotná Java podporuje XML od verzie J2EE v1.3 (Java 2 Enterprise Edition v1.3). Prvým krokom k spracovaniu XML dokumentu je jeho analýza, kontrola syntaxe, overenie, či dokument zodpovedá danému DTD, poprípade doplnenie neuvedených hodnôt atribútov z DTD špecifikácie. Dokument sa potom prevedie na vnútornú dátovú reprezentáciu, ktorú možno ďalej spracovať. Celú túto činnosť (parsovanie) zaistí XML parser.

Existujú dva hlavé prístupy parsovania XML dokumentu, a to prístup založený na stromovej štruktúre a na udalostiach.

Parsovanie do stromovej štruktúry je vo väčšine prípadov založené na DOM (*Document Object Model*), čo je špecifikácia programového rozhrania konzorcia W3C. DOM definuje logickú štruktúru dokumentu a spôsob prístupu a manipulácie s dokumentom. V Jave tento balík možno nájsť ako `org.w3c.dom`.

Príkladom parsovania založeného na udalostiach je *SAX* (The Simple API for XML). Nakoľko by sa mohlo zdať, že použitie stromovej štruktúry je výhodné, existujú prípady, kde je to úplne inak. V súbore o veľkosti niekoľko megabytov sa pomocou stromovej štruktúry vyhľadáva veľmi neefektívne, preto sa využíva API založené na udalostiach, ktoré nemusí udržovať kontext celého dokumentu v pamäti.

Do jazyka Java bola zakomponovaná podpora XML až o niečo neskôr v podobe *Java API for XML Processing* (JAXP). JAXP definuje rozhranie na parsovanie a manipuláciu s XML dokumentmi a obsahuje aj rozhranie na XSL transformácie. Rozhrania na parsovanie možno nájsť v balíku `java.xml.parsers` [15].

Okrem samotnej JAXP existuje niekoľko open-source parserov, ktoré sa v súčasnosti hojne používajú, a to napríklad: *Xerces*, *DOM4J*, *JDOM* a iné.

2.2.4.1 JDOM

DOM aj SAX sú všeobecné špecifikácie, nezávislé na programovacom jazyku. Preto sa v roku 2000 zrodil projekt s názvom JDOM, ktorý sa pokúšal vyvinúť API na prácu s XML dokumentmi. JDOM, tak ako jeho názov napovedá, pracuje podobne ako DOM, teda reprezentuje XML dokumenty stromovou štruktúrou. JDOM je však menej pamäťovo náročný, rýchlejší a hlavne omnoho jednoduchšie sa s ním pracuje [15].

2.2.4.2 DOM4J

DOM4J je open-source knižnica, ktorá pracuje s XML, *XPath* a *XSLT* na platforme Java. Využíva *JCF* (Java Collections Framework) a plne podporuje DOM, SAX a JAXP. DOM4J sa snaží docieliť čo najjednoduchšej manipulácie s XML dokumentmi, za čo najmenšej spotreby pamäte, hlavne pri práci s veľkými dokumentami [16].

2.2.4.3 XERCES

Xerces je parser vyvíjaný v rámci Apache XML projektu. Poskytuje početné knižnice na parsovanie a generovanie XML dokumentov. Implementuje štandartné API, medzi inými aj DOM, SAX a SAX2, v jazykoch Java, C++ a Perl [17].

2.3 Relax NG a manipulácia s XML v Jave

Pomocou Javy a jazyka Relax NG boli implementované rôznorodé knižnice a aplikácie na validáciu dokumentov a schém, jak v XML (rng), tak aj v kompaktnom (rnc) formáte. Mnoho z nich sú open-source, ktoré môžeme voľne využiť pri tvorbe vlastného programu. Tieto tzv. validátory sú implementované v C (Libxml2), C# (Tenuto), Pythone (XVIF) a samozrejme aj v Jave (Jing, MSV).

Ďalšou neoddeliteľnou súčasťou práce s XML dokumentmi, je generovanie schémy dokumentu zo samotného .xml súboru. Samozrejme nesmieme zabudnúť na konverziu z jedného typu schémy do druhého. Predákmi v tejto časti sú Trang, InstanceToSchema a Sun RELAX NG Converter. Všetky tieto aplikácie sú implementované v Jave [9].

V neposlednom rade samozrejme existujú aj generátory kódu, ktoré sú schopné podľa schémy vygenerovať triedy a pomocou nich potom vytvoriť XML dokument. Na internete taktiež možno nájsť nespočetné množstvo prehliadačov a editorov XML.

2.3.1 InstanceToSchema

Utilita InstanceToSchema (i2s) je časť XML editora s názvom *xmloperator*. Tento open-source program implementovaný v Jave spoľahlivo generuje schému vo formáte RNG z XML dokumentu. Je to konzolová aplikácia, ktorá potrebuje na svoju činnosť J2SE 1.3 alebo 1.4 a SAX parser vyhovujúci JAXP. Software zakladá na kategóriach vzorov, ktoré reprezentujú sadu vzorov Relax NG. Snaží sa preskúmať celý dokument a zisťuje tak či daný vzor sedí k elementu, ktorý reprezentuje a zahŕňa širokú škálu rôznorodosti XML.

2.3.2 Trang

Trang je voľne šíriteľný konvertor medzi jednotlivými schémami založený na Relax NG. Je súčasťou projektu *thaiopensource*. Trang podporuje tieto jazyky: Relax NG (aj XML aj kompaktnú syntax), DTD pre XML 1.0 a W3C XML schema. Schéma v akomkoľvek z vyššie uvedených jazykov je možno prekonvertovať do iného jazyka. Jedinou výnimkou je, že W3C XML schema môže byť len výstupným formátom. Trang sa snaží vytvoriť čo najzrozumiteľnejšie a najčitateľnejšie schéma pre ľudské oko. Využíva niektoré časti validátora Jing, z čoho vyplýva, že je implemenovaný v jazyku Java [9].

3 Návrh aplikácie

Obsahom tejto kapitoly je návrh aplikácie, jej teoretickej funkčnosti, koncepcia jednotlivých častí a ich prepojenie. Budeme predpokladať, že užívateľ bude chcieť vizualizovať hlavne dátovo orientované XML dokumenty, teda na reprezentáciu takýchto dát sa dokonale hodí tabuľka. Neodsúdime však ani textovo orientované dokumenty, no ich vzhľad už nebude taký prívetivý.

3.1 Ciele implementácie

Cieľom práce je vytvoriť aplikáciu, ktorá interaktívne zobrazí dátovo orientovaný XML dokument. Chceme teda naprogramovať prehliadač dokumentov, preto by sme sa mali zamerať predovšetkým na:

- Jednoduché a prehľadné GUI
- Jednoduché a intuitívne ovládanie
- Možnosť prehliadať XML dokument a jeho schému v pôvodnom tvare
- Schopnosť generovať schému dokumentu
- Vizualizácia načítaného XML dokumentu pomocou tabuliek a stromu

Program môže v konečnom dôsledku slúžiť aj ako editor XML dokumentu alebo jeho schémy, avšak iba ich textových reprezentácii. Dokument vo vizualizovanej forme nebude možné editovať alebo ukladať na disk.

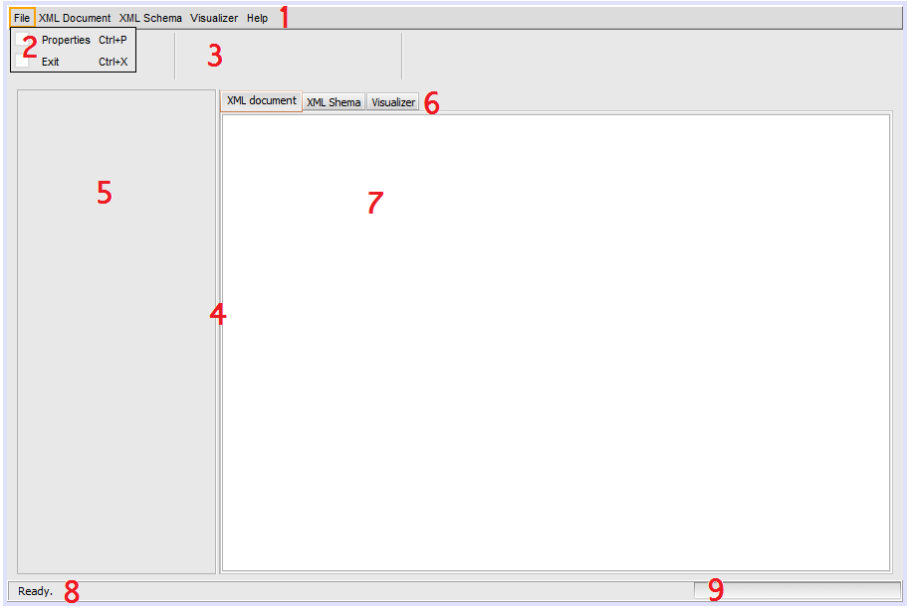
3.2 Koncept GUI

Základom aplikácie s grafickým užívateľským rozhraním je charakteristické dobre vyzerajúce, prehľadné a intuitívne ovládateľné rozhranie. Keďže implementujeme prehliadač je veľmi vhodné zvoliť štruktúru podobnú prieskumníku operačného systému Microsoft Windows. Navrhneme si teda jednoduchý vzhľad aplikácie. Prílišný počet grafických efektov, blýskajúcich sa tlačítiek pôsobí rušivo pri čítaní alebo inej činnosti, ktorá vyžaduje väčšiu koncentráciu, preto zvolíme statickejšiu grafiku.

Náčrtok môžeme pozorovať v obrázku 3.1. Určite budeme potrebovať panel s rôznymi menu na ovládanie aplikácie (č. 1), pričom každé menu bude obsahovať podmnožinu akcií (č. 2) pre danú časť aplikácie. Každá táto časť oddeľuje určitý logický súbor funkcionality. Tak ako aj v prieskumník u MS Windows používame niektoré ovládacie prvky viac, je výhodné ich sprístupniť aj bez toho aby sme ich museli zakaždým hľadať, vybrať na tzv. toolbar (č. 3).

Hlavné okno rozdelíme vhodne na dve plochy pomocou separátoru (č. 4), čím dovolíme voľne meniť šírku oboch plôch. Do týchto dvoch okien (č. 5 a 7) ľahko zobrazíme to, čo potrebujeme. Taktiež môžeme jednotlivé okná pomenovať, pre väčšiu prehľadnosť, ako napríklad č. 6.

Ďalším vhodným komponentom GUI je rozumne podať užívateľovi správu, o tom čo program aktuálne robí. Zvyčajným prvkom je tzv. statusbar (č. 8), kde program jednou vetou popíše aktuálnu činnosť a poprípade zobrazí v akom štádiu spracovávanie tejto činnosti je, a to buď v percentách alebo grafickým komponentom (č. 9).



Obrázok 3.1: Návrh aplikácie XMLspark

3.3 Zobrazenie obsahu dokumentu

Ako sme v úvode naznačili, prehliadačom rozumieme software schopný prezerať si obsah súborov zvoleného typu. Keďže XML nie je nič iné ako text s charakteristickou štruktúrou, usúdime, že zobraziť tento text nie je na zahodenie. V podstate môžeme implementovať editor textu pre XML dokumenty, a umožniť tak užívateľovi modifikovať otvorený dokument bez potreby spúšťania cudzích aplikácii.

Oceniteľným prvkom je určite schopnosť validovať vstupný XML dokument a predísť tak zbytočným problémom pri práci s chybným dokumentom. Na zobrazenie textu sa v jazyku Java obyčajne používa komponent *textbox*. Textbox automaticky bude potrebovať skrolovacie panely, aby sme boli schopní vidieť text presahujúci hranice tohto prvku. Obsah textboxu bude najvýhodnejšie zobraziť vo veľkom okne aplikácie, napríklad ako v obrázku 3.1 (č. 7), pretože dokumenty obsahujú zvyčajne mnoho znakov.

3.4 Zobrazenie schémy dokumentu

Schéma XML dokumentu je mnohokrát jeho neoddeliteľnou súčasťou. Dôležitosť schém sme si priblížili v kapitole 2.1.4, a preto by naša aplikácia určite mala podporovať aj rôznorodosť

jednotlivých jazykov na tvorbu schém XML dokumentov. S určitosťou si dovolím povedať, že ďalším veľkým prvkom, v tomto prípade editor a súčasne aj prehliadač schém, bude na mieste.

Avšak v drvivej väčšine prípadov nemáme k dispozícii schému dokumentu a treba nám ju osobitne vytvoriť, poprípade použiť nejaký program na jej vygenerovanie. A čo tak zakomponovať generátor schém do našej aplikácie? Zväčší to síce pamäťovú náročnosť, ale ľudia majú radi programy, ktoré majú viac než len jednu schopnosť. Pri súčasnom stave informačných technológii si môžeme zvýšenie nárokov v tak malom programe dovoliť a rozšíriť tak funkčnosť.

Schéma XML dokumentu je ako aj samotný XML súbor je tiež text, preto môžeme podobne ako u dokumentu vypísať obsah súboru so schémou do textboxu. Nakoľko už v hlavnom okne s textboxom bude načítaný dokument, je vhodné pridať člen so záložkami, a tak pomocou záložiek sa prepínať medzi jednotlivými textami (obrázok 3.1 č. 6).

3.5 Vizualizácia

Pole pôsobnosti dátovo orientovaných dokumentov je predovšetkým v spracovaní dát počítačom, pričom tieto dáta nie sú až tak dobre zobraziteľné pre samotných ľudí, ak sa nepoužije špecifický software. Na rozdiel od textovo orientovaných dokumentov, kde po odstránení tagov porozumieme obsahu celkom bez problémov, dátovo orientované dokumenty sa stávajú nečitateľné, pretože nevieme čo reprezentujú. Reprezentácia dát je samozrejme uložená v tagoch a tie sa tak stávajú neoddeliteľnou súčasťou samotných dát [18].

Základom dátovo orientovaných dokumentov je štruktúra dát, ktorá je v prípade dokumentu písané v jazyku XML stromová. Jednou z možností vizualizácie dát v XML bude určite teda strom (bližšie v kapitole 3.5.1).

Ako ďalšia možnosť sa nám naskytuje vizualizovať dátovo orientované XML dokumenty ako tabuľky. Tento postup je logický, pretože väčšina informácii v takýchto dokumentoch pripomína zoznam a môže byť reprezentovaná ako kolekcia elementov s rovnakým alebo podobným obsahom. Kolekcie podobných dát sa odjakživa zobrazujú do tabuliek, kde sú veľmi dobre čitateľné (viac v kapitole 3.5.2).

3.5.1 Strom

Už sme spomínali, že štruktúra XML dokumentu je vždy stromová, pre užívateľov bude určite vhodné vizualizovať dokument stromom. Pokiaľ chceme rýchlo a prehľadne zobraziť obsah XML toto je najlepšia technika, ktorú môžeme použiť. Tak ako aj strom aj XML ma svoj koreň. Pomocou parseru typu DOM ľahko prejdeme všetky uzly v dokumente a postupne ich pripájame za sebou tak, ako sú uložené v súbore. Takýto strom môžeme vidieť na obrázku 3.2, kde dokument *catalog* obsahuje dvanásť elementov typu *book*, ktoré reprezentujú záznam o knihách.

Obrázok 3.2: Strom dokumentu obsahujúci malý počet elementov.

Obrázok 3.3: Strom dokumentu obsahujúci veľký počet podobných elementov.

Avšak technika ilustrácie stromom má svoje nevýhody. Keďže predpokladáme, že pracujeme s dátovo orientovanými súbormi, pravdepodobne sa stretneme s veľkým množstvom hierarchických dát, kde sa uzly opakujú mnohonásobne a strom sa stane neprehľadným a zbytočne dlhým ako na obrázku 3.3.

Strom so stovkami rovnakých uzlov naberá na veľkosti, a treba ho zjednodušovať. Existuje niekoľko publikovaných techník na redukciu stromov ako napríklad prepisovanie stromovej štruktúry (popísaná v [18]), ale z môjho pohľadu nenapomáha užívateľovi pochopiť obsiahle dokumenty, lebo je treba definovať rôzne pravidlá na zapuzdrovanie jednotlivých mohutných častí. A tu prichádzajú na scénu tabuľky.

3.5.2 Tabuľky

Dlhé sekvencie elementov s podobnou alebo rovnakou stavbou sa obvykle stávajú terčom zapuzdrovania do menších, kompaktnejších objektov. Na zobrazenie takéhoto množstva dát sa

vyložene hodia tabuľky. Vďaka ich jednoduchosti a dobrej prehľadnosti sa v nich môže orientovať aj obyčajný užívateľ. Ako však dáta zo súboru vizualizovať?

Jedným z možných spôsobov je použiť klasické relačné tabuľky na zobrazenie štruktúry XML dokumentu. I keď relačné tabuľky sú kritizované za ich malú flexibilitu pri zobrazovaní hierarchických a objektovo orientovaných dát, ktoré sa nachádzajú práve v XML, ešte stále sa využívajú relačné databázy viac ako objektové.

Hlavnou myšlienkou tejto techniky je využiť tabuľky tam, kde sú najlepšie – zobrazovanie kolekcií obrovskej masy dát. Veľmi rýchlo zistíme, že vyhľadávanie takýchto dát v dátovo orientovaných XML dokumentoch je prekvapivo ľahké, pretože samy dokumenty tohto typu ich obsahujú v drvivej väčšine prípadov.

Ako ale takéto sekvencie identifikovať? Dokumenty XML vždy podliehajú určitej schéme, podľa ktorej sú napísané, teda prečo túto schému nevyužiť práve na detekciu našich sekvencii? Áno, je to práve schéma, ktorá odhalí pravú tvár dokumentu, poukazujúc tak na nami toľko hľadané kolekcie dát.

V schémach písaných v jazyku XML schema alebo Relax NG tieto časti odhalíme veľmi rýchlo, a to pomocou kľúčového slova “unbounded” v prípade XML schémy alebo “oneOrMore” v RNG schéme. Tieto magické slová nám ihneď odhalia opakujúce sa časti, a tým pádom aj obsah tabuliek. Po vytvorení tabuliek pomocou schémy nám stačí už len prehľadávať dokument a naplniť tak tabuľky dátami, postupne pridávať jednotlivé riadky reprezentujúce sekvenciu dátových elementov, a vizualizácia je hotová.

Celý proces, ako aj ďalšie vizualizačné metódy sú presne popísané v [18] a [19].

4 Implementácia programu XMLspark

Aplikácia XMLspark je výsledkom spracovania problematiky interaktívnej vizualizácie dátovo orientovaných XML dokumentov. Implementačný jazyk je Java za pomoci Swing GUI a balíkov JDOM, RelaxNG, DOM4J a Apache-commons. Program bol vytvorený a je funkčný pod platformou JDK 6. XMLspark bol naprogramovaný pomocou NetBeans IDE 6.5.1.

Celý náš projekt má 3 hlavné balíky, a to `base`, `convert` a `xmltorng`. Najdôležitejším balíčkom je `base`, ktorý obsahuje celé GUI a balíky s triedami na vizualizáciu stromom (`base.tree`) a tabuľkami (`base.table`). Ostatné dva balíky slúžia na generovanie schémy dokumentu v rôznych formátoch. Všetky tieto balíky možno nájsť v adresári `src`, obsahujúci všetky zdrojové kódy programu.

Adresár `img` obsahuje obrázky čo sa používajú v grafickom rozhraní predovšetkým ako tlačítka. V adresári `other/lib/` sa nachádzajú použité prídavné balíčky, ktoré môžeme pomenovať aj knižnice, na zjednodušenie práce s XML dokumentom (jdom, xerces a dom4j) a súbormi (commons-io). Jednoduché príklady dokumentov môžeme nájsť v `other/samples/`. Samotný adresár `other` slúži na ukladanie medzi-výstupu generátoru schém a ďalších, napríklad vzorových, súborov.

Hlavnou triedou (*main*), ktorá sa v podstate stará o priebeh celého programu, je `MainWindow` a nachádza sa v balíku `base`. Tu sa inicializuje celé GUI, načítavajú sa všetky obrázky, konfiguračný súbor a nastavujú sa počiatočné premenné a pod.. XMLspark môžeme rozdeliť na tri logické časti: manipulácia s XML dokumentom, so schémou a *vizualizér*. Podrobnejšie celú aplikáciu popíšem na nasledujúcich riadkoch.

4.1 Grafické rozhranie

Grafické užívateľské rozhranie je prvá vec, ktorá nám padne do oka keď spustíme akýkoľvek program, preto sme ho implementovali jednoducho a prehľadne (viď obrázok 4.1). Celý program je v anglickom jazyku, kvôli univerzálnosti.

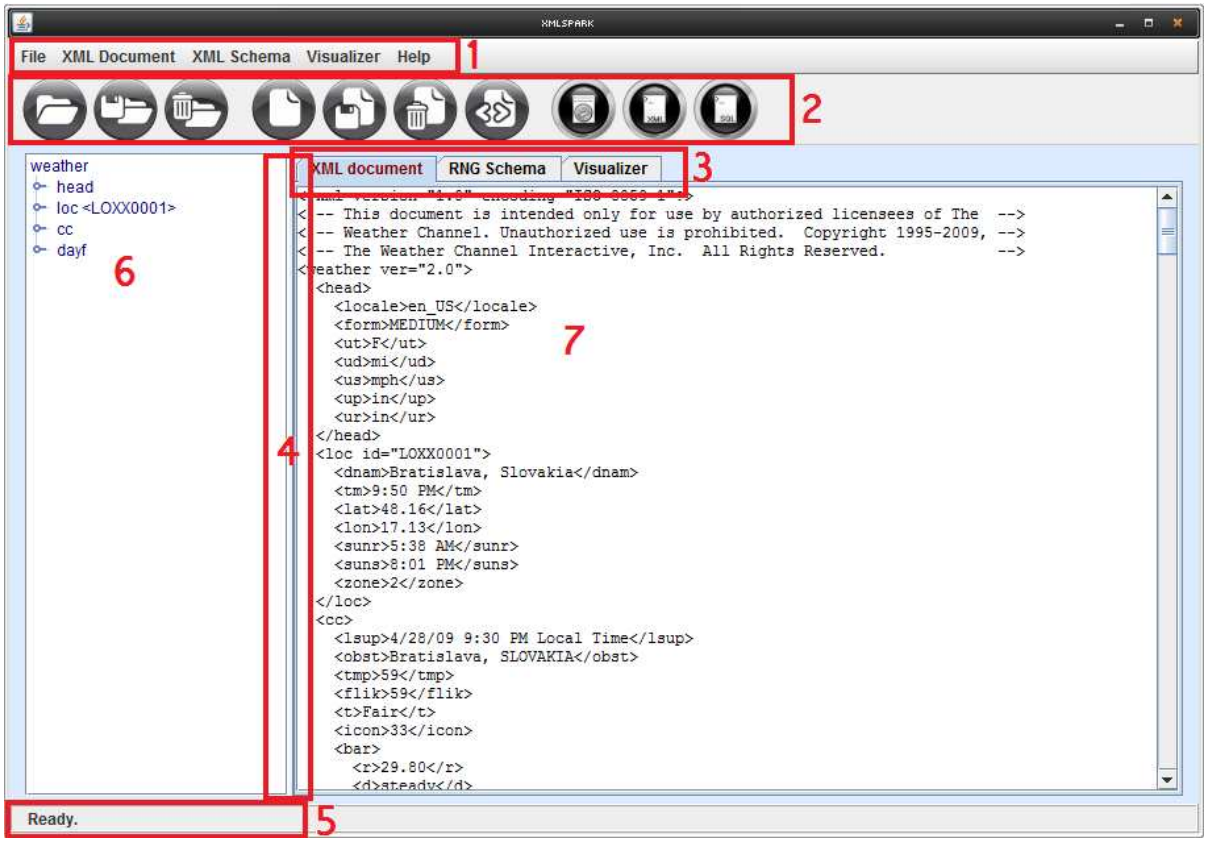
Hlavné ovládacie prvky sú v hornej časti v podobe klasických `JMenu` (č. 1) alebo zjednodušeného panelu nástrojov (prvok `JToolBar`) (č. 2). Každé tlačítko (`JButton`) toolbaru má stručný popisok (tooltip) jeho funkčnosti. Všetky tlačítka sú implementované tak aby sa zrušila ich funkcionalita keď sú nevhodné alebo nepoužiteľné, a tak sa ich farba sa zmení z čiernej na sivú. To isté platí aj o menu o pár centimetrov vyššie (č. 1).

Jednotlivé tlačítka sú rozdelené do troch skupín. Prvé tri slúžia na manipuláciu s XML dokumentom. Druhá skupina tlačítok obsluhuje buď načítanie schémy dokumentu alebo spúšťa jej generátor. Posledná skupina tlačítok slúži na spustenie vizualizácie už otvoreného dátovo orientovaného dokumentu.

Hlavné okno nám delí pohyblivá priečka (č. 4) na dve časti. Vľavo (č. 6) sa vykresľuje ľubovoľný otvorený XML dokument v komponente typu `JTree` (viac v 4.4.1). Na pravej strane (č. 7) je priestor pre reprezentáciu samotného dokumentu, a to či už v podobe textu alebo tabuliek. Pre túto časť sme využili komponentu typu `JTabbedPane`. Obsah tejto komponenty sa mení pomocou lišty so záložkami (č. 3), vďaka ktorej si môžeme prepínať buď textovú reprezentáciu XML dokumentu, zobraziť jeho schému alebo tabuľky naplnené dátami z dokumentu.

Posledným viditeľným prvkom je *statusbar* (č. 5), kam sa pomocou metódy `UpdateStatus` vypisuje aktuálna činnosť, ktorú program práve vykonáva. Tento člen GUI naberá význam v prípade väčších dokumentov, pretože programu trvá spracovanie dokumentu výrazne dlhšie a informuje tak užívateľa, že sa niečo deje, a nemá program z netrpezlivosti vypínať.

Aplikácia XMLspark podporuje tri módy zobrazovania. Prvý z nich je síce najzdĺhavejší, no umožňuje najväčšie možnosti, lebo povoľuje uloženie načítaného dokumentu a jeho schémy a súčasne aj vizualizáciu. Druhý mód umožňuje zobraziť, generovať a editovať len schému dokumentu a spustiť vizualizáciu. Posledná voľba vizualizuje súbor priamo bez zobrazovania schémy alebo dokumentu. Módy zobrazovania možno nastaviť v menu `File` v `Properties`. Tieto nastavenia sa ukladajú do XML súboru s názvom `config.xml` pomocou knižnice dom4j. Konfiguračný súbor sa načítava pri štarte a ukladá pri skončení programu.

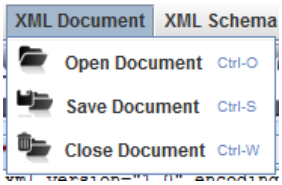


Obrázok 4.1: Grafické rozhranie programu XMLspark.

4.2 Manipulácia s XML dokumentom

Po štarte programu program povolí malý počet akcií, pri čom jednou z nich je otvorenie XML dokumentu a jeho následné spracovanie. O manipuláciu s XML dokumentom sa stará prvá skupina tlačítok v toolbare (obr. 4.2) alebo menu s názvom *XML document* (obr 4.3).

Obrázok 4.2: Tlačítka 1



Obrázok 4.3: Menu 1

Prvé tlačítko (komponenta `JButton`) alebo položka v menu (*Open Document*) slúži na otvorenie dokumentu, ktorý chceme zobraziť, poprípade neskôr s ním ďalej pracovať. Kliknutím alebo stlačením klávesovej skratky sa zavolá metóda `XMLOpen`, ktorá nám pomocou komponenty `JFileChooser` umožní prehliadať súbory s príponou .xml. Vybraný súbor sa potom načíta do textového poľa v aplikácii za pomoci metódy `docToString` z triedy `XMLhandler`. Takto otvorený XML dokument ostáva otvorený v pamäti programu a je voľne prístupný editácii v podobe textu. Editovaný dokument potom môžeme jednoducho uložiť s vykonanými zmenami (druhé tlačítko) alebo zavrieť (tretie tlačítko).

Uloženie dokumentu ovláda metóda `XMLSave`, pričom samotné uloženie dát vykonáva `writeFromString` pomocnej triedy `XMLhandler`. Zavretie súboru metódou `XMLClose` len zruší otvorený dokument a nastaví premenné na počiatočné hodnoty. Grafické rozhranie sa následne tiež zresetuje do úvodného stavu vďaka metóde `XMLClosed`.

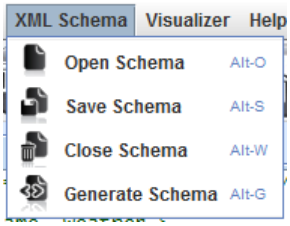
Ak sme už nejaký dokument otvorili, aplikácia nám umožní generovať XML schému (viac v nasledujúcej kapitole).

4.3 Práca so schémami dokumentov

Jednou z ďalších schopností aplikácie je manipulácia so schémou načítaného XML dokumentu. Ovládacie prvky tejto časti sú tlačítka strede toolbaru (obr. 4.4) alebo menu *XML schema* (obr 4.5). Pokiaľ k otvorenému dokumentu existuje schéma, je ju možné otvoriť. Ak neexistuje potom je nutné ju vygenerovať.

Zo schémy, či už vygenerovanej alebo otvorenej, a dokumentu nám aplikácia dovolí spustiť vizualizátor a predvedie nám tak XML dokument pod úplne iným rúškom (viac v kapitole 4.4).

Obrázok 4.4: Tlačítka 2



Obrázok 4.5: Menu 2

4.3.1 Súbor so schémou

Prvé tri tlačítka a položky menu pracujú takmer rovnako ako v prípade súboru XML. Prvé tlačítko alebo položka v menu slúži na otvorenie schémy, druhé na uloženie (*Save Schema*) a tretie na zavretie (*Close Schema*) aktuálnej schémy. Metódy, ktoré spracúvajú jednotlivé kliky na tlačítka sa tiež volajú podobne ako v predošlom prípade, a to `XSDOpen`, `XSDSave`, `XSDClose` a `XSDClosed`. Výnimku však tvorí štvrté tlačítko alebo položka menu, ktorá je tu navyše. Toto magické tlačítko nám umožňuje generovať schéma otvoreného dokumentu.

4.3.2 Generátor schém

Aplikácia obsahuje rozsiahly generátor schém z XML dokumentov. Tento generátor sa skladá z dvoch veľkých balíkov, a to `xmltorng` a `convert`.

Balík `xmltorng` je prispôsobený našim potrebám z open-source zdrojového kódu programu *InstanceToSchema* (viď kapitola 2.3.1) projektu *xmloperator*. Hlavnou triedou tohto balíčka je trieda `SchemaGenerator`, ktorá sa nachádza v `xmltorng.i2s`. Z `MainWindow` sa zavolá metóda `GenerateXMLtoRNG` triedy `SchemaGenerator`, a tá vygeneruje schému dokumentu v jazyku RelaxNG do súboru s názvom `virtualout.rng` v adresári `other`. Tento výstup je pre nás veľmi výhodný, pretože RNG schéma je vlastne súbor napísaný v jazyku XML. Jedinou chybičkou krásy je, že InstanceToSchema produkuje schému bez kontroly dátových typov, teda všetky elementy budú typu text. Pre nás je to však postačujúce.

Ako sme už spomínali, XMLspark dokáže generovať schéma dokumentu v štyroch formátoch a preto je nutné zakomponovať vnútorný konvertor, ktorý z RNG schémy vytvorí ľubovoľne zvolený formát. Jazyk schémy kam sa skonvertuje sa určuje podľa nastavení v `Properties`. Tu si užívateľ v `JComboBox` komponente vyberie formát. Pokiaľ je výstupný formát iný ako RNG prichádza na scénu balíček `convert`.

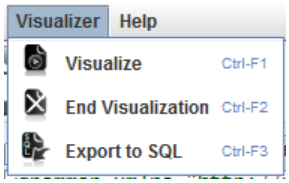
Tento balíček je prispôsobenou verziou open-source programu *Trang* (viď kapitola 2.3.2). Hlavným prvkom, ktorého metóda `SchemaToSchema`, je `Driver` z balíka `convert.relaxng.translate`. Metóda `SchemaToSchema` generuje zo vsupného súboru `virtualout.rng` výstup v inom jazyku schémy. Napríklad keď chceme generovať DTD z XML dokumentu celý postup vyzerá nasledovne.

Z načítaného súboru sa pomocou `InstanceToSchema` vygeneruje `virtualout.rng`. Tento súbor je metódou `SchemaToSchema` skonvertovaný do `virtualout.dtd` a obsah skonvertovaného výstupu vo `virtualout.dtd` je vypísaný do texboxu pre schému umiestneného v `MainWindow`.

4.4 Implementácia vizualizéru

Ak už užívateľ otvoril XML dokument a otvoril schému vo formáte RNG alebo ju vygeneroval môžeme začať s vizualizáciou. U generovanej schémy na formáte nezáleží, pretože aplikácia automaticky generuje RNG, pričom túto schému uchováva, a až potom konvertuje do iného formátu. Vizualizácia sa spúšťa stlačením prvého tlačítka v tretej skupine (obr. 4.6) alebo položkou Visualize v menu Visualizer (obr. 4.7). Druhé ovládacie prvky slúžia na ukončenie vizualizácie a tretie sú prípravou pre možné rozšírenie aplikácie o exportovanie vygenerovaných tabuliek do SQL. Vizualizácia začína validádiou schémy a dokumentu. Ak validácia prejde bez problémov vygeneruje sa prvý strom a potom tabuľková reprezentácia.

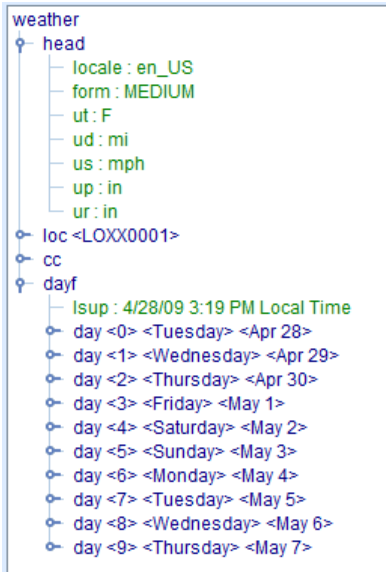
Obrázok 4.6: Tlačítka 3



Obrázok 4.7: Menu 3

4.4.1 Reprezentácia stromom

Prvým implementovaným vizualizačným prvkom je strom. Je to logické, lebo ako sme už niekoľko krát spomínali XML je typické svojou stromovou štruktúrov, a teda ho tak aj môžeme reprezentovať. Táto stromová reprezentácia sa generuje priamo z dokumentu XML, a celý proces tvorby tohto stromu vykonáva trieda `XMLTree`, ktorá sa nachádza v balíku `base.tree`. `XMLTree` k tvorbe stromu využíva triedy `JDOMAdapterNode`, ktorá zastrešuje interpretáciu elementu (uzlu stromu) a jeho metódy na prechádzanie stromovej štruktúry v dokumente pomocou knižnice JDOM. `JDOMToTreeModelAdapter`, ako už z názvu vyplýva uskutočňuje samotné naplnenie stromu dátami z dokumentu za pomoci triedy `XMLTreeCellRenderer`. Keď `XMLTree` vyparsuje XML dokument, predá vytvorený strom do `MainWindow`, kde strom vykreslí ako komponenta typu `JTree` na ľavej strane hlavného okna. Výsledok potom vyzerá ako obrázok č. 4.8. Modrým písmom sa vyznačujú elementy, ktoré majú potomkov a na zeleno sú zobrazené takzvané *listové* elementy.



Obrázok 4.8: Výstup vizualizátora – strom

4.4.2 Reprezentácia tabuľkami

Celé generovanie tabuliek zariaďuje trieda `TableCreator` v balíku `base.table`. Túto vizualizáciu inicializuje `MainWindow`, ktoré volá metódu `parse`. Na začiatku sa vygeneruje prvá tabuľka, ktorá sa volá podľa koreňového elementu XML dokumentu. Nasleduje tvorba modelov tabuliek podľa schém. Začne sa nájdením *referenčných elementov*, ktoré sa zvyčajne nachádzajú na konci RNG schémy. Referenčný element je v podstate odkaz na iný element, ktorý je potrebný kvôli viacnásobnému výskytu v rozdielnych častiach XML dokumentu. Tieto elementy sa uložia do zoznamu aby boli pripravené pri prechádzaní schémou. Potom sa rekurzívne volá `ChildrenListHandle` na každý uzol, čím sa prejde celá schéma vyhľadávajúc špecifické elementy v súbore písanom jazykom RNG.

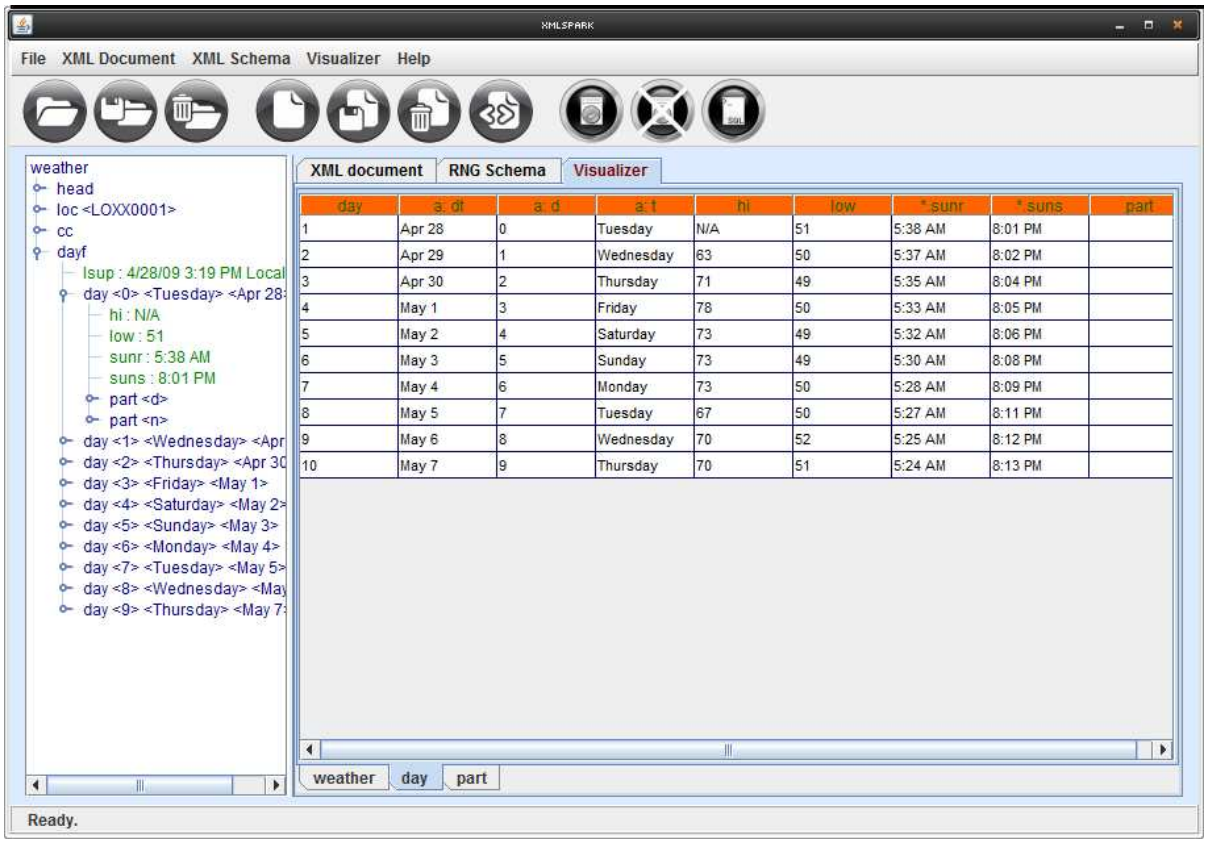
`ChildrenListHandle` reaguje predovšetkým na elementy s menom `oneOrMore`, `element` a `ref`. `OneOrMore` definuje kolekciu rovnakých po sebe idúcich elementov, teda toto je ukazateľom, že treba pridať novú tabuľku kde sa bude pridávať jeden element za druhým ako stĺpce, až pokiaľ neskončí daná sekvencia. Taktiež sa pri tvorbe novej tabuľky vyhľadajú všetky atribúty opakovaného elementu. Je ich možno nájsť hneď medzi prvými stĺpcami, a to s maskou *"a: názov_attribútu"*.

Pri objavení referenčného elementu program iba priamo skočí na referenciu a pokračuje normálne ďalej. Tag v schéme RNG s názvom `element` definuje klasický element ako ho poznáme v XML a teda sa zapíše do aktuálnej tabuľky ako nový stĺpec s menom, ktoré je uložené ako atribút s názvom `name`.

Pokiaľ rodič aktuálneho `element`-u nie je tvorcom tabuľky, tak sa názov stĺpca v tabuľke bude mať meno *"rodič.element"*. Je to kvôli lepšej čitateľnosti tabuliek.

Keď už sú hlavičky tabuliek vytvorené, už sa len popridávajú riadky s dátami z XML dokumentu. Napĺňanie riadkov zariaďuje metóda `FillTables` z triedy `TableCreator`. Dokončení napĺňania sa očíslujú jednotlivé riadky v tabuľkách, odstránia sa prípadné prázdne riadky a vizualizácia je ukončená. Výsledné tabuľky môžu vyzerať napríklad ako na obr. 4.9.

Na dolnej lište v obrázku vidíme vytvorené tri tabuľky a ich mená. Program je limitovaný aby povolil zobraziť dokumenty, ktoré vygenerujú maximálne 30 tabuliek. Toto nastavenie možno zmeniť jedine v samotnom zdrojovom kóde aplikácie (premenná `XMLTableMaxCount` v `MainWindow`).



Obrázok 4.9: Výsledok vizualizácie súboru s predpoveďou počasia.

5 Testovanie

Testovanie aplikácií je veľmi dôležitá súčasť popri ich samotnom vývoji. Preto ju nemôžme podceňovať, ba naopak, treba jej venovať dostatočné množstvo času. Pre testovacie účely si treba pripraviť vzorové dáta a zhodnotiť aké všetky stavy, napríklad v grafickom rozhraní, môžu nastať. Testovanie musí robiť aj samotný programátor, a taktiež aj niekto úplne cudzí, aby sme zistili, či je grafické rozhranie a vlastne aj celý program intuitívny, ľahko ovládateľný a pochopiteľný pre užívateľov, ktorí nemajú o ňom skoro žiadnu predstavu.

Z pohľadu programátora som aplikáciu odtestoval vzhľadom na vstupné dokumenty XML, a to predovšetkým na prázdnych, nevalidných alebo bohato-štruktúrovaných dokumentoch. Samozrejme som použil dokumenty, ktoré sa používajú v internete alebo ich produkujú aplikácie. XML dokumenty z GPS trackeru, ktoré mi poskytol Bc. Ivan Kramárik, spôsobli snáď najväčšie problémy, lebo boli veľmi zložito štruktúrované. Nachádzali sa v nich početné referenčné elementy a veľa prázdnych elementov s veľkým počtom atribútov. Z internetu som použil dokumenty ako napríklad: katalóg kníh, zoznam CD nosičov, správa o počasí na nasledujúcich desať dní, zoznam staníc merajúcich počasie a podobné.

Druhou fázou testovania bolo poskytnutie aplikácie obyčajnému nezainteresovanému užívateľovi. Tu som využil Bc. Martina Gábora. Poskytol mi spätnú väzbu predovšetkým v oblasti GUI. Otestoval väčšinu možností a odhalil pár drobných chýb, ktoré boli následne opravené.

Testovanie teda v konečnom dôsledku splnilo svoj účel a jeho opodstatnenie sa preukázalo.

6 Záver

Počas trojmesačnej tvorby tejto bakalárskej práce sa moje skúsenosti a znalosti v oblasti jazyka XML badateľne rozšírili. Pochopil som syntax jazyka RelaxNG a naučil som sa tvoriť schémy dokumentov v tomto jazyku.

Aplikáciu XMLspark som implementoval podľa zadania a návrhu, pričom som ju rozšíril o prvky vhodné aj pre pokročilejších užívateľov, ktorí určite ocenia generátor schém vo viacerých jazykoch. Táto časť bola súčasne najtvrdším orieškom na implementáciu, pretože som musel študovať cudzí voľne šíriteľný zdrojový kód a prispôsobiť ho mojím potrebám. Druhou obtiažnou časťou bola implementácia samotnej vizualizácie, kde som musel vyhodnocovať všetky stavy, ktoré môžu nastať. Počas implementácie vizualizácie som narazil na problémy pri práci s dátovým typom `List`, pretože u Javy na rozdiel od jazyka C je ťažko rozlíšiť čo je len odkaz a čo konkrétna premenná a vznikali preto problémy s referenciou.

Vizualizácia dát reprezentovaných v XML dokumente má využitie hlavne pre obyčajných ľudí, pretože sú schopní rýchlejšie pochopiť obsah súboru. Nástroj XMLspark teda v praxi súži na zjednodušené a elegantné zobrazenie dokumentov, ktoré sú ako text prakticky nečitateľné. I keď je v súčasnom stave XMLspark využiteľný, určite je vhodné ho rozšíriť o pár bonusových vlasností.

6.1 Možné rozšírenia

Aby sa program uchytil v praxi obyčajne nestačí aby bol jedno účelný. Preto treba navrhnúť a zvážiť jeho možné vylepšenia.

Prvým vylepšujúcim prvkom, ktorý by určite bolo vhodné implementovať je vyhľadávanie určitých frázi, a to či už v texte XML dokumentu alebo jeho schémy, poprípade v strome alebo tabuľkách reprezentujúcich tento dokument.

Vyhľadávací modul by mal vedieť vyhľadávať podľa názvu elementu, názvu atribútu a samozrejme ich obsahu. Okrem samotného vyhľadávania by tento modul mohol slúžiť aj ako nástroj na hromadné premenovanie viacerých elementov alebo ich atribútov.

V súčasnom stave je možno v programe XMLspark editovať len textovú reprezentáciu dokumentu alebo jeho schémy. Veľkým prínosom by teda určite bolo pridať možnosť editácie samotných tabuliek a stromu alebo schopnosť prenášať a kopírovať časti stromu, meniť poradie stĺpcov a riadkov tabuľky a potom takto zmenený dokument uložiť na disk.

Ďalším žiadaným rozšírením je exportovanie tabuliek do formátov ako napríklad HTML, MS Excel, SQL a podobne. Tu by sa avšak už museli využiť ďalšie prídavné knižnice a program by začal naberať na veľkosti.

Export do SQL je asi najnutnejším rozšírením, pretože SQL ako aj dátovo orientované XML sú predurčené na ukladanie veľkého množstva dát a mohli by sme tak XML využiť na prenášanie dát medzi nekompatibilnými databázami. Aplikácia už obsahuje prípravu na podporu tejto vlastnosti, ktorá mala byť implementovaná do základnej verzie, no kvôli časovému sklzu bola jej implementácia odložená.

Posledným z vylepšení programu XMLspark je neoddeliteľne úprava grafického užívateľského rozhrania, a to možnosťou otvorenia viacej dokumentov naraz, poprípade voľba nastavenia tlačítiek v toolbare alebo zgrupovanie veľkého počtu uzlov v strome z dôvodu lepšej prehľadnosti a podobne. Takýchto rozšírení možno nájsť nespočetné množstvo a v praxi by ich bolo nutné konzultovať s prípadným záujemcom, aby spĺňali jeho očakávania a potreby.

7 Literatúra

[1] KOSEK, Jiří. Vše o WWW : XML [online]. 1999-2006 [cit. 2009-05-09]. Dostupný z WWW: <http://www.kosek.cz/clanky/xml/xml-uvod.html>.

[2] Wikipedia/XML [online]. 2001- [cit. 2009-05-09]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Extensible_Markup_Language>.

[3] KOSEK, Jiří. Vše o WWW : Historie XML [online]. 1999-2006 [cit. 2009-05-09]. Dostupný z WWW: <http://www.kosek.cz/clanky/xml/xml-historie.html>.

[4] KOSEK, Jiří. Vše o WWW : Syntaxe XML [online]. 1999-2006 [cit. 2009-05-09]. Dostupný z WWW: <http://www.kosek.cz/clanky/swn-xml/syntaxe.html>.

[5] GOLDFARB, Charles F.. Roots of SGML [online]. 1996 [cit. 2009-05-10]. Dostupný z WWW: <http://www.sgmlsource.com/history/roots.htm>.

[6] KATOLICKÝ, Arnošt. Potřeba orjektového přístupu k implementaci XML [online]. 2000 [cit. 2009-05-10]. Dostupný z WWW: <http://www.volny.cz/akatolicky/XML_proj.htm>.

[7] KOSEK, Jiří. Vše o WWW : XML schémata [online]. 2004 [cit. 2009-05-10]. Dostupný z WWW: <http://www.kosek.cz/xml/schema/uvod.html>.

[8] Wikipedia : XML schema [online]. 2001- [cit. 2009-05-10]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/XML_Schema>.

[9] Relax NG [online]. 2001- [cit. 2009-05-10]. Dostupný z WWW: <http://www.relaxng.org/>.

[10] KOSEK, Jiří. Vše o WWW : Relax NG [online]. 2004 [cit. 2009-05-10]. Dostupný z WWW: <http://www.kosek.cz/xml/schema/rng.html>.

[11] Wikipedia : Relax NG [online]. 2001- [cit. 2009-05-10]. Dostupný z WWW: <http://en.wikipedia.org/wiki/RELAX_NG>.

[12] Konzorcium W3C [online]. 1994-2008 [cit. 2009-05-10]. Dostupný z WWW: <http://www.w3.org/>.

[13] Wikipedia : Java [online]. 2001- [cit. 2009-05-10]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Java_(programovací_jazyk)>.

[14] Wikipedia : Java Virtual Machine [online]. 2001- [cit. 2009-05-10]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Java_Virtual_Machine>.

[15] J. Měcháček. XML a Java. Zpravodaj ÚVT MU. ISSN 1212-0901, 2001, roč. XII, č. 2, s. 9-12.

[16] DOM4J [online]. 2001-2008 [cit. 2009-05-10]. Dostupný z WWW: <http://www.dom4j.org/>.

[17] Apache Xerces [online]. 1999-2009 [cit. 2009-05-10]. Dostupný z WWW: <http://xerces.apache.org/>.

[18] Jelinek, J. - Slavik, P. XML visualization using tree rewriting. Proceedings of the 20th spring conference on Computer graphics. ACM, 2004. s. 65-72. ISBN:1-58113-967-5.

[19] Petr Chmelar, Radim Hernych a Daniel Kubicek. Interactive Visualization of Data-Oriented XML Documents. Advances in Computer and Information Sciences and Engineering. Springer Netherlands, 2008. s. 390-393.

8 Prílohy

Príloha 1. Zoznam použitého software
Príloha 2. Štruktúra priloženého CD nosiča
Príloha 3. Užívateľská príručka k XMLspark
Príloha 4. CD

8.1 Zoznam použitého software

NetBeans IDE 6.5.1
Ant 1.7.1
GIMP 2.2
PSPad 4.5.0
MS Word 2003
PDFCreator 0.9.0.8

Implementované kompilované a testované na operačnom systéme:
MS Windows Vista Home Premium SP1 Version 6.0.6001

8.2 Štruktúra priloženého CD nosiča

`CD\Interaktivní vizualizace XML.pdf`	- technická správa bakalárskej práce
`CD\Desky – Martin Seko – BP.pdf`	- dosky technickej správy
`CD\README`	- popis inštalácie a spustenia
`CD\Java\`	- priložené inštalačné balíčky
`CD\XMLspark\`	- samotný program
`CD\XMLspark\XMLspark.jar`	- spustí aplikáciu XMLspark
`CD\XMLspark\build.xml`	- Ant skript na kompiláciu programu

8.3 Užívateľská príručka k XMLspark

Autor: Martin Seko

e-mail: xsekom00@stud.fit.vutbr.cz

Bakalárska práca

8.3.1 Čo je XMLspark?

- XMLspark je nástroj na interaktívnu vizualizáciu dátovo orientovaných XML dokumentov.
- XMLspark dovoľuje editovať text XML dokumentu a jeho schémy.
- XMLspark slúži ako generátor schém dokumentov v štyroch rôznych formátoch, a to: DTD, XML schema, RNG a RNC.
- Vizualizované dáta sú reprezentované vo forme stromu alebo tabuliek.

8.3.2 Popis ovládacích prvkov

Menu: **XML dokument**

Otvára zvolený XML dokument.

Uloží otvorený XML dokument.

Zavrie otvorený XML dokument, schému a ukončí vizulizáciu.

Menu: **XML schema**

Otvorí zvolené schéma dokumentu.

Uloží otvorené schéma dokumentu.

Zavrie otvorené schéma a zruší vizualizáciu. Ponechá však otvorený XML dokument.

Generuje schéma dokumentu (mód zobrazenia 1), alebo otvára XML dokument a generuje schému (mód zobrazenia 2).

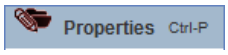
Menu: **Visualizer**

Spúšťa vizualizáciu otvoreného XML dokumentu a jeho schémy
V prípade módu zobrazenia 3 aj otvára XML dokument.

Ukončí spustenú vizualizáciu, zavrie XML dokument aj jeho schému.

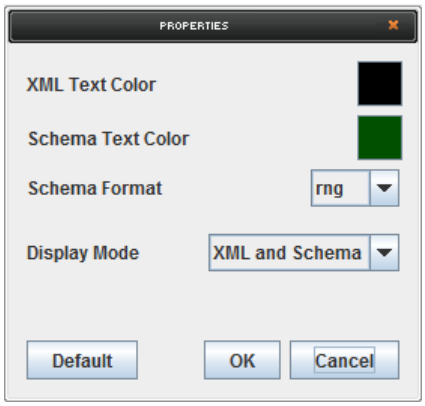
Export do SQL skriptu – neimplementovaná funkcia.

Menu: **File**



Zobrazí aktuálne nastavenia programu.

Nastavenia



Nastavenia najlepšie meniť hneď po štarte aplikácie, pretože ich zmena počas vizualizácie zruší niektoré otvorené alebo vizualizované prvky.

XML Text Color – farba zobrazeného textu XML dokumentu.

Schema Text Color – farba zobrazeného textu schémy.

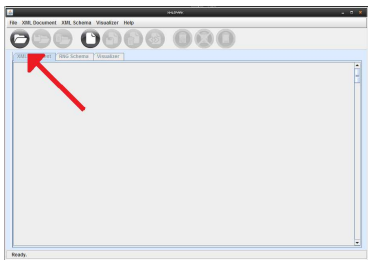
Schema Format – aktuálny formát schémy dokumentu.

Display Mode – aktuálny zobrazovací mód aplikácie

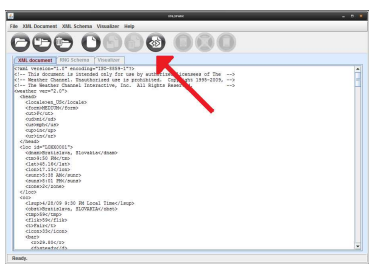
Správanie aplikácie v jednotlivých módoch popísané nižšie.

8.3.3 Činnosť aplikácie

Zobrazovací mód 1: **XML and Schema**

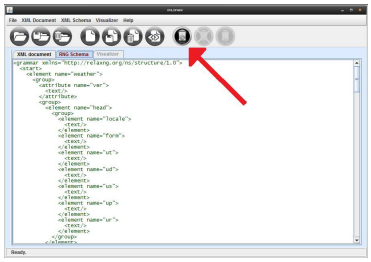


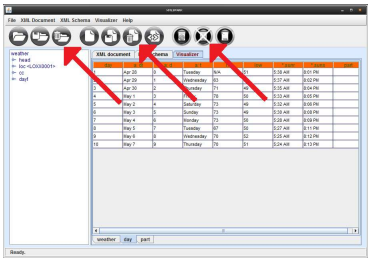
- Otvoríme XMLspark.
- Klikneme na “Open XML Document” (červená šípka).
- Vyhľadáme súbor, ktorý chceme otvoriť.
- Počkáme až sa súbor otvorí.



- Otvorením XML dokumentu sa sprístupnia ďalšie tlačítka.
- Klikneme na “Generate XML schema” (červená šípka) alebo otvoríme schému.

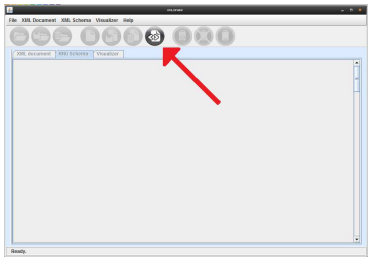
Doporučuje sa schému nechať vygenerovať, otvorená schéma nemusí byť validná!

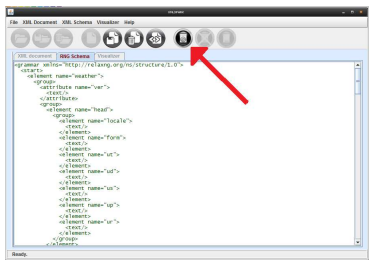


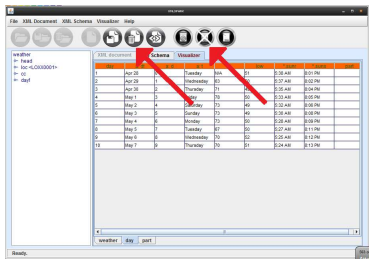


- Počkáme až sa vygeneruje a zobrazí schéma.
- Po vygenerovaní schémy sa sprístupní vizualizátor.
- Klikneme na “Visualize” (červená šípka).
- XMLspark postupne vygeneruje stromovú a potom tabuľkovú reprezentáciu XML dokumentu.
- Výsledkom vizualiácie bude strom na ľavej strane a tabuľky na pravej strane. Veľkosť ľavého a pravého okna je možno meniť stredovou priečkou.
- Vizualizáciu ukončíme kliknutím na ľubovoľné z troch tlačítiek “Close XML document”, “Close XML schema” alebo “End Visualization” (červené šípky).

Zobrazovací mód 2: **Schema only**





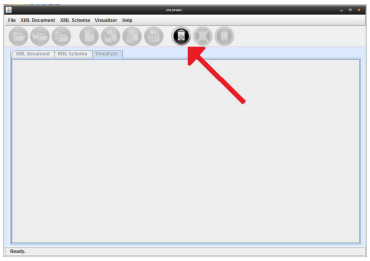


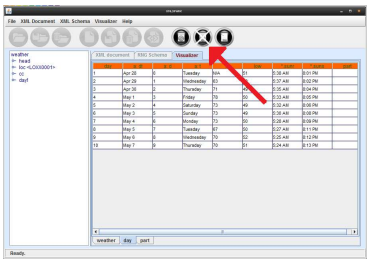
- Otvoríme XMLspark.
- Klikneme na “Generate XML schema” (červená šípka).
- Vyhľadáme XML dokument súbor, ktorý chceme otvoriť.
- Počkáme až sa súbor otvorí a vygeneruje sa jeho schéma.

Po zobrazení XML schémy sa sprístupní vizualizátor.

- Klikneme na “Visualize” (červená šípka).
- XMLspark postupne vygeneruje stromovú a potom tabuľkovú reprezentáciu XML dokumentu.
- Výsledkom vizualiácie bude strom na ľavej strane a tabuľky na pravej strane. Záložky pravého okna, už nedovolia zobraziť XML dokument.
- Vizualizáciu ukončíme kliknutím na ľubovoľné z dvoch tlačítiek “Close XML schema” alebo “End Visualization” (červené šípky).

Zobrazovací mód 3: **Visualization only**





- Otvoríme XMLspark.
- Klikneme na “Visualize” (červená šípka).
- Vyhľadáme XML dokument súbor, ktorý chceme otvoriť.
- Počkáme až sa súbor otvorí a vygeneruje sa jeho schéma a prebehne jeho úplná vizualiácia.
- Výsledkom vizualiácie bude strom na ľavej strane a tabuľky na pravej strane. Záložky pravého okna sú úplne statické a nedovoľujú prezeranie ani schémy ani XML dokumentu.
- Vizualizáciu ukončíme kliknutím na tlačítko “End Visualization” (červená šípka).