PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

BAKALÁŘSKÁ PRÁCE

Elektronická verze časopisu Falstaff

2011

Petr Švorčík

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval samostatně.

duben 2011 Petr Švorčík

Anotace

Práce si klade za cíl převést a zpřístupnit obsah školního časopisu Falstaff do digitální podoby a publikovat jej přes webové rozhraní, bez nutné znalosti použitých technologií.

Touto cestou bych rád poděkoval vedoucímu mé bakalářské práce Mgr. Aleši Keprtovi, Ph.D. za ochotu, odborné vedení, cenné rady a připomínky, které mi poskytl při zpracování této práce.

Obsah

Seznam tabulek

1. Úvod

Úkolem práce je vytvoření informačního systému s webovým uživatelským rozhraním, který zpřístupní články z časopisu Falstaff v elektronické podobě. Falstaff je studentský časopis, který před lety nepravidelně vycházel na Slovanském gymnáziu v Olomouci a obsahoval literární tvorbu různých žánrů. Na rozdíl od pravidelně vycházejících běžných časopisů či novin, jejichž www stránky vyžadují neustálé doplňování nového obsahu, Falstaff má podobu nepříliš často vycházejících sborníků. Z tohoto důvodu se pro jeho elektronické vydání moc nehodí tzv. redakční systémy používané v jiných časopisech.

Tato práce by se dala rozdělit do dvou samostatných částí:

1. Převod obsahu (vlastních článků a obrázků) do jednotného formátu.
2. Tvorba samotného uživatelského rozhraní.

Před převodem obsahu do digitální podoby byly k dispozici dva zdroje. Jednotlivá tištěná čísla časopisu a soubory pro redakci ve formátech používaných v dané době. Soubory byla směska počínající formátem pro AmiPro a konče u MS Office. Už na začátku byla zavržena varianta naskenovat jednotlivá čísla a následně je převést třeba přes nějaký ORC software do textového formátu. Důvodem byla kvalita jednotlivých čísel a také fakt, že kompletní soubor všech čísel v tištěné podobě by byl jen těžko k sehnání, pokud ho vůbec někdo vlastní.

Vzhledem k tomu, že úkolem je vytvoření webového rozhraní, jeví se jako nejlepším řešením převést všechny články do formátu XML. V čem autor práce vidí hlavní výhody XML?

1. Univerzální datový formát - relační databázové systémy jsou vhodné pro data, která lze „nenásilně" ukládat do tabulek. Neumí si však adekvátně poradit s komplexními dokumenty a hodně hierarchickými strukturami, což v praktickém životě není nic vzácného.
2. Význam dat je nedílnou součástí dat - XML dokumenty nepotřebují externí logiku, která by říkala, co který blok dat vyjadřuje. Vše je součástí XML dokumentu. Tato vlastnost nemusí být vždycky přínosem, protože si každý může pojmenovat značky vyjadřující význam dat, jak chce. V případě, že je dodržován domluvený, sjednocený formát XML dokumentu pro konkrétní oblasti, přinese to zjednodušení a možnost větší automatizace při zpracování.
3. Podpora internacionalizace - jazyk XML je od počátku navržen pro mezinárodní použití a podporuje standart Unicode.

U takto uložených dat není pak problém je zobrazit ve formě html nebo je jinak zpracovat. Každý XML soubor představuje jeden článek, který ve své hlavičce nese informaci o autorovi, zařazení článku, čísle časopisu a mnoho dalších důležitých informací. Tyto informace jsou použity pro tvorbu jednotlivých indexů, rejstříků a podobně.

1.1. Dokumentově orientované XML dokumenty

Tento typ XML dokumentů je většinou určen přímo pro čtení či zpracování lidmi. Charakterizuje je méně pravidelná či nepravidelná struktura, hrubá granulace dat a hojné používaní smíšeného obsahu. Pořadí, v jakém jsou za sebe psány jednotlivé elementy, je relevantní. Tyto dokumenty jsou nejčastěji psány ručně a to přímo v jazyce XML či například ve formátech RTF, PDF, atd., které jsou poté do XML zkonvertovány. Data těchto dokumentů obvykle nepocházejí z databází. Příkladem dokumentově orientovaných XML dokumentů mohou být např. knihy, reklamy, stránky ve formátu XHTML a další.

Dokumentově orientované XML dokumenty se ukládají do tzv. „nativních XML databázových systémů". Nativní XML databázový systém je systém, který je od základu navržen k ukládání a manipulaci s XML daty. Přístup k datům se odehrává skrze XML a související standarty, např. XPath, DOM či SAX. Pod tento pojem spadají všechny databázové systémy, jejichž reprezentace dat zachovává úplnou strukturu a související metadata. Formát uložení dat není důležitý a přístup k datům pomocí jiných než XML technologií není povolen. Základní jednotkou uložení dat v nativní XML databázi je XML dokument.

V nativních XML databázových systémech se pro zrychlení práce s obsahem užívají indexy. Indexy umožňují systému řízení báze dat rychleji vyhledat požadované místo v datech. Značného zrychlení je dosaženo především pokud chceme načíst celý XML dokument nebo jeho fragment. Na takovou operaci většinou stačí vyhledání dle jednoho indexu, předpokladá se totiž, že data XML dokumentu (či fragmentu) následují od vyhledané pozice.

2. Použité technologie

V této části jsou popsány principy technologií, na kterých je práce založena. Jedná se o XML, CSS a PHP.

2.1. XML

XML (eXtensible Markup Language) je standart vyvinutý konsorciem W3C, který byl navržen pro uchování dat a metadat. Jeho účelem je strukturovat, ukládat a předávat informace. XML nedefinuje přímo jednotlivé tagy, ale předepisuje, jak mají tagy vypadat a jakou mají mít strukturu. Záleží tedy na uživateli, jaké názvy si zvolí a jak je bude strukturovat. Uživatelem definované značky je možné definovat v XML schématu, případně jednodušším souboru DTD (Document Type Definition), pomocí kterého lze v parseru kontrolovat, zda daný dokument odpovídá definici.

V současné době se XML stává hlavním formátem pro výměnu informací. Jeho hlavní výhodou je multiplatformní nezávislost, jednoduchost a otevřenost. XML je textový formát, ve kterém lze použít různé znakové sady pro různé jazyky. Snadná je také konverze jiných formátů, ta se často využívá při zobrazování XML dokumentu. XML nemá prostředky pro definici vzhledu, který je nutné popsat některým stylovým jazykem, například XLS nebo CSS.
Deklarace XML:

<?xml version='1.0' encoding='znaková sada' standalone='yes—no'?>

XML dokument může obsahovat deklaraci XML. Ta musí být na začátku dokumentu. Povinným atributem je atribut *version*. Atributy *encoding* a *standalone* jsou nepovinné. Jejich pořadí je v rámci deklarace XML závazné.

Deklarace XML začíná posloupností znaků *<?xml* a končí posloupností *?>*. Všechny deklarace XML musí obsahovat aspoň atribut *version* s hodnotou 1.0. Znaková sada použitá v dokumentu může být specifikována v atributu *enconding*. XML dokumenty jsou ze své podstaty Unicode, a to i když jsou uloženy v jiném kódování. Doporučuje se používat názvy znakových sad registrovaných u IANA (Internet Assigned Numbers Authority).

Celý XLM dokument musí být v jednom kódování. Není tedy možné někde uprostřed změnit znakovou sadu.

2.1.1. Struktura XML schématu

XML schémata poskytují sadu vestavěných datových typů. Některé z těchto typů jsou primitivní typy popsané ve specifikaci, zatímco další jsou odvozené typy popsané ve schématech. Primitivní i odvozené typy jsou k dispozici autorům schémat, aby je použili, jak jsou nebo od nich odvodili nové.

Strukturované typy se používají k popisu elementů, se kterými jsou spojeny dětské elementy nebo atributy. Textové typy se používají k definici elementů, jenž mají pouze textový obsah, a k definici hodnot atributů. V případě textových typů obsahuje jazyk XML schémat také prostředky k určení typu elementů a atributů.

2.1.2. XML strom

Hlavní struktura XML dokumentu je stromová. Strom, který XML dokument vyjadřuje, čítá několik různých typů uzlů:

- element
- dokument
- instrukce pro zpracování
- komentář
- datový uzel
- atribut - lze jej zapisovat přímo do uzlu elementu nebo na nižší hierarchickou úroveň jako dětské uzly. V literatuře se lze setkat s oběma formami.

Uzel typu element
Element se skládá z jednoho z následujících výrazů:

<p a_1="A_1" ... an="A_n"> c_1 ... c_m </p>	pro n>=0, m>=0
<p a_1="A_1" ... a_n=" A_n"/>	pro n>=0

Tabulka 1. Výraz uzlu element

Element typu p obsahuje seznam dětí c_i a množinu atributů, které jsou tvořeny dvojicí jméno atributu a_i a hodnota atributu A_i. Typ elementu je identifikován jeho názvem. Na rozdíl od dětí, u atributů nezávisí na pořadí, v jakém jsou uvedeny. Stejný uzel tak může být linearizován různými výrazy. Všechna jména atributů a_i musí být různá, hodnoty atributů A_i nikoliv. C_i tvoří tzv. obsah elementu. Text mezi počátečním a koncovým tagem se nazývá obsah elementu *(Definice z [W3CXML1.1])*.

Druhý uvedený výraz je speciálním případem prvního výrazu pro *m=0*, tedy vyjádřením, že element nemá žádný obsah. V takovémto případě se element nezapisuje <*p*> </*p*>, i když je to přípustné, ale používá se zkrácená forma zápisu <*p* />. Dle této logiky se označují elementy jako „párové" či „nepárové", přičemž každý nepárový element lze formálně zapsat jako párový.

Typ elementu, názvy atributů a hodnoty atributů sestávají z textových znaků, mají však určitá omezení a pravidla:

- musí být tvořeny Unicode písmeny, číslicemi, pomlčkami a tečkami
- musí být dlouhé alespoň jeden znak

- musí začínat písmenem či pomlčkou
- jméno atributu nesmí začínat na *xml*

Pro hodnoty atributů nejsou definována žádná omezení, hodnotou atributu může být například i prázdný řetězec.

Jméno atributu *id* je rezervováno, tvoří tzv. identifikátor elementu. Více viz. *([W3CLink])*.

Uzel typu dokument

Jedná se o speciální případ elementu. Tento element je typu p (toto p musí být shodné s kořenovým elementem p uváděným výše) a neobsahuje žádné atributy. Může však obsahovat URL, která odkazuje na specializovaný datový model XML určený pro tento uzel a jeho děti. Obecná struktura uzlu typu dokument je tedy následující:

<!doctype p 'url'> c_1 *...* c_m *pro m>0*

Právě jeden prvek z c_i musí být uzel typu element, který je typu p. Pokud jsou na této úrovni nějaké další děti pak to mohou být pouze komentáře nebo instrukce pro zpracování; další datové elementy kromě p nejsou přípustné. Typ elementu p a volitelně uváděné URL jsou textové řetězce. Pro název typu elementu platí stejná omezení, jaká byla uvedena výše u uzlu typu element. Pro URL neplatí žádná další omezení.

Je přípustné, aby dokument neměl uveden typ. Kořenovým uzlem XML stromu může být anonymní uzel typu dokument, který nemá uveden typ ani URL. Takový dokument-uzel je v XML dokumentu reprezentován vynecháním výrazu *<!doctype>*. Jinak řečeno – pokud první výraz v XML dokumentu není *<!doctype...>*, pak má dokument anonymní kořen.

Uzel typu instrukce pro zpracování

Takový uzel je vždy listem stromu. Obsahuje pouze instrukci *i*. Instrukce je posloupnost nula či více znaků bez jakýchkoliv omezení kromě jednoho. Tímto omezením je, že nesmí začínat znaky *xml* (na velikostí písmen nezáleží), za kterými následuje mezera či nový řádek. (Instrukce začínající *xml* + bílé místo (whitespace) mají speciální význam. Pojmem „bílá místa"se označují znaky, které se objevují v textových řetězcích a které nemají viditelnou podobu. Jedná se o mezery, tabulátory a konce řádků.) Instrukce tedy vypadá následovně:

<i>

Instrukce se v XML dokumentech používají pro potřeby aplikací, například pro XML parser.

Uzel typu komentář

Komentář je obdobou instrukce. Jedná se o list stromu, který obsahuje pouze komentář *k*.

<!—k–>

Komentáře jsou na rozdíl od instrukcí určeny pro potřeby lidí (programátorů, uživatelů, atd.). Obvykle se do nich vepisují vysvětlení elementů, atributů, různé poznámky a další.

Uzel typu data

Datové uzly jsou listy stromu. Jejich účel je jediný, obsahují vlastní data. Vše, co v XML dokumentu není uzavřeno mezi znaky < a >, je pokládáno za data. Datový uzel nemůže být prázdný, musí obsahovat vždy alespoň jeden znak.

Nebezpečné znaky

V XML dokumentu nelze libovolně používat veškeré znaky. Určité znaky jsou pro dokument nebezpečné, a proto je třeba k jejich zápisu použít tzv. únikový mechanismus. Za nebezpečné se považují znaky, které by způsobily ukončení uzlu či syntaktickou nesprávnost dokumentu. Jedná se o znaky: *&*, *<*, *>*. Tyto znaky je třeba nahradit jejich znakovými entitami.

Znaková entita je ve tvaru *&#kód_znaku10* nebo *&#xkód_znaku16*. Kód znaku je v Unicode v prvním případě zapsán v desítkové soustavě, ve druhém případě v soustavě šestnáctkové. Pomocí znakových entit můžeme do dokumentu vložit jakýkoliv Unicode znak.

Nový řádek může mít tři formy: CR (carriage-return), LF (line-feed) nebo formu dvojznakové sekvence CRLF. Kterákoliv z těchto forem je nový řádek a je ignorována, vyskytuje-li se těsně před nebo za značkou.

2.1.3. XPath

Jazyk XPath vyvíjený konsorciem W3C slouží k adresování, výběru částí XML dokumentu. Tento jazyk se k přímému provádění operací příliš nepoužívá. Využívají ho však jiné jazyky a technologie, např. XSLT, SimpleXML. V současné době je poslední schválenou verzí XPath verze 1.0, rozpracovaná je verze 2.0.

XPath provádí své operace nad abstraktním datovým modelem XML. Pro každý XML dokument je v paměti počítače vytvořen strom, nad kterým XPath vyhodnocuje své výrazy. Mezi uzly stromu se lze pohybovat po různých osách.

XPath výraz může vracet následující hodnoty:

1. množinu uzlů
2. číslo

3. textový řetězec
4. logickou hodnotu (true/false)

Vybíráme-li uzly ze stromu vytvořeného z XML dokumentu, můžeme tyto uzly specifikovat následovně:

1. relativní cestou - od aktuálního uzlu či jiného specikovaného uzlu
2. absolutní cestou - od kořene stromu (začíná znakem /)

Cesta k uzlu se obecně skládá z několika částí. Cesta se vyhodnocuje zleva doprava, jednotlivé části cesty jsou odděleny znakem /. Cestu tvoří identifikátor osy, test uzlu a predikát.

Specifikace jazyka XPath poskytuje 13 os pro pohyb po stromu XML dokumentu:

osa	**význam osy**
self 1	aktuální uzel
ancestor	rodič aktuálního uzlu a jeho další předci až ke kořenu
ancestor-or-self	aktuální uzel, jeho rodič a další předci
attribute	uzly představující atributy elementu
child	děti aktuálního uzlu
descendant	potomci aktuálního uzlu
descendant-or-self	aktuální uzel a jeho potomci
following	uzly následující za aktuálním uzlem
following-sibling	uzly následující za aktuálním uzlem, které jsou sourozencem aktuálního uzlu
namespace	uzly odpovídající jmenému prostoru
parent	rodič aktuálního uzlu
preceding	uzly předcházející aktuálnímu uzlu
preceding-sibling	uzly předcházející aktuálnímu uzlu, které jsou sourozenci aktuálního uzlu

Tabulka 2. Osy v XML stromu

Testy uzlu se omezují na výběr uzlů na vybrané ose. Testovat lze název a typ uzlu. Můžeme tím vyjádřit, že vybírané uzly musí být určitého typu (elementy, textové uzly, komentáře ...), mít určitý název (např. „kapitola") nebo musí patřit do určitého jmenného prostoru.

Takhle vypadají příkladové XPath výrazy:

- attribute::autor - vybere atribut autor z aktuálního uzlu

- descendant-or-self::node() - vybere všechny potomky do libovolné hloubky

Některé často používané věci lze zapisovat zkrácenou formou. Například místo *atributte::* lze psát @, potomky do libovolné hloubky lze zapsat místo *descendant-or-self::node()* jako *//*.

2.2. CSS

CSS (Cascading Sytle Sheets) neboli kaskádové styly vznikly jako souhrn metod pro úpravu vzhledu stránek. První návrh normy byl zveřejněn v roce 1994, v roce 1996 byla pak vydána specifikace CSS 1, v roce 1998 CSS 2, nyní se pracuje na verzi CSS 3 (předpokladem dokončení je rok 2015).

CSS se využívá k formátování obsahu HTML, XHTML a XML dokumentů. Ve srovnání s formátováním pomocí atributů v HTML, CSS formátovací schopnosti rozšiřuje. Styly umožňují přesně určit, jak bude který element vypadat. Na rozdíl od atributů můžeme stylem definovat jednotný vzhled elementu pro celý dokument a to jediným zápisem pro příslušný element (nikoli v každém tagu příslušného elementu). Stejně tak můžeme pomocí stylu určit odlišné formátování pro třeba jen jediný výskyt určitého elementu. Tím se jednak zbavíme velkého množství kódu, jednak se tento kód stane mnohem přehlednější. Navíc pokud se jednou rozhodneme změnit například barvu písma všech odstavců, bude to pro nás otázka několika málo vteřin. Měnit každý atribut u každého elementu v HTML by byla katastrofa. Jeden styl můžeme navíc snadno použít pro libovolné množství stránek.

2.3. PHP

Zkratka PHP nebo-li Hypertext Preprocessor vznikla z původního názvu Personal Home Page. Jedná se o skriptovací jazyk navržený pro tvorbu dynamických internetových stránek. Používá se na straně serveru, ale může být použit také z příkazové řádky nebo GUI aplikace. Jazyk PHP je platformě nezávislý a dynamicky typový (hodnota proměnné se určí v momentě přiřazení hodnoty). Je to jeden z nejpoužívanějších jazyků pro tvorbu malých až středně velkých webů.

3. XML schéma článku

Vytvořené XML schéma může sloužit pro validaci XML souboru popisující článek. Definice XML schématu začíná definicí jednoduchých typů a atributů:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.w3schools.com" xmlns="http://www.w3schools.com" elementFormDefault="qualified">
  <!–Definice jednoduchých elementů–>
  <xs:element name="podpis" type="string" />
  <xs:element name="obrazek" type="anyURL" />
  <xs:element name="p" type="string" />
```

Nyní si uvedeme stručný popis toho, co jednotlivé elementy označují. Element <*podpis*> označuje podpis autora článku, element <*obrazek*> označuje odkaz na případný obrázek k danému článku, element <*p*> označuje nový odstavec v textu.

```
<!–Definice atributů–>
  <xs: attributeGroup name ="popisclanku">
    <xs:attribute name ="autor" type="xs:string" />
    <xs:attribute name ="autor1" type="xs:string" />
    <xs:attribute name ="autor2" type="xs:string" />
    <xs:attribute name ="titul" type="xs:string" />
    <xs:attribute name ="casopis" type="xs:string" />
    <xs:attribute name ="skupina">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="úvodník" />
          <xs:enumeration value="próza" />
          <xs:enumeration value="poezie" />
        </xs:restriction>
      </xs:simpleType>
    </xs:atribute>
    <xs: attribute name ="razeni" type="xs:integer" />
  </xs:attributeGroup>
    <xs: attributeGroup name ="odstavce">
      <xs:attribute name ="nadpis" type="xs:string" />
      <xs:attribute name ="zacatekodst" type="xs:string" />
      <xs:attribute name ="konecodst" type="xs:string" />
      <xs:attribute name ="tucne" type="xs:string" />
      <xs:attribute name ="podrzene" type="xs:string" />
      <xs:attribute name ="kurziva" type="xs:string" />
```

```
    <xs:attribute name ="modra" type="xs:string" />
    <xs:attribute name ="veta" type="xs:string" />
    <xs:attribute name ="vk" type="xs:string" />
    <xs:attribute name ="vpravo" type="xs:string" />
    <xs:attribute name ="center" type="xs:string" />
    <xs:attribute name ="podtrzene" type="xs:string" />
  </xs:attributeGroup>
```

Skupina atributů pojmenovaná *popisclanku* slouží k indexaci jednotlivých článků. Atributy *autor, autor1, autor2* označují jméno případně jména autorů článku, atribut *titul* označuje název článku pro rejstříky, obsahy, atd., které jsou většinou totožné s nadpisem článku, atribut *casopis* označuje skupinu, do kterého časopisu a čísla daného článku patří, atribut *skupina* obsahuje jednu ze tří hodnot označují typ literárního díla pro rejstříky a případné formátování textu, atribut *razeni* označuje pořadí článku v daném čísle časopisu. Další skupina atributů pojmenovaných *odstavce* slouží k vlastnímu formatování textu článku. Atribut *nadpis* je pro nadpis článku, atributy *zacatekodst* a *konecodst* označují začátek a konec odstavce v souvislosti s dalším formátováním. Atributy *podtrzene, kurziva, tucne* slouží prodefinování stylu písma. Atribut *modra* slouží k definici barvy písma. Atribut *veta* se používá v souvislosti se změnou barvy nebo stylu písma. Atributy *vpravo* a *center* označují formátování textu.

```
<!–Definice složených elementů–>
<xs:element name="konp">
  <xs:complexType>
    <xs:sequence>
      <xs: element ref="p" minOccurs=1 maxOccurs="unbounded" />
    </xs:sequence>
  </xs:compexType>
</xs:element>
```

Element <*konp*> se používá u poezie, která je formátována na střed a je potřeba ji doplnit o nějakou část textu formátovanou zleva, jako normalní text. Obsahuje jeden nebo více elementů <*p*>.

```
<xs:element name="textpodpis">
 <xs:complexType>
  <xs:sequence>
    <xs: element ref="p" minOccurs=1 maxOccurs="unbounded" />
  </xs:sequence>
 </xs:compexType>
</xs:element>
```

Element <*textpodpis*> slouží k doplnění textu k podpisu, který obsahuje jeden nebo více elementů <*p*>.

```
<xs:element name="basen">
 <xs:complexType>
  <xs:sequence>
    <xs: element ref="p" minOccurs=1 maxOccurs="unbounded" />
  </xs:sequence>
 </xs:compexType>
</xs:element>
```

Element <*basen*> slouží k vložení textu formátovaného na střed (např. básně), který obsahuje jeden nebo více elementů <*p*>.

```
<xs:element name="poznamka">
 <xs:complexType>
  <xs:sequence>
    <xs: element ref="p" minOccurs=1 maxOccurs="unbounded" />
  </xs:sequence>
 </xs:compexType>
</xs:element>
```

Element <*poznamka*> slouží k doplnění textu pokračujícím za podpisem, který obsahuje jeden nebo více elementů <*p*>.

```
<xs:element name="text">
 <xs:complexType>
  <xs:sequence>
    <xs: element ref="p" minOccurs=1 maxOccurs="unbounded" />
    <xs: element ref="basen" minOccurs=0 maxOccurs="unbounded" />
    <xs: element ref="konp" minOccurs=0 maxOccurs="unbounded" />
    <xs: element ref="textpodpis" minOccurs=0 maxOccurs="unbounded"
/>
    <xs: element ref="podpis" minOccurs=0 maxOccurs="unbounded" />
    <xs: element ref="obrazek" minOccurs=0 maxOccurs="unbounded" />
    <xs: element ref="poznamka" minOccurs=0 maxOccurs="unbounded"
/>
  </xs:sequence>
 </xs:compexType>
</xs:element>
```

Element <*text*> obsahuje jeden nebo více elementů <*p*> a nepovinně elementy <*basen*>, <*konp*>, <*textpodpis*>, <*podpis*>, <*obrazek*>, <*poznamka*>.

```
<xs:element name="clanek">
  <xs:complexType>
    <xs:sequence>
      <xs: element ref="text" minOccurs=1 maxOccurs="unbounded" />
    </xs:sequence>
  </xs:compexType>
 </xs:element>
</xs:schema>
```

Posledním je kořenový element <*clanek*>, který obsahuje jeden nebo více složených elementů <*text*>.

Příklad:

```
<?xml version="1.0" encoding="utf-8"?>
<clanek autor="Jan Přidal" titul="Lásku lidem Stvořitel daroval" casopis="f_3" skupina="próza" razeni="2" />
  <text>
   <p>Motto : Lásku lidem Stvořitel daroval, aby jim vynahradil to, že musí přemýšlet... </p>
   <basen>
     <p>Budiž, má lásko, moje,</p>
     <p>políbit chci Tě na rty Tvoje,</p>
     <p>podobáš se luční víle,</p>
     <p>pro niž půjdu všechny míle.</p>
     <p> </p>
     <p>Láska přetrvá i svět,</p>
     <p>překrásná jak sedmikrásky květ,</p>
     <p>čistá a jasná, bez poskvrny,</p>
     <p>jak růže s ostrými to trny.</p>
     <p> </p>
   </basen>
   <konp>
     <p>Věnováno My Lady - Dny bez Tebe jsou temné a pochmurné, nejkrásnější ze Sluncí...</p>
   </konp>
   <podpis>Jan Přidal</podpis>
  </text>
 </clanek>
```

4. Implementace jádra systému

Tato část popisuje zobrazování článků uložených v XML. Také je zde popsán princip vyhledávání, tvorba indexového souboru, vestavěný FTP klient.

4.1. Vytvoření indexového souboru

4.1.1. Procházení adresáře

Adresáře se prochází pomocí třídy, jejichž cílem je procházet soubory a adresáře na základě filtru definovaného regulárním výrazem.

```
<?php
$it=new DirectoryIterator(dirname(__FILE__));
foreach ($it as $itFile) {
  if (!$it->isDot()) {
    if (!$it->isDir()) {
      echo $itFile->getFileName()."<br/>";
    }
  }
}
?>
```

V konstruktoru *DirectoryIterator* se nastaví objekt na cestu k procházenému adresáři, použije se *dirname(__FILE__)*, což je aktuální adresář, kde se skript nachází. Pomocí cyklu se pak už jen prochází jednotlivé položky. Prvně se testuje funkce *isDot()*, zda nejde o položku typu "." nebo ".." (adresářové odkazy na nadřízený prvek). Pokud ano, pokračuje se na další položku cyklu. Další testovací podmínkou je *isDir()*. Pokud ano, pokračuje se na další položku. Pokud se dostane na soubor, volá se funkce *getFileName()*, která vrací název souboru. Kromě funkce *getFileName* je možné použít spousty dalších funkcí, například na zjištění cesty souboru, času vytvoření souboru apod.

K filtrování souborů se využívá vestavěná SPL třída *FilterIterator*. V konstruktoru je parametrem cesta k procházenému adresáři, přičemž voláme rodiče - tedy *FilterIterator* s vytvořeným objektem *DirectoryIterator*. Dále je předáván regulární výraz pro vyhledávání.

```
class FiltrAdresar extends FilterIterator {
private $_rx, $_soubory;
public function __construct($cesta, $regex=null, $soubory=0) {
```

```
        if (is_object($cesta)) parent::__construct($cesta);
         else parent::__construct(new DirectoryIterator($cesta));
          $this->_rx=$regex;
          $this->_soubory=$soubory;
    }

    public function accept() {
      $vnitrni=$this->getinnerIterator();
      if ($vnitrni->isDot()) return false;
      if ($this->_soubory) if ($vnitrni->isDir()) return false;
      if ($this->_rx) return ereg($this->_rx, $vnitrni->getFileName());
      return true;
    }

    public function key() {
     return $this->getvnitrniIterator()->getcestaname();
    }
}
```

Funkce *accept* vrací true pro objekty, které projdou filtrem. Omezení je dvojí. Na prvním řádku se vyřadí objekty typu "." a ".." pomocí porovnání funkce *isDot()* a na druhém řádku se používá k porovnání název souboru funkcí *ereg* s regulárním výrazem.

4.1.2. Vytvoření indexového souboru

Pomocí třídy pro procházení adresáře daného časopisu jsou do xml souboru pro každý článek zapsány informace o adresáři, názvu souboru, autorovi a kategorii článku. Tyto informace tvoří soubor *adresar.xml*, který se následně používá ve všech ostatních částech programu pro vyhledávání nebo zobrazení požadovaného článku.

4.2. Zobrazení článku

4.2.1. SimpleXML

Dokumenty XML mají hierarchickou strukturu tvořenou vnořením jednotlivých elementů. Hierarchické struktury lze v počítači reprezentovat mnoha způsoby, v poslední době je populární modelování pomocí objektů. Knihovna SimpleXML využívá právě tento způsob. Dokument XML načte celý do paměti, do struktury objektů, jejichž jména odpovídají názvům elementů zpracovávaného dokumentu. Díky tomu se pak velmi jednoduše přistupuje k jednotlivým informacím.

Pro vytvoření struktury objektů z dokument XML slouží funkce *simplexml_load_file()*, která jako parametr očekává jméno souboru ke zpracování. Výsledkem je objekt, který zastupuje celý dokument XML.

$xml = simplexml_load_file("<název článku>.xml");

Podelementy jsou přitom dostupné jako členské proměnné. K elementu *clanek* se tak dostaneme zápisem:

$xml->clanek;

Tímto způsobem můžeme v úrovni XML přeskočit několik úrovní a podívat se třeba na podpis.

$xml->clanek->podpis;

Tento zápis nám již rovnou vrátí podpis, protože element *podpis* už obsahuje jen text.

V případě, že element obsahuje další podelementy, není mapován na řetězec, ale na pole objektů, které reprezentují jednotlivé podelementy. K druhému odstavci se proto dostaneme zápisem:

$xml->clanek->text->p[1];

SimpleXML zpřístupňuje nejen elementy a jejich obsah, ale i atributy. Atributy jsou reprezentovány asociativním polem, které je dostupné na objektu odpovídajícího elementu. Např. atribut *autor* u elementu *clanek* získáme pomocí zápisu:

$xml['autor'];

Knihovna SimpleXML se hodí zejména pro zpracování menších dokumentů s jednoduchou a pravidelnou strukturou.

4.2.2. Výpis článku

Článek se formátuje použitím externího stylopisu - to je soubor *styl.css*, na který se stránka odkazuje tagem *<link>*. V souboru je umístěný stylopis. Hlavní výhoda je v tom, že na jeden takový soubor se dá nalinkovat mnoho stránek, takže pak všechny používají jednotné formátování uvedené v tomto souboru. Bližší popis toho, jak jsou jednotlivé tagy článku formátovány, je uveden v kapitole XML schéma článku.

4.3. Rejstříky

4.3.1. Výpis kategorií

Rejstřík umožní vyhledat články zařazené do jedné skupiny (např. úvodník, próza, atd.) podle klíčových slov uložených článků v XML souborech.

Nejprve se načte indexový soubor, z něho se načtou všechny elementy *skupina*. Následně jsou odstraněny všechny násobné výskyty a zůstanou použité kategorie článků.

4.3.2. Výpis článků v kategorii

Výpis článků v kategorii umožní vypsat všechny články patřící do určité kategorie.

Z indexového souboru se pro každý článek načtou informace o umístění souboru (elementy *adresa* a *soubor*) a skupina článku.

```
$xml1=simplexml_load_file("adresar.xml");
if (!IsSet($pocet)) $pocet=0;
foreach($xml1-¿xpath('//clanek') as $clanek) {
  $adresa = $clanek->adresa;
  $soubor = $clanek->soubor;
  $skupina = $clanek->skupina;
```

Z adresy a názvu souboru se složí adresa článku. Pokud článek odpovídá zadané skupině, tak je načten z daného článku element *podpis* a atributy elementu *clanek: autor* a *casopis*.

```
if ($skupina1 == $skupina){
$xml=simplexml_load_file("clanky/".$adresa."/".$soubor);
$autor=$xml["autor"];
$casopis=$xml["casopis"];
$cislo_cas=Str_Replace("f_","Falstaff ",$casopis);
$podpis=$xml->text->podpis;
```

Výpis je potom ve formátu: *<název článku> (<podpis autora>) - <číslo časopisu>*.

Hypertextový odkaz na název článku vypíše daný článek, odkaz podpis autora vypíše všechna díla autora a odkaz na číslo časopisu vypíše obsah celého čísla. Výpis je potom setříděn podle čísla časopisu a je zobrazeno 35 záznamů na stránku.

4.4. Falstaff

4.4.1. Výpis vydaných čísel časopisu

Výpis vydaných čísel časopisu umožní vypsat seznam publikovaných čísel časopisu. Z indexového souboru se načtou všechny elementy adresa. Následně jsou odstraněny násobné výskyty a výstupem je seznam vydaných čísel časopisu setříděný podle čísla časopisu.

4.4.2. Obsah čísla časopisu

Z indexového souboru se pro každý článek načtou informace o umístění souboru (elementy *adresa* a *soubor*).

Pokud článek odpovídá danému číslu časopisu, je vypsán element *podpis* a atribut *titul* elementu *clanek*. Výpis je potom ve formátu: *<pořadí článku> <název článku> (<podpis>)*.

Odkaz na název článku vypíše daný článek, v závorce je uveden odkaz na podpis autora, který je převeden na jméno autora, tzn., že odkazuje na všechny články publikované autorem bez ohledu na použitou přezdívku (někteří autoři publikovali pod více jmény). Výpis je setříděn podle pořadí článků v časopisu.

Na pravé straně je vypsán první článek daného čísla.

4.5. Autoři

4.5.1. Výpis autorů

Výpis autorů umožní vyhledat články napsané jedním autorem podle klíčových slov uložených článků v XML souborech.

Nejprve se načte indexový soubor, z něho se načtou všechny elementy *autor*. Následně jsou odstraněny všechny násobné výskyty. Nakonec zůstanou již jen jména autorů, setříděná podle abecedy.

4.5.2. Výpis článků autora

Výpis článků autora umožní vypsat seznam všech článků publikovaných daným autorem.

Z indexového souboru se pro každý článek načtou informace o umístění souboru (elementy *adresa* a *soubor*) a element *autor*. Pokud autor článku odpovídá vybranému autorovi, jsou vypsány atributy *autor, titul, casopis* elementu *clanek*. Výpis je potom ve formátu: *<název článku> (<číslo časopisu>)*.

Hypertextový odkaz na název článku vypíše daný článek, v závorce je informativně uvedeno číslo časopisu.

Výpis je setříděn podle čísla časopisu.

4.6. Administrace

Tato část obsahuje jednoduchého klienta doplněného o odkaz na vytvoření indexového souboru přímo na ftp serveru.

4.6.1. FTP klient

Při změně stavajícího čísla či publikací nového čísla časopisu je třeba vždy znovu vytvořit nový indexový soubor. Standartně by šlo tento soubor vytvořit pomocí systémových funkcí PHP, ovšem u většiny poskytovatelů jsou tyto funkce z bezpečnostních důvodů zakázany. Jedná se např. o funkci *chmod()*. Ruční nastavení práv pro přepsání indexového souboru by bylo pro většinu uživatelů nepraktické, takže se využívají FTP funkce jazyka PHP. Tyto funkce přistupují k souborům přes protokol FTP, který nebývá nijak omezen.

Současně je tohle rozhraní možné využít i pro úpravu stávajících čísel či pro publikaci čísel nových. Tato část se skládá z následujících souborů:

1. zapis.php - hlavní skript provádějící požadované operace se soubory a adresáři, který se také stará o výpis obsahu aktuálního adresáře
2. proftp.php - soubor s definovanými funkcemi, kterými jsou např. zjištění aktuálního adresáře, data vytvoření souboru, velikosti, atd.
3. nastaveni.php - konfigurační soubor (login na FTP server, cesta k adresáři s ikonami atd.)
4. administrace.php - zobrazuje stránku pro zadání přihlašovacích údajů

Po zavolání stránky administrace.php se zobrazí přihlašovací formulář a po jeho odeslání se adresa FTP serveru, číslo portu, přihlašovací jméno a heslo uloží do session proměnné a provede se přesměrování na stránku zapis.php, která se postará o přihlášení k FTP serveru a zobrazí obsah aktuálního adresáře. Také se zobrazí formuláře pro zadání jména nového adresáře a pro zadání nebo vybrání jména souboru, který budeme chtít nahrát na server. Dále je nabídnut odkaz pro odhlášení a vytvoření indexového souboru.

4.7. Fulltextové vyhledávání

Fulltextové vyhledávání funguje na základě boolovského modelu vyhledávání slov v XML souborech. Vzhledem k tomu, že se jedná o poměrně malé množství dokumentů (desítky až stovky), funguje tak i takto jednoduchá implementace dostatečně rychle.

Závěr

Cílem této bakalářské práce bylo převést vydaná čísla časopisu Falstaff do formátu umožňujícího elektronické zpracování a vytvoření uživatelského rozhraní pro jejich zobrazení a případnou publikaci nových čísel.

Pro splnění stanoveného cíle bakalářské práce bylo nejdříve potřeba převést veškeré články do formátu XML, dále vytvořit XML schéma pro kontrolu převedených článků a toto schéma nakonec použít i jako vodítko pro případnou tvorbu nových čísel časopisu. Důraz byl také kladen na použití technologií, které jsou dostupné na většině free webhostingů, neboť u podobných projektů se nepředpokládá komerční nasazení. Také vzhled byl optimalizován směrem na dobrou čitelnost článků a co nejjednodušší rozhraní bez zbytečných grafických kreací. Přece jenom zde šlo hlavně o obsah jednotlivých čísel. Nicméně stávající šablona je velmi lehce modifikovatelná díky CSS stylům.

Conclusions

The aim of this bachelor's diploma thesis is to convert some issues of Falstaff magazine to a data format enabling electronic processing and user interface creation needed for their display or for possible publication of new issues of the magazine.

To fulfil the aim of the thesis, it was necessary to convert all the articles to XML format first. Next, XML scheme for the control of the converted articles was created and this scheme was used as a guideline for a possible production of new issues of the magazine. The use of technologies available on most free web hostings was also emphasized because projects like these are not supposed to be used commercially. Appearance was also optimized to enable good legibility and simple interface, without useless graphic creations. After all, it was the content of the individual issues which was in the centre of our attention. The current template is easily modifiable thanks to CSS styles.

Reference

[1] Keprt Aleš. *Digitalizace časopisu Falstaff.* Ve sborníku: Tvorba softwaru 2005. Tanger s.r.o., Ostrava, 2005, pp.69-75, ISBN 80-86840-14-X.

[2] Skonnard Aaron, Gudgin Martin. *XML pohotová referenční příručka.* Grada, 2006.

[3] Lavin Peter. *PHP objektově orientované.* Grada, 2009.

[4] Kosek Jiří. *XML pro každého.*

[5] Vanický Miroslav *XML a databáze.* 2004.

[6] XML Schema W3C 1996-2010. *http://www.w3.org/XML/Schema*

[7] XML Path Language (XPath), W3C1999. *http://www.w3.org/TR/xpath*

[8] Cascading Style Sheet, W3C 2010. *http://www.w3.org/Style/CSS*

[9] Extensible Markup Language (XML) 1.0, W3C 2006. *http://www.w3.org/TR/xml*

A. Manuál pro použití

Instalace

Pro instalaci stačí rozbalit přiložený soubor *falstaff.zip* do domovského adresáře vašeho serveru nebo webhostingu (server musí obsahovat PHP alespoň verze 5 a FTP server) a spustit script *install.php*, který ve 3 krocích nastaví cestu k adresáři s články a přihlašovací údaje k FTP serveru.

Vytvoření nového čísla časopisu

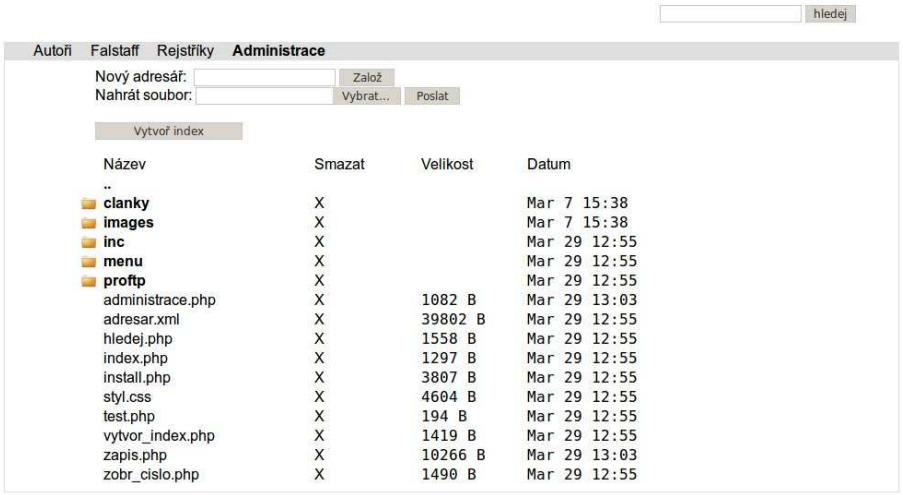
Nejprve se musí vytvořit nový adresář. Zadejte název adresáře a stiskněte tlačítko *založ*. V případě časopisu Falstaff adresář začíná písmenem *f* a následuje číslo časopisu.

Nahrátí nového článku

Vyberete požadovaný adresář. Vybere se soubor s novým článkem přes tlačítko *vybrat* a stiskněte tlačítko *poslat*. Soubor musí být ve formátu XML vytvořený podle XML schématu popsaného v kapitole 3.

Obnovení indexového souboru

Po nahrátí, smazání nebo úpravě článku je nutné obnovit indexový soubor stiskem tlačítka *vytvoř index*, aby se změna projevila.



ukázka administračního menu

B. Obsah přiloženého CD

`bin/`
: Falstaff v ZIP archivu pro zkopírování na webový server. Adresář obsahuje i všechny články časopisu.

`doc/`
: Dokumentace práce ve formátu PDF vytvořená dle závazného stylu KI PřF pro diplomové práce, včetně všech příloh, a všechny soubory nutné pro bezproblémové vygenerování PDF souboru dokumentace (v ZIP archivu), tj. zdrojový text dokumentace, vložené obrázky, apod.

`src/`
: Kompletní zdrojové texty webové aplikace Falstaff se všemi potřebnými zdrojovými texty a adresářové struktury pro zkopírování na webový server.

`readme.txt`
: Instrukce pro nasazení webové aplikace Falstaff na webový server včetně požadavků pro její provoz a webová adresa, na které je aplikace nasazena pro testovací účely a pro účel obhajoby práce.