Spletna aplikacija za generiranje poljubno strukturiranih datotek

Črtomir Drofenik, Borko Bošković, Janez Brest

Univerza v Mariboru, Fakulteta za eletrotehniko, računalništvo in informatiko
Inštitut za računalništvo
Smetanova ulica 17, 2000 Maribor, Slovenija
E-pošta: crtomir.drofenik@gmail.com, {borko.boskovic,janez.brest}@uni-mb.si

Web application for generating custom structured documents

The web application offers users an easier way to create specific files for their individual needs. It extracts data from different data sources (for example Microsoft Excel, Word, Open Office Writer, Calc, etc.) and converts it into a unique data structure. The user can manipulate the "template" file in which he specifies the position and value of data, collected from the previous data source. The template engine then compiles and returns a custom made document. This application is capable of creating custom xml, html and even latex files. It is designed to support new implementations of data sources so that it's up to date with the most popular data sources.

1 Uvod

V članku opisujemo spletno aplikacijo, namenjeno izdelavi poljubno strukturiranih dokumentov iz obstoječega podatkovnega vira. Ta vir je lahko različnega izvora (npr. Microsoft Word, Excel, OpenOffice Writer, Calc, SQL baza, itd.). Poudarek naše rešitve je predvsem na združevanju vseh podatkovnih virov v enotno podatkovno strukturo. Tako lahko preprosto in hitro dostopamo do najrazličnejših podatkov znotraj le-te.

V nadaljevanju je podrobneje predstavljena struktura in opis sestavnih delov ter njihovo delovanje. Opisan je tudi postopek obdelovanja različnih podatkovnih virov, kakor tudi generiranja poljubnih dokumentov glede na podano obliko. To definiramo znotraj predloge, ki jo mora uporabnik zapisati.

Aplikacija je zapisana v programskem jeziku Java. Ogrodje spletne strani je zapisano v JSF (JavaServer Faces [1]) . Vsebuje tudi poseben prevajalnik, ki se izvede nad podano predlogo. Zato je potrebno poznavanje principov generiranja kode (v našem primeru podatkov), ki so opisani v [2] in [3], kakor tudi posebnih potreb predloge.

Zaradi vsestranske strukture aplikacije je nadgradnja oz. dodajanje novega paketa za obravnavo želenega podatkovnega vira zelo preprosta.

2 Model in struktura aplikacije

Arhitektura aplikacije je zasnovana tako, da uporabnik poda podatkovni vir in izbere želeno obliko končne datoteke (npr. html, xml itd.).

Na začetku se vsi podatki znotraj datoteke preberejo in prestavijo v enotno podatkovno strukturo, ki je bila ustvarjena za ta namen. Ta se doda na predlogo, ki jo je uporabnik ustvaril, nato se s preprostimi ukazi sklicuje na prej zbrane podatke. Na koncu se tvori želena datoteka poljubne oblike.

Sama aplikacija omogoča preprosto dodajanje paketov za obravnavo novih podatkovnih virov. S tem je zagotovljena sprotna ažurnost aplikacije s specifičnimi željami uporabnika, kakor tudi hitra prilagoditev poljubnemu tipu datoteke. Za učinkovitejšo uporabo in razumevanje je potrebno znanje ključnih komponent aplikacije, ki jih predstavlja model (Slika 1).

Slika 1: Model strukture aplikacije.

2.1 Podatkovni viri

Posamezniki ali podjetja tvorijo najrazličnejše datoteke, v katerih predstavijo podatke. Najbolj popularni programi za tvorjenje takšnik datotek so predvsem Word in Excel paketa Microsoft Office ter Writer in Calc iz paketa OpenOffice oz. LibreOffice. Mnogokrat se podatki hranijo tudi v podatkovnih bazah. Vsi ti podatki so predstavljeni na različne načine in tudi dostop do njih je zaradi tega različen. Z izjemo podatkovne baze morajo biti datoteke za potrebo te aplikacije v določeni obliki, zato da se lažje definira dostop do vsebovanih podatkov.

2.2 Predloga

V prilogi se definira oblika ter kje in kako se bodo prikazali podatki, ki smo jih prevzeli iz podatkovnega vira. Zanj aplikacija vsebuje tudi poseben prevajalnik, s katerim manipuliramo podatke, razpoznane iz prej podanega podatkovnega vira. Sposobnosti in omejitve predloge so podrobneje opisane v 4. poglavju.

2.3 Jedro

Temelji aplikacije so zapisani v jedru. To je razdeljeno na notranje razrede in vmesnike ter zunanje pakete, s katerimi preoblikuje podatkovne vire v lastno podatkovno strukturo (Slika 2). Znotraj jedra se prevede tudi predloga, seveda samo, če je bila podana v pravilni obliki.

Slika 2: Jedro aplikacije.

2.4 Izhod aplikacije

Dokument, ki je bil generiran iz podatkovnega vira in oblikovan po vzorcu predloge, predstavlja izhod aplikacije. Ta je lahko shranjen kot tipične strukturirane datoteke, npr. xml ali html. Pomembno je predvsem pravilno poznavanje dostopov do obdelanih podatkovnih virov, kakor tudi poznavanje delovanja prevajalnika za predlogo. Omenjeni zadevi sta podrobneje opisani v nadaljevanju članka.

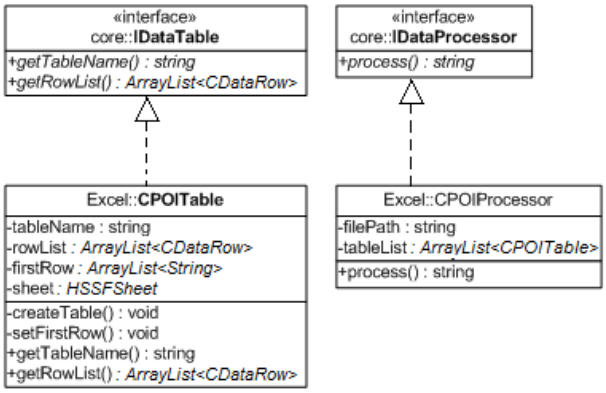
3 Obdelava podatkovnih virov

Podatki, ki so znotraj nekega podatkovnega vira, morajo biti primerne oblike, da jih je možno pravilno razčleniti in porazdeliti v aplikacijsko specifično podatkovno strukturo. V naslednjem podpoglavju je opisan primer, kako naša aplikacija tvori podatkovno stukturo in s čim si pomaga.

3.1 Primer obravnave preglednice

Uporabnik mora v primeru Microsoft Office Excel določiti prvo vrstico kot imena posameznih stolpcev. Pri razpoznavanju podatkov iz podatkovne baze to ni potrebno, saj se že s povpraševalnim (query) stavkom določijo posamezni stolpci. Tvorba prve vrstice je ključna, saj vso povpraševanje s strani predloge prihaja s klicom imena stolpca. Podatkovna struktura vrstice je zato izdelana tako, da vsebuje imena vseh stolpcev v podatkovnem viru in jih tako pretvarja v indekse številke, saj so vsi podatki znotraj vrstice (*CDataRow*) predstavljeni kot nizi v seznamu večih polj (ArrayList). Prednost uporabe prav te podatkovne stukture je podrobneje opisana v [4].

Pri vsem tem razpoznavanju podatkov sta pomembena vmesnika *IDataProcessor in IDataTable*, vsebovana znotraj centrale razredov in vmesnikov. V njih so zapisane vse potrebne metode za dostop do podatkovne stukture. Zato ju mora vsak paket, ki bi naj bil dodan, implementirati tako, kot je razvidno iz slike 3.



Slika 3: Razredni diagram za paket Microsoft Office Excel.

Na sliki sta predstavljena vmesnika in paket za obravnavo Microsoft Office Excel dokumenta, kar pomeni, da bo imela aplikacija opravka s preglednico. V metodi *process()* je definiran dostop do podatkovnega vira in tvorjenje primerkov *CPOITable***.** Ti kot parameter prejmejo list preglednice (sheet). Zaradi podatkovne strukture *CDataRow* se mora znotraj tabel določiti prva vrstica, zato razred *CPOITable* vsebuje metodo *setFirstRow()*. Nato se izvede *createTable()*, kjer se aplikacija sprehodi skozi vse neprazne celice in tvori seznam vrstic *rowList*. Pri tem si pomaga s knjižnico, ki jo je izdelalo podjetje Apache ([5]). Znotraj tega razreda sta implementirani tudi metodi *getTableName()* in *getRowList()*, ki pa se kličeta samo, kadar ju potrebuje prevajalnik za predlogo. Na koncu tvorjenja podatkovne strukture metoda *process()* znotraj *CPOIProcessor* vrne tekstovni del tako, da se mora samo še poslati datoteka v želenem podatkovnem viru (xml, html itd.).

3.2 Dodajanje novega paketa

Iz prejšnjega podpoglavja je razvidno, da je priprava podatkovne stukture precej preprosta. Programer bi na podoben način lahko definiral želene pakete za obdelavo različnih virov. Poudarek je predvsem na tem, da se implementirajo abstraktne metode obeh glavnih vmesnikov (*IDataProcessor* in *IDataTable*). Te so ključne za dostop prevajalnika predloge do ustvarjene podatkovne strukure. Odvisno je tudi, na kakšen način je sestavljen podatkovni vir. Če bi uporabnik želel razpoznati podatke iz xml datoteke, potrebuje samo en primerek tabele. V primeru Excela to ni mogoče, saj lahko vsak list znotraj preglednice vsebuje različne podatke, do katerih mora biti zagotovljen dostop. Kadar dostopamo do podatkovne baze, velja enako kot za xml, saj lahko s povpraševanjem uporabnika zgradimo samo eno tabelo.

4 Prevajanje predloge

Predloga, ki jo tvori uporabnik, mora biti napisana po določenih pravilih programskega jezika, ki se uporablja za obdelavo tega tipa dokumenta. Vse informacije o sintaksi in sposobnostih pogona so predstavljene v [6].

Posebnost predloge v tej aplikaciji je ta, da ima na sam kontekst dodano podatkovno strukturo za manipulacijo podatkov, omenjeno v prešnjem poglavju. Ta je lahko predstavljena na različne načine, odvisno od vira, ampak mora vsebovat enoličen dostop. V primeru MS Excel datoteke je podatkovnih struktur toliko, koliko je v sami datoteki listov (sheet). Vsak od teh je dosegljiv z lastnim imenom, s katerim je označen znotraj datoteke. Primer sklicevanja na podatkovno strukturo znotraj predloge je takšen:

```
$List1
```

To sklicevanje kliče metodo *getRowList()*, ki smo jo spoznali v prejšnjem poglavju. Da bi jo učinkovito uporabili, moramo napisati sledečo kodo:

```
#foreach ( $vrstica in $List1)
   obdelava vrstic
#end
```

Prevajanje tega sklopa predloge nam omogoča prehod čez celotno podatkovno strukturo. Da pa bi dostopali do posameznega podatka znotraj vrstice, moramo vedeti, kako se imenuje stolpec, v katerem je podatek, ki ga želimo prikazati. Primer testnega dokumenta MS Excel je viden na sliki 4.

	A	B	C
1	NAZIV	NASLOV	DRZAVA
2	Janez Novak	Razlagova ulica 22	Slovenija
3	Jože Horvat	Dunajska cesta 4	Slovenija
4	Zlatko Kovačič	Trubarjeva ulica 23	Slovenija
5			

Slika 4. Primer preglednice MS Excel.

Kot je razvidno iz dokumenta, morajo imeti tudi podatkovni viri določeno strukturo, da je podatke možno na določen način razvrščati. Primer izpisa podatkov iz prvega stolpca je naslednji:

```
#foreach ( $vrstica in $List1)
   $vrstica.get(»NAZIV«)
#end
```

Moč predloge je predvsem ta, da je možno dodajanje znakov, ki se ne bodo prevajali kot del jezika, ampak jih bo le-ta preskočil. Primer izpisa podatkov v neki poljubni obliki:

```
#foreach ( $vrstica in $List1)
   <oseba>
      <naziv>$vrstica.get(»NAZIV«)</naziv>
      <naslov>$vrstica.get(»NASLOV«)</naslov>
   </oseba>

#end
```

Ob prevajanju zgornjega dela predloge bi končna datoteka izgledal tako:

```
<oseba>
   <naziv>Janez Novak </naziv>
   <naslov>Razlagova ulica 22</naslov>
</oseba>

<oseba>
   <naziv>Jože Horvat</naziv>
   <naslov>Dunajska cesta 4</naslov>
</oseba>

<oseba>
   <naziv>Zlatko Kovačič</naziv>
   <naslov>Trubarjeva ulica 23</naslov>
</oseba>
```

Iz primera je razvidno, da je končna oblika zgrajena kot xml. Če pa bi bilo potrebno podatke pokazati v obliki HTML, pa bi v predlogo napisali naslednje:

```
<html>
<head>
   <title>Generirana tabela</title>
</head>
<body>
<table border=1>
#foreach ( $vrstica in $List1)
 #if($vrstica.getRowNumber() == 0)
 <tr>
      <td>NAZIV<td>
      <td>NASLOV<td>
 </tr>
 <tr>
      <td>$vrstica.get(»NAZIV«)</td>
      <td>$vrstica.get(»NASLOV«)</td>
 </tr>
 #{else}
 <tr>
```

```
        <td>$vrstica.get(»NAZIV«)</td>
        <td>$vrstica.get(»NASLOV«)</td>
      </tr>
    #end
  #end
  </table>
  </body>
```

Dokument, ki bi bil ustvarjen ob prevajanju zgornje predloge, bi znotraj brskalnika zgledal tako, kot je prikazano na sliki 5.

NAZIV	NASLOV
Janez Novak	Razlagova ulica 22
Jože Horvat	Dunajska cesta 4
Zlatko Kovačič	Trubarjeva ulica 23

Slika 5: Izgled generirane tabele znotraj brskalnika.

S to spremembo je razvidna tudi uporaba if-else stavka, za katerega je seveda potrebno poznavanje razreda *CDataRow*. Ta vsebuje poleg *getRowNumber()* tudi metodo *getDate()*, s katero se izpiše datum in čas kreiranja dokumenta. Z njima si programer lahko pomaga pri gradnji oblike končnega dokumenta.

5 Zaključek

V današnjem času je ogromno podatkovnih virov (npr. .doc,.xls, .odt itd.) in tudi podjetja imajo včasih povsem svoje tipe virov, ki so oblikovani po njihovih kriterijih. Naša rešitev je predvsem zaradi prej naštetih razlogov narejena zelo vsestransko in enostavno zato, da je možno hitro in preprosto dodati nov paket za obravnavo specifičnega podatkovnega tipa. S tem bi posamezniku ali podjetju pomagali pri pretvarjanju podatkovnih virov iz ene strukture v drugo.

V članku je bila predstavljena spletna aplikacija za generiranje poljubno strukturiranih datotek. V njem smo na grobo predstavili strukturo delovanja in opis posameznih delov. Z obdelavo podatkovnih virov smo pokazali eno od možnih idej za lažje raspoznavanje in obravnavanje podatkov, shranjenih v najbolj popularnih datotekah. Poudarek je bil na zasnovanih vmesnikih, ki so temelji aplikacije. Z njihovo implementacijo lahko na preprost in enostaven način dodamo še dodatne pakete za razpoznavanje novih podatkovnih virov. Podrobneje je opisan tudi prevajalnik predloge, ki jo tvori uporabnik. Z njim je možno izvajati različne funkcije, ampak le ob upoštevanju sintaktičnih in semantičnih pravil.

Literatura

[1] D. Geary, C. Horstmann, Core JavaServerFaces, 2007 (Sun Microsystems, Inc.)

[2] V. Žumer, M. Mernik. Principi programskih jezikov – študentsko gradivo, 2001.

[3] J. Herrington. Code Generation in Action, 2003 (Manning Publication Co.)

[4] M. N. Maurice, P. Wadler. Java Generics and Collections, 2006 (O'Reilly Media).

[5] A. Vohra, D. Vohra. Pro XML Developement with Java Technology, 2006 (Apress)

[6] J. D. Gradecki, J. Cole. Mastering Apache Velocity, 2003 (Wiley Publishing, Inc.)