Tradutor de documentos XML para Ontologia em triplos RDF

Relatório de Desenvolvimento

Universidade do Minho
Licenciatura em Ciências da Computação
3ºano-Projeto

Pedro Dias – 63389
Diogo Ferreira – 62154

Índice

1-Introdução

No âmbito da unidade curricular Projeto foi-nos sugerida a realização de um trabalho prático com o principal objetivo a tradução de ficheiros XML para ficheiros RDF (XML2RDF).

Este objetivo enquadra-se no projeto Museu Pessoa que tem como principal objetivo a recolha de historias de vida do povo.

As entrevistas realizadas as pessoas são guardadas em um repositório e divididas em quatro ficheiros XML, estes são:

- Entrevista Original
- Entrevista Editada
- BI
- Legenda das fotos

Neste projeto iremos apenas tratar os ficheiros *Entrevista Editada, BI e legenda das fotos*. A imagem em baixo ilustra o projeto *Museu Pessoa*, sendo a parte rodeada a vermelho o objetivo deste trabalho pratico.

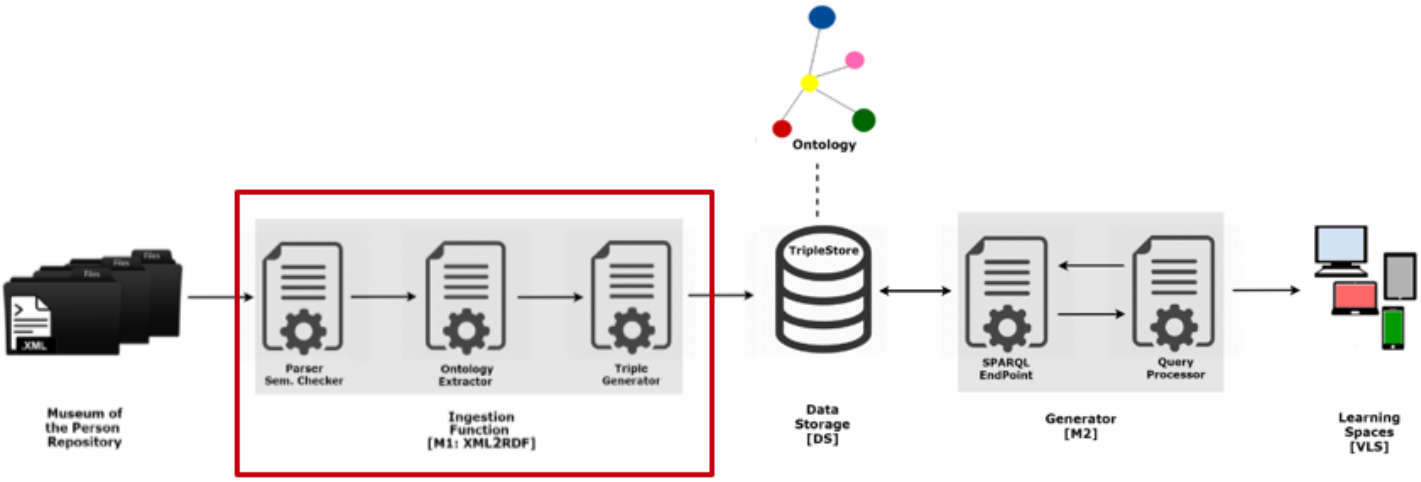

Ilustração 1 - Projeto Geral Museu Pessoa

2-Análise e Especificação

2.1-Descrição informal do problema

Foi-nos proposto o desenvolvimento de um filtro de texto capaz de ler de três ficheiros do formato XML (Entrevista Editada, BI e Legenda das fotos) e posteriormente a transformação dos mesmos em apenas um ficheiro do formato RDF.

2.2-Dados

Nos documentos XML, são-nos dadas palavras marcadas ao longo de todo o texto.

Essas marcações variam entre vários tipos. De entre muitas marcações temos por exemplo as seguintes (relativamente aos ficheiros de legenda de fotos):

- <foto ficheiro="XX-YY-ZZ.jpg"> - indica o nome do ficheiro
- <quem> AABBCC </quem> - indica quem se encontra na foto
- <quando data="XXXX-YY-ZZ" /> - indica qual a data da foto

2.3-Pedidos

Neste Trabalho é nos pedido para transformar as marcações XML em cima referias em marcações em RDF.

Foi-nos sugerido a utilização da linguagem ***ANTLR*** (ANother Tool for Language Recognition), da linguagem ***JAVA*** e a utilização da ferramenta ***ANTLRWorks***(2.1), para filtrar com recurso a expressões regulares todos os marcadores presentes no documento que nos possam ajudar a resolver o problema.

3- Conceção/desenho da Resolução

3.1-Estruturas de Dados

Para a realização deste trabalho foi preciso dividir o mesmo em 3 ficheiros:

- MainLexerBI2RDF.java
- BImaisLegendas.g4
- Person.java

O ficheiro MainLexerBI2RDF.java serve apenas para pegar no output do ficheiro BImaisLegendas.g4 e escrever no ficheiro RDF final.

O código do ficheiro BImaisLegendas.g4 foi por sua vez dividido em 3 partes:

- Código referente ao BI
- Código referente as Legendas
- Código referente as Entrevistas editada

O ficheiro Person.java serve para guardar os dados da pessoa entrevistada (profissão, episódios, imagens, etc) para posteriormente conseguir escrever no output.

3.2- Algoritmos

Iremos apenas expor neste relatório o código referente as legendas para melhor compreensão do trabalho em geral. O código referente as Entrevistas Editadas e BI é parecido, com pequenas diferenças.

O "mode sSAVE" serve para guardar no ficheiro "info.ser" as informações relativas as variáveis countinterview, counttheme e ProjectMap.

```
mode sFOTOS;
GetSFOTOS   :  '<foto'   -> mode(sFOTO)
           ;
OutFOTOSSAVE  : '</' ->mode(sSAVE)
           ;
DefaultsFOTOS:   .                                    { ; }
            ;

mode sSAVE;
GetSAVEFOTOS  : 'fotos'   {try{
                                                FileOutputStream fileOut = new FileOutputStream("info.ser");
                                                ObjectOutputStream out = new ObjectOutputStream(fileOut);
                                                out.writeObject(countinterview);
                                                out.writeObject(counttheme);
                                                out.writeObject(ProjectMap);
                                                out.close();
                                                fileOut.close();
                                                System.out.printf("Serialized data is saved in /tmp/info.ser");
                                           }catch(IOException i){
                                                          i.printStackTrace();
                                                          }
                                     }
           ;
OutSAVEFOTOS   : '>'                          ->mode (DEFAULT_MODE)
           ;
```

Ilustração 2 - Código Legenda de fotos-sFOTOS

O "mode sFOTO" apenas lê as palavras, ficheiro, quem ,quando, facto e onde e manda para os mode's correspondentes.

```
mode sFOTO;
GetFOTO      :    [ ]+'ficheiro="'   -> mode (sFICHEIRO)
             ;
GetQUEM      :    '<quem>'             -> mode (sQUEM)
             ;
GetQUANDO    :    '<quando'           -> mode (sQUANDO)
             ;
GetFACTO     :    '<facto>'            ->  mode (sFACTO)
             ;
GetONDE      :    '<onde>'            ->  mode (sONDE)
             ;
OutFOTOS     :    '</'                 ->mode (sPRINTTUDO)
             ;

DefaultsFOTO :    .                  { ; }
             ;
```

Ilustração 3 - Código Legenda de fotos - sFOTO

No "mode sQUEM" gravamos o nome da pessoa na variável "quem" e de seguida vemos se a mesma esta contida no "pessoas" (HashMap previamente criado), caso não esteja é adicionado em "pessoas". A variável "newCountKeyQuem" serve para guardar o identificador correspondente a variável "quem". Os "mode sFICHEIRO" , "mode sQUANDO", "sFACTO" e "sONDE" são análogos a "sQUEM".

```
mode sQUEM;
GetSQUEM   : ~('<')+     {quem=getText();
                          if(!(pessoas.containsKey(quem))){
                                    pessoas.put(quem,countKeyQuem);
                                    newCountKeyQuem=countKeyQuem;
                                    countKeyQuem++;
                                    whoAdded=true;
                                    }
                              else{newCountKeyQuem=pessoas.get(quem);whoAdded=false;}
                          }
        ;
```

Ilustração 4 - Código Legenda de fotos - sQUEM

Após todas as variáveis apanhadas estejam preenchidas, é no "mode sPRINTTUDO" onde se imprime todos os campos correspondentes ao RDF.
O 1º "if" é usado para que se verifique se a marcação "<quem>" é inicializada no ficheiro XML, caso esta não seja não se escreve o "bloco" de codigo no RDF.

O 2º "if" é usado para que caso este "bloco" de código já tenha sido escrito no RDF, não seja novamente escrito (assim evita-se repetição de "blocos").

```
mode sPRINTTUDO;
GetsPRINTTUDO   : 'foto'{
    if(!quem.equals("")){       //Escrever bloco equivalente aos Quem's  da foto (I'S)
      if(whoAdded){
        System.out.println("<rdf:Description rdf:about=\"&ecrm;I"+newCountKeyQuem+
            "_Description_Interview_"+countinterview+"\">");
        System.out.println("\t<rdf:type rdf:resource=\"&ecrm;E55_Type\"/>");
        System.out.println("\t<P2_has_type rdf:resource=\"&ecrm;Description\"/>");
        System.out.println("\t<P3_has_note rdf:datatype=\"&xsd;string\">"+quem+"</P3_has_note>");
        System.out.println("</rdf:Description>\n\n");
        }
       }
```

Ilustração 5 - Código Legenda de fotos - sPRINTTUDO

4-Codificação e Testes

4.1-Alternativas, Decisões e Problemas de Implementação

Uma das grandes dificuldades neste projeto foi o uso das expressões regulares , a interligação entre os vários ficheiros envolvidos e interpretação do ficheiro RDF. Tivemos que excluir algumas partes inicialmente previstas e algumas incoerências nos ficheiros XML, respetivamente a possibilidade de existir episódios dentro de episódios e as marcações no XML que significavam o mesmo, mas marcadas de forma diferente.

4.2- Testes realizados e Resultados

Para a realização dos testes utilizamos principalmente o terminal do sistema OS X.
Os passos a seguir são os seguintes:

Os 3 primeiros pontos é para se puder utilizar o ANTLR.

- export CLASSPATH=".:/usr/local/lib/antlr-4.5.3complete.jar:$CLASSPATH"
- alias antlr4='java -jar /usr/local/lib/antlr-4.5.3-complete.jar'
- alias grun='java org.antlr.v4.gui.TestRig'

Estes últimos 3 são para a execução dos testes.

- javac MainLexerBI2RDF.java
- javac BImaisLegendas.java
- java MainLexerBI2RDF ficheiroTeste.xml > ficheiroFinal.rdf

Em baixo esta apenas o ficheiro de teste da legenda das fotos, mas foram realizados testes em todos os ficheiros XML.

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- edited with XMLSPY v2004 rel. 3 U (http://www.xmlspy.com) by Museu da Pessoa (Museu da Pessoa) -->
<!DOCTYPE fotos SYSTEM "http://alfarrabio.di.uminho.pt/mpessoa/dtd/fotos.dtd">
<fotos>
    <foto ficheiro="022-F-01.jpg">
        <quem>Ana de Lourdes de Oliveira Chaminé e António Oliveira
        Machado</quem>
        <quando data="1961-01-15"/>
        <facto> Os noivos cortam o bolo de casamento</facto>
    </foto>
</fotos>
```

Ilustração 6 - ficheiroTeste.xml

```xml
<rdf:Description rdf:about="&ecrm;022-F-01.jpg">
    <rdf:type rdf:resource="&ecrm;E41_Appellation"/>
</rdf:Description>


<rdf:Description rdf:about="&ecrm;I1_Interviewed_1"/>
    <rdf:type rdf:resource="&ecrm;E38_Image"/>
    <rdf:type rdf:resource="&foaf;Image"/>
    <foaf:depicts rdf:resource="&ecrm;Interviewed_1/>
    <P67_refers_to rdf:resource="&ecrm;IAna de Lourdes de Oliveira Chaminé e António Oliveira
        Machado"/>
    <P3_has_note rdf:datatype="&xsd;string"> Os noivos cortam o bolo de casamento</P3_has_note>
    <P1_is_identified_by rdf:resource="&ecrm;022-F-01.jpg"/>
    <P4_has_time-span rdf:resource="&ecrm;TS1"/>
</rdf:Description>



<rdf:Description rdf:about="&ecrm;TS1>
    <rdf:type rdf:resource="&ecrm;E52_Time-Span"/>
<P78_is_identified_by rdf:resource="&ecrm;1961-01-15"/>
</rdf:Description>


<rdf:Description rdf:about="&ecrm;I1>
    <rdf:type rdf:resource="&ecrm;E55_Type"/>
    <P2_has_type rdf:resource="&ecrm;Description"/>
    <P3_has_note rdf:datatype="&xsd;string">Ana de Lourdes de Oliveira Chaminé e António Oliveira
        Machado</P3_has_note>
</rdf:Description>
```

Ilustração 7 - ficheiroFinal.rdf

5- Conclusão

Concluímos este relatório com uma avaliação positiva do projeto. Este projeto possibilitou-nos a experiencia de compreender e solucionar problemas que ocorrem com frequência no mundo de trabalho. Desta forma, trabalhamos com a linguagem ANTLR e JAVA para conseguir ler e manipular as marcações XML, o que nos ajudou a compreender melhor o uso de expressões e manipulação de conteúdo texto.
No futuro pretendemos a finalização deste trabalho através da correção de problemas posteriormente encontrados e a inclusão de novas funcionalidades.
Agradecemos a colaboração do Professor Pedro Rangel e da Cristiana Araújo ao longo do trabalho prático.