FCTUC FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Projeto 1

Ano Letivo 2020 / 2021

Departamento de Engenharia Informática

Integração de Sistemas
1º Semestre

Discentes

António Lopes – 2017262466 – uc2017262466@student.uc.pt

Lucas Ferreira – 2016243439 – uc2016243439@student.uc.pt

Índice

Parâmetros de avaliação..... Pág. 1

Nota Prévia..... Pág. 2

Selector..... Pág. 3-6

Processor..... Pág. 7-8

HTMLViewer..... Pág. 9-10

Conclusão..... Pág. 11

Parâmetros de avaliação

a. De um modo geral, cumprimos com todos os requisitos e conseguimos fazer todos os pressupostos pedidos no enunciado, que irão ser abordados nas páginas seguintes.

b.

Autoavaliação do grupo
90%

c.

	António Lopes	Lucas Ferreira
Selector	✓	✖
Processor	✓	✓
HTMLViewer	✖	✓
Ficheiros XML/XSLT/XSD	✓	✓
Relatório	✓	✓

d.

António Lopes	Lucas Ferreira
90%	85%

e.

António Lopes	Lucas Ferreira
11h	9h

Nota Prévia

De acordo com o primeiro trabalho prático, solicitado a realizar na cadeia de Integração de Sistemas, vamos focar a nossa atenção nos principais objetivos deste projeto que é aprender XML e como processar XML usando JAXB aliado ao XML Schema e XSLT. Neste é necessário implementar algumas técnicas para ler, validar e processar documentos XML e apresentar num ecrã em HTML após sofrer o anterior sofrer alterações feitas pelo Java e um documento XSLT.

Neste trabalho, as informações obtidas foram baseadas numa fonte fornecida para tal efeito, que é o **bookdepository.com**, onde se encontra uma seleção de livros para crianças que devem ser integrados, processados e categorizados pelas aplicações criadas com base no que foi referido no parágrafo anterior.

A primeira parte consiste numa leitura de um ficheiro XML a partir de um ficheiro Java chamado **Selector**, onde a informação contida no primeiro é processada de forma a que seja escrito um novo ficheiro XML mais pequeno com apenas as preferências do utilizador.

Já na segunda parte é reaproveitado o ficheiro produzido anteriormente para um novo XML onde a informação é totalmente reorganizada, metendo os autores apresentados no mesmo e gerando um novo ficheiro a partir de outro ficheiro Java nomeado **Processor**, modificando-o de modo a ficar com alguma informação extra, que contam com todos os livros de cada autor ordenado por rank de Best Seller. Além dessas, é criada uma outra tabela com apenas um número de autores escolhido pelo utilizador, contendo apenas os melhores livros a nível de ranking de Best Sellers e o nome do seu autor, organizados pelas tags do novo XML.

Para finalizar temos novamente mais um ficheiro Java que tem o nome de **HTMLViewer** que basicamente converte o ficheiro criado na segunda parte num HTML aliado à transformação do XSLT que simplesmente organiza a informação produzida no **Processor** em tabelas de modo a serem visíveis numa página web em HTML carregada pelo **Glassfish 5.0**.

Selector

Para o desenvolvimento da primeira parte do trabalho foi criado um ficheiro XML que possui a informação necessária de cada livro, como já foi dito no tópico anterior. Nesse ficheiro é guardado o título do livro, o seu rating de 1 a 5, o overall rating das votações das pessoas, o seu rank no que toca a Best Sellers, o seu autor, a editora que publicou, uma pequena descrição do livro, o ano e a respetiva data de lançamento.

```
<book id="0001">
    <title>Three Little Pigs</title>
    <rating>4.21</rating>
    <totalRatings>27569</totalRatings>
    <bestSellerRank>17174</bestSellerRank>
    <author>Random House Disney</author>
    <publisher>Random House USA Inc</publisher>
    <language>English</language>
    <description>The three little pigs have never looked so cute
    <year>2005</year>
    <date>11 Jan 2005</date>
</book>
```

Figura 1 – Exemplo de um livro no ficheiro xml

Com ajuda da biblioteca JAR chamada trang foi possível gerar automaticamente o ficheiro XSD do ficheiro XML, permitindo assim com a ajuda do JAXB compiler, xjc, gerar as classes necessárias para conseguirmos ler esse ficheiro. Ao compilarmos o programa, a informação do ficheiro .xml e do ficheiro .xsd é passada para as classes correspondentes.

```java
private static Catalog jaxbXmlFileToObject(File XMLfile) {
    /**
    * reads the XMLfile to Java object
    * returns rg
    */

    JAXBContext jc;
    Catalog rg = new Catalog();
    try {
        jc = JAXBContext.newInstance(Catalog.class);
        // Create unnarshaller
        Unmarshaller jaxbUnmarshaller = jc.createUnmarshaller();

        SchemaFactory schmFact = SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
        File schemaFile = new File("beforeSelector.xsd");
        Schema schemaXSD;
        try {
            schemaXSD = schmFact.newSchema(schemaFile);
            jaxbUnmarshaller.setSchema(schemaXSD);
        } catch (SAXException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }


        rg = (Catalog) jaxbUnmarshaller.unmarshal(XMLfile);
    }
    catch (JAXBException e) {
            e.printStackTrace();
        }
    return (rg);
    }
```

Figura 2 – Método executado pelo JAXB

É nesta parte que entra o **Selector**, o qual foi concebido para selecionar a informação obtida do XML. Neste foram criados vários métodos que permitem filtrar essa informação da maneira com base em opções selecionadas pelo utilizador que são mostradas em forma de menu.

```
File XMLfile = new File("beforeSelector.xml");

Catalog catalogo = jaxbXmlFileToObject(XMLfile);

Catalog novo = new Catalog();
ArrayList<String> listaAutores = getListaAutores(catalogo);

int flag = 0;

while(flag != 1){
    System.out.println("----------Selector----------");
    System.out.println("Select By:");
    System.out.println("1 - Author");
    System.out.println("2 - Year");
    System.out.println("3 - Title");
    System.out.println("4 - Rating");
    System.out.println("5 - TotalRatings");
    System.out.println("6 - BestSellerRank");
    System.out.println("7 - Author + Rating + Year");
    System.out.println("8 - Author + Year");
    System.out.println("9 - Author + Rating");
    System.out.println("10 - Rating + Year");
    System.out.println("Select option:");
    int option = 0; //scanner para ver a opçao
    option = verifyInput(0,10);
```

Figura 3 – Criação dos ficheiros objeto a partir do XML e o menu apresentado

Exemplificando, na escolha por ano é pedido ao utilizado introduzir um ano mínimo da na qual pretende obter informação, ou seja, no caso do ano 2004, por exemplo, serão selecionados somente todos os livros desse ano e escritos no novo ficheiro XML. O mesmo processo acontece nos casos da escolha por rating, total de ratings feito pelo público, e classificação de best seller.

```
public static Catalog getBooksWithYear(Catalog catalogo, int year){
    Catalog rg = new Catalog();
    BigInteger bigInteger = BigInteger.valueOf(year);
    for(int i = 0 ; i < catalogo.getBook().size(); i++){
        //System.out.println(catalogo.getBook().get(i).getYear());
       // System.out.println(catalogo.getBook().get(i).getYear().compareTo(bigInteger));
        if(catalogo.getBook().get(i).getYear().compareTo(bigInteger) != -1){
            rg.getBook().add(catalogo.getBook().get(i));
        }
    }
    return rg;
}
```

Figura 4 – Getter de todos os livros por ano, quando presentes no XML

No caso da escolha por autor é selecionado, gerado e inserido dentro de uma Arraylist da classe Catalog, que irá processar as opções do utilizador, para posteriormente ser escrito no ficheiro XML exclusivamente todos os livros desse autor. Na opção de escolha por título devolve apenas o livro escolhido.

```java
public static Catalog getBooksWithAuthorName(Catalog catalogo, String name){
    Catalog rg = new Catalog();
    for(int i = 0 ; i < catalogo.getBook().size(); i++){
        if(name.equals(catalogo.getBook().get(i).getAuthor()) == true){
            rg.getBook().add(catalogo.getBook().get(i));
        }
    }
    return rg;
}
```

Figura 5 – Getter de todos os livros de um certo autor

```java
private static void jaxbObjectToXML(Catalog rg)
{
            /**
             * converts Java Oject (rg) to XML file
             */
    try
    {
        //Create JAXB Context
        JAXBContext jaxbContext = JAXBContext.newInstance(Catalog.class);

        //Create Marshaller
        Marshaller jaxbMarshaller = jaxbContext.createMarshaller();

        //Required formatting
        jaxbMarshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE);

       //Store XML to File
        File file = new File("XMLafterSelector.xml");

        //Writes XML file to file-system
        jaxbMarshaller.marshal(rg, file);
    }
    catch (JAXBException e)
    {
        e.printStackTrace();
    }
}
```

Figura 6 – Método do JAXB para gerar o novo ficheiro XML a partir dos objetos criados

Processor

O programa Processor lê e preenche as classes respetivas da mesma maneira que o Selector, uma vez que tem o mesmo tipo de formatação e as mesmas restrições. Ao iniciar o programa é pedido ao utilizador para selecionar a quantidade de autores que pretende ter na tabela de ranking final, isto pois o ficheiro .xml não só vai ter a informação dos livros por autores como também uma tabela com a classificação dos best sellers dos autores.

```
<AuthorCatalog>
    <person>
        <personname>Random House Disney</personname>
        <book id="1">
            <title>Three Little Pigs</title>
            <rating>4.21</rating>
            <totalRatings>27569</totalRatings>
            <bestSellerRank>17174</bestSellerRank>
            <author>Random House Disney</author>
            <publisher>Random House USA Inc</publisher>
            <language>English</language>
            <description>The three little pigs have never looked so cute
            <year>2005</year>
            <date>11 Jan 2005</date>
        </book>
    </person>
```

Figura 7 – Exemplo de um autor e respetivo livro no ficheiro xml criado após o Processor

```
<author>
    <authorname>Margaret Wise Brown</authorname>
    <rank>77</rank>
</author>
<author>
    <authorname>Antoine de Saint-Exupery</authorname>
    <rank>315</rank>
</author>
<author>
    <authorname>A.A. Milne</authorname>
    <rank>442</rank>
</author>
<author>
```

Figura 8 – Exemplo da tabela com a classificação dos best sellers dos autores

De modo a obtermos a formatação desejada no novo ficheiro .xml foram criadas várias classes. A classe AuthorCatalog possui uma lista de classes Person que guarda o nome do autor dos livros e a lista dos livros que este escreveu, e ainda uma lista de Authors, que guarda também o nome do autor e a classificação deste, ordenada pela classificação do seu melhor livro.

Para preenchermos a lista de classes Person foi criada uma função que recebe a classe que contém a informação dos livros e uma lista de todos os autores e prossegue ao preenchimento da lista de Persons e a ordenação dos livros de cada autor.

```
public static AuthorCatalog createPersonClass(ArrayList<String> listaAutores, Catalog catalogoLivros){
    AuthorCatalog rg = new AuthorCatalog();
    for(int i = 0 ; i  < listaAutores.size(); i++){
        Person novo = new Person();
        novo.setPersonname(listaAutores.get(i));
        for(int j = 0; j < catalogoLivros.getBook().size();j++){
            if(catalogoLivros.getBook().get(j).getAuthor().equals(listaAutores.get(i))){
                novo.getBook().add(catalogoLivros.getBook().get(j));
            }
        }
        Collections.sort(novo.getBook(), new Comparator<Book>() {
        public int compare(Book book1, Book book2) {
            return book1.getBestSellerRank().compareTo(book2.getBestSellerRank());
        }
    });
        rg.getPerson().add(novo);
    }
    BigInteger bigInteger = BigInteger.valueOf(listaAutores.size());
    rg.setNumberofauthorsprocessed(bigInteger);
    return rg;
}
```

Figura 9 – Método para adicionar um autor e adicionar seus os livros por ordem de rank a uma lista

No fim da execução do método anterior é chamada o método que cria a lista ordenada de autores por ranking de bestseller.

```
public static AuthorCatalog createAuthorRank(AuthorCatalog catalogo, int N){
    ArrayList<String> added = new ArrayList();
    for(int i = 0 ; i < N; i++){
        int x = -1;
        BigInteger maior = BigInteger.valueOf(99999999);
        for(int j = 0; j < catalogo.getPerson().size();j++){
            //da skip dos que foram added
            if(!added.contains(catalogo.getPerson().get(j).getPersonname())){
                if(catalogo.getPerson().get(j).getBook().get(0).getBestSellerRank().compareTo(maior) == -1){ //escolhe o maior
                    maior = catalogo.getPerson().get(j).getBook().get(0).getBestSellerRank();
                    x = j;
                }
            }
        }
        //guarda o indice do rank mais baixo
        Author nova = new Author();
        nova.setAuthorname(catalogo.getPerson().get(x).getPersonname());
        nova.setRank(maior);
        catalogo.getAuthor().add(nova);
        added.add(catalogo.getPerson().get(x).getPersonname());
    }
    return catalogo;
}
```

Figura 10 – Método para adicionar um novo autor e o seu livro com melhor rank a uma lista

HTMLViewer e XSLT

Para fechar e finalizar os tópicos abordados no âmbito do projeto, é necessário agora juntar as "forças" entre o **HTMLViewer.java** e **o XSLtransformation.xsl** para converter o ficheiro criado na segunda parte com o **Processor** num HTML, com a ajuda da transformação do XSLT, que organiza toda a informação em tabelas de modo a serem visíveis numa página web.

```
<xsl:for-each select="AuthorCatalog/person">
    <div style="font-size:15pt">
        Author: <strong>
            <xsl:value-of select="personname"/>
        </strong>
    </div>
    <table border="2">
        <tr bgcolor="#9acd32" style="font-size:13pt">
            <th width="400">Title</th>
            <th width="200">Rating</th>
            <th width="200">Total ratings</th>
            <th width="200">Best Seller Rank</th>
            <th width="200">Publication Date</th>
        </tr>

        <xsl:for-each select="book">
            <tr style="font-size:11pt">
                <td><xsl:value-of select="title"/></td>
                <td><xsl:value-of select="rating"/></td>
                <td><xsl:value-of select="totalRatings"/></td>
                <td><xsl:value-of select="bestSellerRank"/></td>
                <td><xsl:value-of select="date"/></td>
            </tr>
        </xsl:for-each>
    </table>
```

Figura 11 – Transformação produzida no XSLT para organizar as tabelas de autores

```java
public class HTMLViewer {

    public static void main(String[] args) {
        try {
            TransformerFactory tFactory = TransformerFactory.newInstance();
            Source xslFile = new StreamSource( systemId: "../projeto2/XSLtransformation.xsl");
            Source xmlFile = new StreamSource( systemId: "../projeto2/XMLafterProcessor.xml");
            OutputStream htmlFile = new FileOutputStream( name: "outputHTML.html");
            Transformer transform = tFactory.newTransformer(xslFile);
            transform.transform(xmlFile, new StreamResult(htmlFile));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Figura 12 – Método da geração do ficheiro HTML da classe HTMLViewer

Após todos estes processos, finalmente é produzido o ficheiro HTML que posteriormente é carregado pelo servidor de aplicação Glassfish e mostrado num browser à escolha do utilizador.

Author: **Margaret Wise Brown**

Title	Rating	Total ratings	Bestseller Rank	Publication Date
Goodnight Moon	4.28	294461	77	24 Jun 2011

Author: **E B White**

Title	Rating	Total ratings	Bestseller Rank	Publication Date
Charlotte's Web	4.17	1346447	859	10 Apr 2012
Stuart Little	3.89	104158	7398	01 Sep 2020
The Trumpet of the Swan	4.08	67370	16107	02 Sep 2020

Figura 13 – Exemplo da tabela de autores selecionados e respetivos livros por Best Seller rank

Author by Bestseller Rank

Author	Bestseller Rank
Margaret Wise Brown	77
Antoine de Saint-Exupery	315
A.A. Milne	442
Roald Dahl	668
E B White	859

Figura 14 – Exemplo da tabela com o número de autores escolhidos nos processor e respetivos livros que têm o melhor rank de Best Seller

Conclusão

De forma a concluir este trabalho prático, conseguimos tirar algumas ilações quanto às implementações e importância de diferentes aspetos que perfazem este projeto, e neste caso, estas são bastante positivas, dado crermos que alcançámos todos os pressupostos pretendidos, juntando à parte da aprendizagem que também consideramos como feito. Em tom de encerramento, a nossa participação e empenho foram bastante positivos.

É de salientar ainda que, como grupo, dedicamos algumas horas extra para conseguir aglomerar os conhecimentos necessários para a realização deste trabalho, dado que sendo alunos de terceira fase, perdemos algumas aulas chave que facilitaria este processo.