University of Coimbra

Department of Informatics Engineering

Systems Integration

Report for Project 1

Author:
António Lima
2011166926

Author:
Pedro Janeiro
2012143629

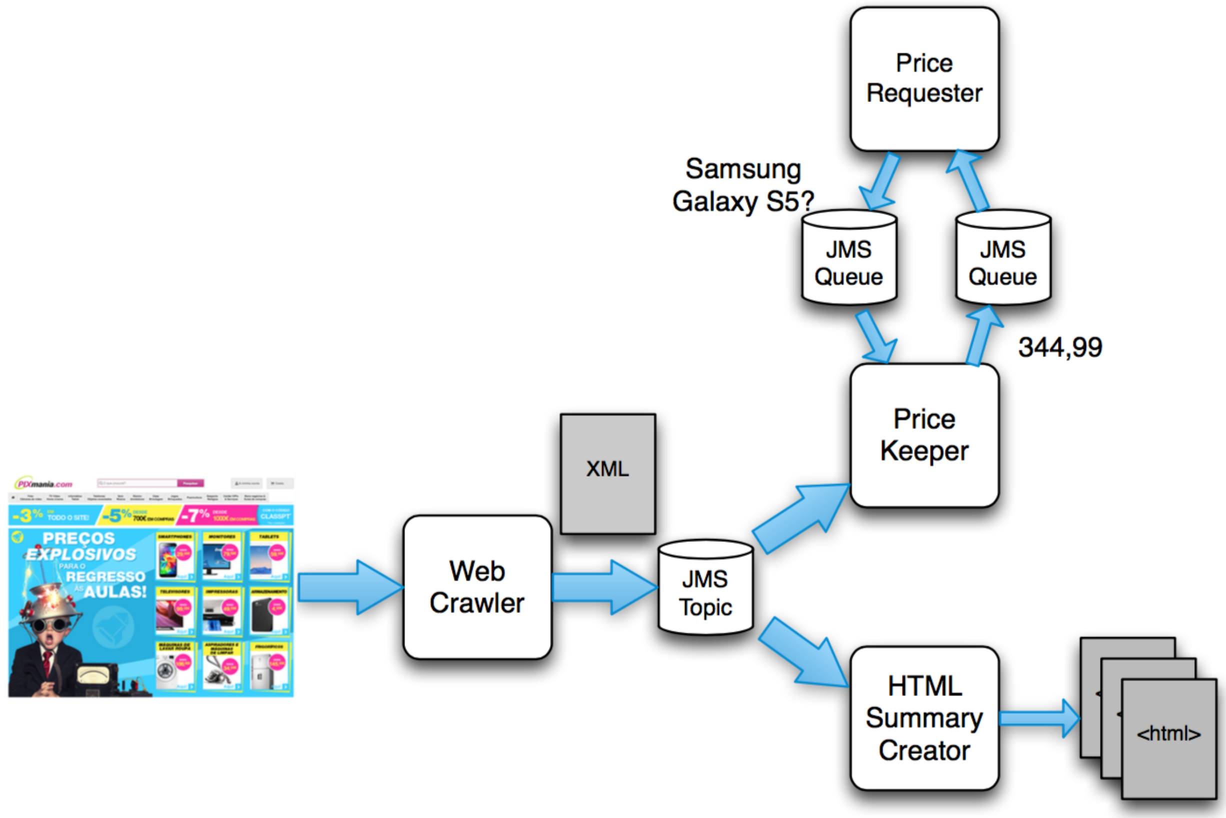
October 16, 2015

Contents

1 Introduction

This assignment is comprised of four applications. The first one is a **Web Crawler** that collects information of smartphones from the Pixmania web site, extracts the relevant data to **XML**, and sends it to a **Java Message Service Topic** (JMS Topic). This topic serves two other applications that process the data, a Price Keeper and an HTML Summary Generator. The **Price Keeper** keeps the prices of smartphones in memory. The **Price Requester** asks prices of smartphones to the Price Keeper. The remaining application, **HTML Summary Creator**, summarizes the information and creates HTML files for later visualization.

The following figure illustrates the previously described scenario.



The four applications are described in greater detail in the following sections.

2 Web Crawler

The Web Crawler is a stand-alone command-line application that reads a web page and sends an XML message to a JMS Topic, carrying details of smartphones from the Pixmania site.

2.1 Execution

It can be run using the following command

```
java WebCrawler <url> [<search_regexes.json>] [<backupHMTLFiles>]
```

where <**url**> is the starting *URL*, <**search_regexes.json**> is a JSON file containing an array of string *regexes* for what are to be considered relevant websites, and <**backupHMTLFiles**> is a boolean (*true* or *false*) for whether or not the crawler should save to disk all the HTML Files that are deemed relevant.

search_regexes.json should look something like this:

```
[
  "http://www\\.pixmania\\.pt/",

  "http://www\\.pixmania\\.pt/smartphone/([a-z]|[A-Z]|-|_|\\d)+/(\\d+)-a\\.html",
  "http://www\\.pixmania\\.pt/iphone/([a-z]|[A-Z]|-|_|\\d)+/(\\d+)-a\\.html",

  "http://www\\.pixmania\\.pt/telefones/telemovel/smartphone/([a-z]|[A-Z]|-|_|\\d)+\\.html",
  "http://www\\.pixmania\\.pt/telefones/telemovel/iphone/([a-z]|[A-Z]|-|_|\\d)+\\.html",

  "http://www\\.pixmania\\.pt/telefones/telemovel/smartphone/([a-z]|[A-Z]|-|_|\\d)+/(\\d+)-a\\.html",
  "http://www\\.pixmania\\.pt/telefones/telemovel/iphone/([a-z]|[A-Z]|-|_|\\d)+/(\\d+)-a\\.html",

  "(http://www\\.pixmania\\.pt/)((telefones/telemovel/)?(smartphone|iphone)/([a-z]|[A-Z]|-|_|\\d)+/(\\d+)-a\\.html)?"
]
```

This way of providing a starting *URL* and file with *regexes* for evaluating relevance is our solution to generalizing the Web Crawler for other websites.

2.2 Unpublished Messages

The Web Crawler starts off by searching for **unpublished messages** to the ***JMS Topic***, left behind on a temporary file from a previous crawling session. If any are found, then it will attempt to **resend** them. If this fails the unpublished messages will be added to the current session's list of collected smartphones.

2.3 Visiting URLs

After this initial verification the crawler will start crawling from the provided *starting URL*. In order to avoid revisiting pages, the crawler keeps a ***HashMap* of visited URLs**.

If the command line argument <**backupHMTLFiles**> was set to true then each URL will be saved to disk under an appropriate directory using the *URI* naming convention.

2.4 Jsoup HTML Parsing

Since websites vary their HTML layouts vastly, instead of using complex regexes to parse each web page we opt to use a simple, more generic, query language (Jsoup).

Using ***Jsoup*** we query each web page for relevant information such as the smartphone's name, brand, price, price's currency, summary description and technical data which we then use to populate a ***Smartphone*** Java object class. Afterwards we gather the **list of URLs** in the page and, if they are relevant, **recursively** visit them.

The **Smartphone** object is generated from the **XSD**, which is to be used for **XML validation** (described later on), in order to guarantee information consistency.

2.5 XML File Creation

After the crawler stops visiting web pages we create the corresponding **XML file** for each of the collected **Smartphone objects** using **JAXB**'s ***marshaller***. Marshalling, by definition, consists of transforming the memory representation of an object to a data format suitable for storage or transmission, in our case, an XML file.

2.6 Publishing Messages

Now that the data has been marshalled we can send it to the **JMS Topic**. After initializing the JMS Topic the crawler will attempt to publish the XML files however, should this fail 5 times, it will save the unpublished files for the next crawling session.

3 HTML Summary Creator

The **HTML Summary Creator** application runs indefinitely, listening for **XML messages** from the **JMS topic** and creates the corresponding **HTML files** using an **XSL template**. The name of the **HTML files** is given according to the **MD5** hash of the corresponding XML file. In addition to all of this, the HTML Summary Creator is also responsible for performing the validation of the **XML files** using a **XSD** file.

3.1 Execution

It can be run using:

```
java HTMLGenerator <xsd> <xsl> <html>
```

Where <**xsd**> is the file with the schema used for validation, <**xsl**> is the file with the stylesheet for generating the HTML files, and<**html**> is the directory where the HTML files should be stored.

4 Price Keeper

As the Web Crawler sends smartphones the **Price Keeper** application keeps track of them.

It uses two different communication channels, a **durable subscription topic** where the Crawler sends its messages and a **queue** where the Price Requester clients can ask the Price Keeper for information.

Last but not least, it is also responsible for performing the validation of the **XML files** using a **XSD** file.

4.1 Execution

It can be run using:

```
java PriceKeeper <xsd>
```

Where <**xsd**> is the file with the schema used for validation.

5 Price Requester

A **Price Requester** can send queries to the Price Keeper for smartphones that satisfy certain criteria, namely the smartphone's **name, price range, brand and a name-brand pair**.

The Price Requester creates a **temporary queue** for the Price Keeper to answer, which easily allows **several requester applications** to run at the same time and minimizes resource usage.

5.1 Execution

It can be run using:

```
java PriceRequester
```

6 JBOSS Configuration

In order to run the different applications, we need to set up the JBOSS Server.

6.1 Creating user

```
% $JBOSS_HOME/bin/add-user.sh
(Type of user): b
(Username): pjaneiro
(Password): |Sisc00l
(Groups): guest
(AS process): no
```

6.2 Creating topic

```
% $JBOSS_HOME/bin/jboss-cli.sh
connect
jms-topic add --topic-address=pixmania --entries=java:jboss/exported/jms/topic/pixmania
```

6.3 Creating queue

```
% $JBOSS_HOME/bin/jboss-cli.sh
connect
jms-queue add --queue-address=queue --entries=java:jboss/exported/jms/queue/queue
```

6.4 Setting client-ids

```
% $JBOSS_HOME/bin/jboss-cli.sh
connect
connection-factory --name=RemoteConnectionFactory --client-id=HTMLGenerator
connection-factory --name=RemoteConnectionFactory --client-id=PriceKeeper
```

6.5 Giving the user permission to change the topic

There are two ways to execute this one: the easy way, and the hard way.

For the easy way, one must create a super-user (running the commands described in the section "Creating user", but using 'a' as the first option). Then, one must access the browser interface of the wildfly server (localhost:9990), login with the super-user created, and run the following sequence of links: *Configuration* → *Subsystems* → *Messaging* → *default* → *destinations* → *SecuritySettings* → (*select the desired role*) → *Advanced* → *Edit*. Then, all that's left to do is to mark the "CreateDurable?" and "DeleteDurable?" checkboxes and press the Save button.

The hard way means editing the file *$JBOSS_HOME/standalone/configuration/standalone-full.xml*, finding the section "security-settings" and adding the lines:

```
<permission type="createDurableQueue" roles="guest"/>
<permission type="deleteDurableQueue" roles="guest"/>
```

6.6 Running the server

```
% $JBOSS_HOME/bin/standalone.sh -c standalone-full.xml
```