Departamento de Engenharia Informática

Faculdade de Ciência e Tecnologia
Universidade de Coimbra

XML and XML Manipulation, Java Message Service and Message Oriented Middleware

Enterprise Application Integration

Coimbra, 16th of October 2015

Flávio J. Saraiva, nº 2006128475, flavioj@student.dei.uc.pt
Mário A. Pereira, nº 1998018322, mgreis@student.dei.uc.pt

Index:

Objectives

The main goal of this project was to build a distributed system constituted by several applications that communicate asynchronously amongst themselves using simple XML messages.

This system should allow for information recovered by a web crawler from a website using “Screen Scraping” techniques to be made available for querying by several clients. In order to satisfy the clients there should be a third application that would receive XML messages from the crawler and execute the queries of the clients returning the result. A fourth application would be necessary to store the data in an HTML file.

In order to allow asynchronous communication between the components of the system, message oriented middleware was also be deployed to act as a message broker between the components running in a WildFly 9.0.1. server. This allowed for the system components to communicate in several different ways using a publish subscribe topic between the Web Crawler, the Price Keeper and the HTML Summary Creator; a message queue that allowed for the Price Keeper to receive the requests from the Price Requester; and several temporary message queues that allowed for the Price Keeper to satisfy the Price Requester requests.

In order to assure successful communication, a communication scheme had to be enforced so the many components of the system would be able to work together. To achieve this goal a XSD scheme was shared by applications and used to ensure that the messages sent and received by each component were well formed. This scheme also allowed us to generate Java classes that could represent our schema and allowed the Marshalling and UnMarshalling of information from XML into java Objects and Vice Versa.

Finally the HTML Summary Creator wa able to output XML data into HTML table using a XSL file. This HTML table was also stored into a file for future reference.

Implementation

For this assignment four applications were created. An additional project called Common containing properties, configuration, XSD, and additional Java Classes was also created.

Finally, to allow communication between the components of the system, it was necessary to use the Java Message Service 2.0. with the WildFly 9.0.1. Server.

Common Project

This project contains all necessary configuration files and the classes common to the four applications.

By combining this files and importing them to each individual project we were able to make sure the common objects and de XSD schema remained the same throughout the system.

This project also contains the necessary files an Object Factory necessary to marshall and unmarshal the objects into XML data dan vice versa.

The XSD schema can be consulted in annex B.

A XML sample file can be consulted in annex A.

The Web Crawler

The Web Crawler is an application that receives the HTML from a set of URLs contained in the webAddresses.txt file, does the necessary operations to remove the relevant information from the HTML, constructs a well formed XML String, and publish that String in a topic.

Having so many different actions to performed the application was divided into several classes:

WebCrawler.java

This class has the main method of the application and contains a scheduler that is able to launch the WebCrawler search engine in regular time intervals.

In order to achieve such goal a Timer Class instance was used. Using the ScheduleAtFixedRate () method we were able to launch the engine thread at regular time intervals allowing for the application to function indefinitely and publish new messages containing new information regularly.

Engine.java

This class contains the Web Crawler itself and extends the Thread class.

When the scheduler launches this thread, first of all, it retrieves the Web addresses from the webAdresses.txt file, then it begins crawling the web pages one by one using the crawl() method. This method uses the Jsoup API in order to extract and manipulate data into a Document class. This contains a DOM and allows us to make queries in order o retrieve the necessary information to populate a Smartphone class object.

Using a list of smartphone objects and the XmlHelper class contained in the common project we were able to marshall the object information into a String containing the XML.

Finally the Engine launches a Sender thread in order to publish the XML into our topic

Sender.java

This class contains the necessary methods in order to publish the XML into the topic. It implements the Runnable interface so it can be launched as a thread.

Before any information can be sent it must be parsed and validated against our XSD schema. If the XML document is well formed it will be published in the topic.

In order for the Web Crawler to deal with transitory failures of the message broker, several issues had to be delt with. In case the thread was not able to publish in the topic a retry mechanism was implemented in the send() method. If this mechanism failed 10 times in a row all data would be written into a file.This data cand be latter retrieved and sent the next time the sender thread executes.

HTML Summary Creator

This application of our system is very simple. It receives an XML string from the topic and uses the XMLHelper.validate() method contained in the common part of the project to validate if the received XML String is well formed.

If the XML is well formed it proceeds to transformed the received message into an HTML table using the to_html.xsl schema present in the package.

Finally it stores the HTML in a local file.

One of the most important components of the application is the to_html.xsl file present in the package. This file allows for the TransformFactory class to get the proper xsl schema in order to retrieve and organize the information contained in the XML String so that it can be transformed into an HTML table and more easily read by the user. The Contents of the file are displayed in Annex C.

Price Keeper

The Price Keeper shares similarities with the HTML Summary Creator. IT also receives an XML message from the subscribed topic and validates it in a similar way.

This application has a second interface to deal with Price Requesters requests. Upon receiving a request it uses the search.xsl file present in the package to allow the transformFactory Class from the common project to retrieve and organize the information obtained into an XML String that can be sent has a resonse to the user request.

The search.xsl file present in the package. This file allows for the TransformFactory class to get the proper xsl schema in order to filter, retrieve and organize the information contained in the XML String so that it can sent as an answer to the client request. The Contents of the file are displayed in Annex E.

Price Requester

The final application of our system is very simple. It receives a String from the console, puts it in the messageQueue, and waits for a response to its request. When this response arrives it uses the XMLHelper.validate() method contained in the common part of the project to validate if the received XML String is well formed.

If the XML is well formed it proceeds to transformed the received message into plain text using the to_text.xsl schema present in the package.

Finally it prints the information in the console. And asks the user for a new String to search for.

One of the most important components of the application is the to_text.xsl file present in the package. This file allows for the TransformFactory class to get the proper xsl schema in order to retrieve and organize the information contained in the XML String so that it can be more easily read by the user. The Contents of the file are displayed in Annex D.

Java Message service 2.0 with WildFly 9.0.1 Server

One of the most important components of our system is the message broker. In order for our applications to communicate amonst themselves JMS 2.0. API was used. This API is a Java Message Oriented Middleware that allowed us to implement loosely coupled, asynchronous communication between the components of our system.

In order to achieve such an objective the Wildfly server was used as a message broker. This allowed us to implement a Queue and a Topic that allowed for our applications to communicate.

Messaging Queue:

java:jboss/exported/IS/Project1/PriceKeeperQueue

This queue allowed for the Price Requesters to communicate with the price keeper in a loosely coupled and asynchronous way. Any message posted in the queue by the requesters will be kept and eventually delivered to keeper when it contacts the message broker server.

Topic:

java:jboss/exported/IS/Project1/WebCrawlerTopic

This topic allowed for the Web Crawler to post messages that would be kept and eventually delivered to all the applications that subscribed this topic. This allows for many to many loosely coupled, asynchronous communication since messages published will eventually be delivered to all the subscribed consumers without the publisher necessarily knowing the subscribers. Messages published before an application subscribed to the topic will not be delivered.

Temporary Messaging Queue:

Beside this permanent messaging structures, in order for the Price Keeper to answer the Price Requester requests, a temporary queue was created with each request from a certain Price Requester. The keeper would receive a request via it’s permanent queue and answer to it via the temporary queue that is only valid for this request. This allows for the keeper to answer individually to each requester .

Discussion of Results

During the implementation of our system the challenges posed by the communication between components along a network with unknown topology became apparent.

The need to ensure that comunications would not be hindered by transitory fails of the system components and network was one of the main challenges of the project.

By using Message Oriented Middleware acting as a message broker the problems related with this failures became easier to solve. Not only we were able to achieve communication asynchronously, we were also able to make sure all messages sent by a component would eventually be delivered to its destiny.

Message Oriented Middleware also allowed us to improve communication by used filtering techniques like the topic bases system in which the sender was able to determine who the recipients of the message would be without necessarily knowing them using a Publish-Subscribe messaging pattern.

Another challenge imposed by the project resided in the need for the need for the recipient of the message to be able to decode and using it. By enforcing a XSD schema we were able to make sure all the XML messages traded between components of our system were well formed and could be decoded and used by its recipient. We were also able to, using an XML binding compiler, to generate Java classes form an XML schema. This allowed us to Marshal and Unmarshal information from XML messages into Java Objects and vice versa, which made implementation a lot easier since the messages were readily transformed into objects that could be manipulated and later turned into back into XML messages to be sent to other components of the system.

Finally, using XSL files, we were able to output HTML tables and plain text from XML data. This allowed us to easily transform this data into more human friendly formats.

Conclusions

With the project we were able to develop all the components of the proposed system that allowed us to retrieve information from a web site and make it available to be queried by remote clients.

It became apparent the importance of Message Oriented Middleware as a communication broker between loosely coupled remote applications and the necessity of using XSD enforced XML data in the messages sent between the components for the system to work properly.

All the objectives of the project were met.

Annexes

Annex A: Sample.XML file

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
This is a sample file produced according to schema.xsd.
that will be produced from the following url:
http://www.pixmania.pt/telefones/telemovel/smartphone/xx-xx-xx-xx-topsellers-1-100-page-19883-s.html
-->
<report version="1.0" timestamp="1444595841" crawler="sample.xml">
   <smartphone>
      <title>SAMSUNG Galaxy S5 - preto - Smartphone</title>
      <description name="processador">Quad-Core 2,5 GHz memória RAM: 2 GB</description>
      <description name="Tecnologia do ecrã">Capacitivo - Super AMOLED</description>
      <description name="Tamanho do ecrã">5,1" (1920 x 1080 pixeis)</description>
         <description name="Resolução máxima (em pixeis)">Máquina fotográfica (atrás): 16 megapixeis Câmara frontal: 2
megapixeis</description>
      <price currency="EUR">344.99999</price>
      <url>http://www.pixmania.pt/smartphone/samsung-galaxy-s5-preto-smartphone/22082447-a.html</url>
   </smartphone>
   <smartphone>
      <title>LG G3 - Titânio - 16 GB - 4G - Smartphone</title>
      <description name="processador">Snapdragon 801 AC Quad Core 2,5 GHz</description>
      <description name="Tecnologia do ecrã">Tátil capacitivo</description>
      <description name="Tamanho do ecrã">5,5" (2560 x 1440 pixeis)</description>
      <description name="Resolução máxima (em pixeis)">13 MP</description>
      <price currency="EUR">308.99999</price>
      <url>http://www.pixmania.pt/smartphone/lg-g3-titanio-16-gb-4g-smartphone/22181506-a.html</url>
   </smartphone>
</report>
```

Annex B: schema.XSD file

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

   <!-- WebCrawler Report: -->
   <xs:element name="report">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="smartphone" type="Smartphone" minOccurs="0" maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attribute name="version" use="required">
            <xs:simpleType>
               <xs:restriction base="xs:decimal">
                  <!-- we only know about version 1.0 -->
                  <xs:enumeration value="1.0" />
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         <xs:attribute name="timestamp" type="xs:nonNegativeInteger" use="required"/>
         <xs:attribute name="crawler" type="xs:string" use="required"/>
      </xs:complexType>
   </xs:element>
```

```
<!-- Custom Types: -->
<xs:complexType name="Smartphone">
   <xs:sequence>
      <xs:element name="title" type="xs:string"/>
      <xs:element name="description" type="Description" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="price" type="Price"/>
      <xs:element name="url" type="xs:anyURI"/>
   </xs:sequence>
</xs:complexType>

<xs:complexType name="Description">
   <xs:simpleContent>
      <xs:extension base="xs:string">
         <xs:attribute name="name" use="required"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>

<xs:complexType name="Price">
   <xs:simpleContent>
      <xs:extension base="xs:decimal">
         <xs:attribute name="currency" default="EUR">
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <!-- ISO 4217 country codes -->
                  <!--
                  <xs:minLength value="3"/>
                  <xs:maxLength value="3"/>
                  <xs:pattern value="[A-Z]{3}"/>
                  -->
                  <!-- https://en.wikipedia.org/wiki/ISO_4217 -->
                  <xs:enumeration value="AED"/>
                  <xs:enumeration value="AFN"/>
                  <xs:enumeration value="ALL"/>
                  <xs:enumeration value="AMD"/>
                  <xs:enumeration value="ANG"/>
                  <xs:enumeration value="AOA"/>
                  <xs:enumeration value="ARS"/>
                  <xs:enumeration value="AUD"/>
                  <xs:enumeration value="AWG"/>
                  <xs:enumeration value="AZN"/>
                  <xs:enumeration value="BAM"/>
                  <xs:enumeration value="BBD"/>
                  <xs:enumeration value="BDT"/>
                  <xs:enumeration value="BGN"/>
                  <xs:enumeration value="BHD"/>
                  <xs:enumeration value="BIF"/>
                  <xs:enumeration value="BMD"/>
                  <xs:enumeration value="BND"/>
                  <xs:enumeration value="BOB"/>
                  <xs:enumeration value="BOV"/>
                  <xs:enumeration value="BRL"/>
                  <xs:enumeration value="BSD"/>
                  <xs:enumeration value="BTN"/>
                  <xs:enumeration value="BWP"/>
                  <xs:enumeration value="BYR"/>
                  <xs:enumeration value="BZD"/>
                  <xs:enumeration value="CAD"/>
                  <xs:enumeration value="CDF"/>
                  <xs:enumeration value="CHE"/>
                  <xs:enumeration value="CHF"/>
                  <xs:enumeration value="CHW"/>
```

<xs:enumeration value="CLF"/>
<xs:enumeration value="CLP"/>
<xs:enumeration value="CNY"/>
<xs:enumeration value="COP"/>
<xs:enumeration value="COU"/>
<xs:enumeration value="CRC"/>
<xs:enumeration value="CUC"/>
<xs:enumeration value="CUP"/>
<xs:enumeration value="CVE"/>
<xs:enumeration value="CZK"/>
<xs:enumeration value="DJF"/>
<xs:enumeration value="DKK"/>
<xs:enumeration value="DOP"/>
<xs:enumeration value="DZD"/>
<xs:enumeration value="EGP"/>
<xs:enumeration value="ERN"/>
<xs:enumeration value="ETB"/>
<xs:enumeration value="EUR"/>
<xs:enumeration value="FJD"/>
<xs:enumeration value="FKP"/>
<xs:enumeration value="GBP"/>
<xs:enumeration value="GEL"/>
<xs:enumeration value="GHS"/>
<xs:enumeration value="GIP"/>
<xs:enumeration value="GMD"/>
<xs:enumeration value="GNF"/>
<xs:enumeration value="GTQ"/>
<xs:enumeration value="GYD"/>
<xs:enumeration value="HKD"/>
<xs:enumeration value="HNL"/>
<xs:enumeration value="HRK"/>
<xs:enumeration value="HTG"/>
<xs:enumeration value="HUF"/>
<xs:enumeration value="IDR"/>
<xs:enumeration value="ILS"/>
<xs:enumeration value="INR"/>
<xs:enumeration value="IQD"/>
<xs:enumeration value="IRR"/>
<xs:enumeration value="ISK"/>
<xs:enumeration value="JMD"/>
<xs:enumeration value="JOD"/>
<xs:enumeration value="JPY"/>
<xs:enumeration value="KES"/>
<xs:enumeration value="KGS"/>
<xs:enumeration value="KHR"/>
<xs:enumeration value="KMF"/>
<xs:enumeration value="KPW"/>
<xs:enumeration value="KRW"/>
<xs:enumeration value="KWD"/>
<xs:enumeration value="KYD"/>
<xs:enumeration value="KZT"/>
<xs:enumeration value="LAK"/>
<xs:enumeration value="LBP"/>
<xs:enumeration value="LKR"/>
<xs:enumeration value="LRD"/>
<xs:enumeration value="LSL"/>
<xs:enumeration value="LYD"/>
<xs:enumeration value="MAD"/>
<xs:enumeration value="MDL"/>
<xs:enumeration value="MGA"/>
<xs:enumeration value="MKD"/>
<xs:enumeration value="MMK"/>

<xs:enumeration value="MNT"/>
<xs:enumeration value="MOP"/>
<xs:enumeration value="MRO"/>
<xs:enumeration value="MUR"/>
<xs:enumeration value="MVR"/>
<xs:enumeration value="MWK"/>
<xs:enumeration value="MXN"/>
<xs:enumeration value="MXV"/>
<xs:enumeration value="MYR"/>
<xs:enumeration value="MZN"/>
<xs:enumeration value="NAD"/>
<xs:enumeration value="NGN"/>
<xs:enumeration value="NIO"/>
<xs:enumeration value="NOK"/>
<xs:enumeration value="NPR"/>
<xs:enumeration value="NZD"/>
<xs:enumeration value="OMR"/>
<xs:enumeration value="PAB"/>
<xs:enumeration value="PEN"/>
<xs:enumeration value="PGK"/>
<xs:enumeration value="PHP"/>
<xs:enumeration value="PKR"/>
<xs:enumeration value="PLN"/>
<xs:enumeration value="PYG"/>
<xs:enumeration value="QAR"/>
<xs:enumeration value="RON"/>
<xs:enumeration value="RSD"/>
<xs:enumeration value="RUB"/>
<xs:enumeration value="RWF"/>
<xs:enumeration value="SAR"/>
<xs:enumeration value="SBD"/>
<xs:enumeration value="SCR"/>
<xs:enumeration value="SDG"/>
<xs:enumeration value="SEK"/>
<xs:enumeration value="SGD"/>
<xs:enumeration value="SHP"/>
<xs:enumeration value="SLL"/>
<xs:enumeration value="SOS"/>
<xs:enumeration value="SRD"/>
<xs:enumeration value="SSP"/>
<xs:enumeration value="STD"/>
<xs:enumeration value="SYP"/>
<xs:enumeration value="SZL"/>
<xs:enumeration value="THB"/>
<xs:enumeration value="TJS"/>
<xs:enumeration value="TMT"/>
<xs:enumeration value="TND"/>
<xs:enumeration value="TOP"/>
<xs:enumeration value="TRY"/>
<xs:enumeration value="TTD"/>
<xs:enumeration value="TWD"/>
<xs:enumeration value="TZS"/>
<xs:enumeration value="UAH"/>
<xs:enumeration value="UGX"/>
<xs:enumeration value="USD"/>
<xs:enumeration value="USN"/>
<xs:enumeration value="USS"/>
<xs:enumeration value="UYI"/>
<xs:enumeration value="UYU"/>
<xs:enumeration value="UZS"/>
<xs:enumeration value="VEF"/>
<xs:enumeration value="VND"/>

```
                    <xs:enumeration value="VUV"/>
                    <xs:enumeration value="WST"/>
                    <xs:enumeration value="XAF"/>
                    <xs:enumeration value="XAG"/>
                    <xs:enumeration value="XAU"/>
                    <xs:enumeration value="XBA"/>
                    <xs:enumeration value="XBB"/>
                    <xs:enumeration value="XBC"/>
                    <xs:enumeration value="XBD"/>
                    <xs:enumeration value="XCD"/>
                    <xs:enumeration value="XDR"/>
                    <xs:enumeration value="XFU"/>
                    <xs:enumeration value="XOF"/>
                    <xs:enumeration value="XPD"/>
                    <xs:enumeration value="XPF"/>
                    <xs:enumeration value="XPT"/>
                    <xs:enumeration value="XSU"/>
                    <xs:enumeration value="XTS"/>
                    <xs:enumeration value="XUA"/>
                    <xs:enumeration value="XXX"/>
                    <xs:enumeration value="YER"/>
                    <xs:enumeration value="ZAR"/>
                    <xs:enumeration value="ZMW"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

</xs:schema>
```

Annex C: to_html.xsl file

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
    Document   : transform.xsl
    Created on : October 4, 2015, 3:56 PM
    Author     : Flávio J. Saraiva
    Description:
        Create a summary of the xml data.
-->

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

    <xsl:output method="html"/>

    <!-- Html page -->
    <xsl:template match="/">
        <html>
            <head>
                <title>Report</title>
                <style>
                    /* table border */
                    table {border-collapse: collapse;}
                    table, th, td {border: 1px solid black;}

                    /* table header background */
                    th {background-color: #9acd32;}
```

```
            /* list 'headers' */
            dt {font-weight: bold;}
        </style>
    </head>
    <body>
        <xsl:apply-templates select="report" />
    </body>
  </html>
</xsl:template>

<!-- Put data in a table -->
<xsl:template match="report">
  <dl>
    <dt>Version</dt>
    <dd>
        <xsl:value-of select="@version"/>
    </dd>
    <dt>Timestamp</dt>
    <dd>
        <xsl:value-of select="@timestamp"/>
        <!-- @todo transform timestamp -->
    </dd>
    <dt>Crawler</dt>
    <dd>
        <xsl:value-of select="@crawler"/>
    </dd>
  </dl>
  <table>
    <thead>
        <tr>
            <th style="text-align:left">Title</th>
            <th style="text-align:left">Description</th>
            <th style="text-align:left">Price</th>
            <th style="text-align:left">URL</th>
        </tr>
    </thead>
    <tbody>
        <xsl:apply-templates select="smartphone" />
    </tbody>
  </table>
</xsl:template>

<!-- Smartphone table row -->
<xsl:template match="smartphone">
  <tr>
    <td>
        <xsl:value-of select="title"/>
    </td>
    <td>
        <dl>
            <xsl:for-each select="description">
                <dt>
                    <xsl:value-of select="@name"/>
                </dt>
                <dd>
                    <xsl:value-of select="."/>
                </dd>
            </xsl:for-each>
        </dl>
    </td>
    <td>
```

```
            <xsl:value-of select="price"/>
            <xsl:text> </xsl:text>
            <xsl:value-of select="price/@currency"/>
         </td>
         <td>
            <a href="{url}">link</a>
         </td>
      </tr>
   </xsl:template>

</xsl:stylesheet>
```

Annex D: to_text.xsl file

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output omit-xml-declaration="yes" indent="yes"/>
   <xsl:output method="text"/>
   <xsl:strip-space elements="*"/>


   <xsl:template match="report/smartphone">
                                                                                                                        <xsl:value-of
select="concat('----------------------------------------------------------------------------------------------------------------------------','&#xA;')"/>
      <xsl:value-of select="concat( title , ' : ' , price , ' ' , price/@currency ,'&#xA;' , url ,'&#xA;','&#xA;')"/>
      <xsl:for-each select="description">
         <dt>
            <xsl:value-of select="@name"/>
         </dt>
         <dt>
            <xsl:value-of select="concat(' ',.,'&#xA;')"/>
         </dt>


      </xsl:for-each>
      <dt>
                                                                                                                        <xsl:value-of
select="concat('------------------------------------------------------------------------------------------------------------------------','&#xA;')"/>

      </dt>
   </xsl:template>
</xsl:stylesheet>
```

Annex E: search.xsl file

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
   Document   : seach.xsl
   Created on : October 4, 2015, 3:56 PM
   Author     : Flávio J. Saraiva
   Description:
      Search for smarphones with a specific content by replacing %XPATH%.
-->

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

   <xsl:output method="xml" indent="yes"/>
```

```
    <!-- root -->
    <xsl:template match="/">
        <xsl:apply-templates select="report"/>
    </xsl:template>

    <!-- copy report -->
    <xsl:template match="report">
        <xsl:copy>
            <xsl:copy-of select="@*"/>
            <xsl:apply-templates select="smartphone"/>
        </xsl:copy>
    </xsl:template>

    <!-- copy smartphones that match -->
    <xsl:template match="smartphone">
        <xsl:choose>
            <xsl:when test="%XPATH%">
                <xsl:copy-of select="."/>
            </xsl:when>
        </xsl:choose>
    </xsl:template>

</xsl:stylesheet>
```