KTH V17P01 Programming Web Services: Homework 1

Fannar Magnusson (fannar@kth.se)
Thorsteinn Thorri Sigurdsson (ttsi@kth.se)

January 31, 2017

1 Introduction

This assignment revolves around XML processing. We want to create an XML processing application for a recruitment service where we consume different XML files from various locations and compile an applicant profile XML to use in our system.

There are four input XMLs:

1. `CompanyInfo.xml` - Contains information on registered companies and their contact information. Provided by company database.
2. `EmploymentRecord.xml` - Contains information on where the applicant has worked in the past, what role he had there and what time period he was employed. Provided by employment office.
3. `Resume.xml` - Represents the information that the applicant provides to our system. Contact information, references and other information. Provided by applicant.
4. `Transcript.xml` - Contains information about the applicant's education. What school he studied at, what degrees he was enrolled in, and the courses he took in each degree. Provided by the university.

We combine information from these four XML documents into an `ApplicantProfile.xml` that the recruitment service can use in its system.

2 Implementation

2.1 Task 1

We created XSD schemas for each of the XML documents, both the four input XMLs and the output `ApplicantProfile.xml`. Then we generated (manually) XML documents that conform to the rules defined in the schemas.

The XML schemas include complex and simple types, attributes and elements, and restrictions.

We validated the generated XML documents with the schemas using an online tool [1].

The `ApplicantProfile.xsd` schema contains a `targetNamespace` declaration pointing to `http://www.bestrecruitment.com` which indicates that the elements defined by the schema come from that namespace. The generated `ApplicantProfile.xml` has to define its elements with that namespace.

2.2 Task 2

In this task we process the input XML using each of the following methods at least once:

1. DOM - Document Object Model

   We use DOM to parse the `CompanyInfo.xml`. We start by creating a `FileInputStream` and then use a `DocumentBuilder` to get elements and attributes by name from the file and store them as local values. After we have got all the values we wanted, we return a new Java object consisting of all the elements.

2. SAX - Simple API for XML

   We used SAX to parse the `EmploymentRecord.xml`. We override the startDocument, endDocument, startElement, endElement and characters functions implemented by the SAX Default handler. In the `startElement` function, if the element is `employment` we create a new Employment Java object and parse and store all values under that element to the object. When we then encounter the `employment` element in the `endElement` function we add the Empyment object to the employemnt record list and continue execution. When the parser has finished execution the objects can be retrived.

3. XSLT - Extensible Stylesheet Language Transformations

   We used XSLT to create a `ApplicantProfile.xml` from the four output XML documents.

4. JAXB - Java Architecture for XML Binding

   We used JAXB to parse the `Resume.xml` and `Transcript.xml`. We started by creating Java domain classes which the XML documents could be mapped to. The classes are tagged with `@XmlRootElement`, and properties tagged with `@XmlElement` or `@XmlAttribute`. When the classes have been created, the only thing left to do for the parsing is to un-marshal the XML document.

[1] `http://www.utilities-online.info/xsdvalidation/`

We also used JAXB to generate output XML documents from the Java instances that were created from parsing the input XML documents. The output files were simply made by marshaling the Java instances back to XML documents. We then manually checked that the output documents were exactly the same as the input documents.

The process should result in a complete applicant profile XML. That is, we use DOM, SAX and JAXB to parse the input XML documents to Java objects. We use JAXB to convert the Java objects back to output XML documents. Finally we use XSLT to merge and pick out elements from the output XML documents into one `ApplicantProfile.xml` document.

We also use XSLT on the original input XML documents (without unmarshalling to objects and marshalling again to XML) to create an ApplicantProfile.xml document directly.

2.3 Task 3

In this task we used XSLT to calculate the GPA of the applicant from his transcript, and put it into an appropriate place in the `ApplicantProfile` XML.

This was done by summing up the `gradedecimal` field for each `course` within a `degreetranscript` in `Transcript.xml` and then dividing it by the number of courses within that degree to get the average.