Homework #1
Web Service Programming
ID2208

XML Processing

KTH – ICT School
VT 2010

Administrative Issues

- We appreciate 2 members per group in all activities:
  - Home works
  - Project

- More than 2 members per group are not appreciated
  - Exceptions are considered case-wise

Course Mailing Lists

- Subscribe to student mailing list of the course:
  - https://mailman.ict.kth.se/mailman/listinfo/id2208_students
- Post your question from teachers to “teachers” mailing list: id2208_teachers@mailman.ict.kth.se
- If you faced any technical problem during homeworks, please send it first to “students” mailing list: id2208_students@mailman.ict.kth.se
- We continuously monitor the students mailing list, and get involved if necessary.

Home works

- **4** Home works
- In-time submission and approval of all Homeworks, gives you 5 Bonus points

- You can get: (maximum) 5 Bonus Points

Project

- **1** Project
- In-time submission and approval of Project, gives you 5 Bonus points

- You can get: (maximum) 5 Bonus Points

Total Bonus Points

- In Total you can get:

- **10** Bonus Points from Project + Homeworks
  - **5** homework bonus points
  - **5** project bonus points

Schedule

05-02-2010 14-02-2010	HW1 Session HW 1 Due	Demo sessions (day after deadline).
12-02-2010 18-02-2010	HW2 Session HW2 Due	Demo sessions (day after deadline).
19-02-2010 25-02-2010	HW3 Session HW3 Due	Demo sessions (day after deadline).
26-02-2010 04-03-2010	HW4 Session HW4 Due	Demo sessions (day after deadline).**Project Introduction**
21-03-2010	Project Due	**Hard Deadline**
	Project Demo	Project Demo session to be announce.
18-03-2010	Final Examination	

Homework1

- XML Processing

- **Aim**: Understanding and getting a hands-on experience with XML processing and transformation technologies

XML Processing

XML processing typically includes three phases:

1. Processing input XML

-Validating and Parsing XML documents (DOM, SAX)

- Querying and extracting information (XQuery)

- Associating the XML information to objects (JAXB)

2. Business Logic Process

- Processing information according to your business logic

3. Processing Output XML

- Building XML document model and directly serializing to XML

- Applying XSLT

Document Object Model (DOM)

DOM (1)

```
//Get a factory object for DocumentBuilder objects
    DocumentBuilderFactory factory =     DocumentBuilderFactory.newInstance();

// to make the parser a validating parse
    factory.setValidating(true);
//To parse a XML document with a namespace,
     factory.setNamespaceAware(true);
// to ignore  whitespace between elements.
     factory.setIgnoringElementContentWhitespace(true);

 // specifies the schema language for validation
     factory.setAttribute(
"http://java.sun.com/xml/jaxp/properties/schemaLanguage,"http://www.w3.org/2001/XMLSchema");
 //specifies the XML schema document to be used for validation.

 factory.setAttribute( "http://java.sun.com/xml/jaxp/properties/schemaSource",  "
YourXSDName“);
```

DOM (2)

```
//Get a DocumentBuilder (parser) object
 DocumentBuilder builder =  factory.newDocumentBuilder();
 //Parse the XML input file to create a document object that represents the input XML
file.
 Document document = builder.parse(new File(XMLFileName));

//Process the DOM tree, beginning with the document node to produce the output.
// For example :

Node root = document.getFirstChild()
NodeList children = root.getChildNodes();

for (Node child = root.getFirstChild();  child != null;
child = child.getNextSibling()) {

              processNode(child);
}
// look at sample DOM processing program
```

```
<xsd:schema>
<xsd:element name="transcript">
        <xsd:complexType>
                <xsd:sequence>
                        <xsd:element name="name" type="xsd:string"/>
                        <xsd:element name="university" type="xsd:string"/>
                        <xsd:element name="degree" type="xsd:string"/>
                        <xsd:element name="year" type="xsd:int"/>
                        <xsd:element name="courses">
                        <xsd:complexType>
                           <xsd:sequence>
                              <xsd:element name="course" minOccurs="0" maxOccurs="unbounded"/>
                           </xsd:sequence>
                        </xsd:complexType>
                        </xsd:element>
                </xsd:sequence>
        </xsd:complexType>
</xsd:element>
</xsd:schema>
```

SAX Parsing Model

SAX (1)

```
// Use an instance of ourselves as the SAX event handler
DefaultHandler handler = new SampleSAXParser();
// Use the default (non-validating) parser
SAXParserFactory factory = SAXParserFactory.newInstance();

SAXParser saxParser = factory.newSAXParser();
saxParser.parse(new File(XMLFileName), handler);
```

SAX (2)

```
.....
// Parse for Education Section

saxp.parse("Input.xml", new YourParserHandler(...));
```

```
 static class YourParserHandler extends DefaultHandler {
.............
```

SAX (3)

```
@Override
public void startDocument()   throws SAXException
 {    .....
 }

 @Override
  public void endDocument() throws SAXException
  {   ....
  }

@Override
 public void characters(char[] arg0, int arg1, int arg2) throws SAXException
  {  .........
  }
```

SAX (3)

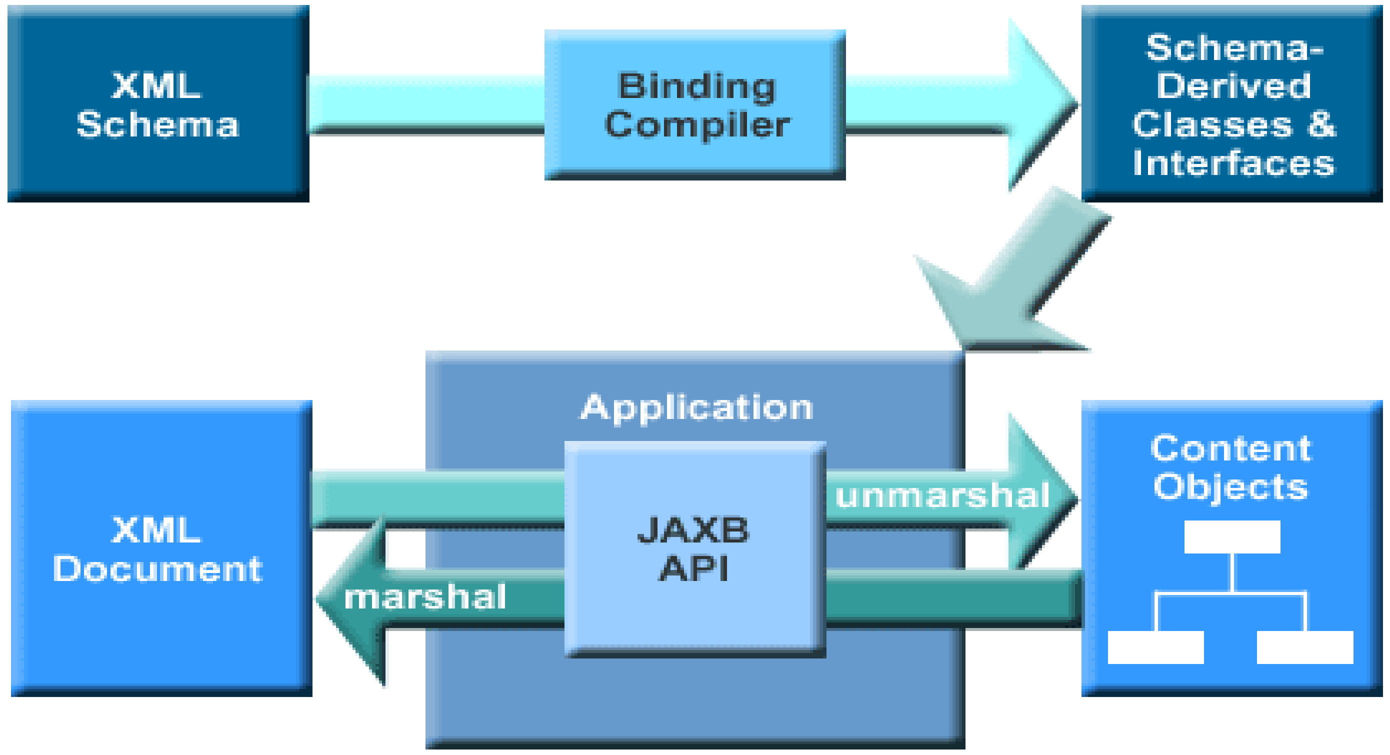
```
public void startElement(String namespaceURI,     String localName, // local
name   String  qName, // qualified name     Attributes   attrs)  throws
SAXException
  {
    //Start of Element  tag
  }

public void endElemen t(String namespaceURI,     String localName, // local
name   String qualifiedName // qualified name)      throws SAXException
  {
    //End of Element tag
  }
```

```
<xsd:schema>
<xsd:element name="transcript">
      <xsd:complexType>
            <xsd:sequence>
                  <xsd:element name="name" type="xsd:string"/>
                  <xsd:element name="university" type="xsd:string"/>
                  <xsd:element name="degree" type="xsd:string"/>
                  <xsd:element name="year" type="xsd:int"/>
                  <xsd:element name="courses">
                  <xsd:complexType>
                    <xsd:sequence>
                      <xsd:element name="course" minOccurs="0" maxOccurs="unbounded"/>
                    </xsd:sequence>
                  </xsd:complexType>
                  </xsd:element>
            </xsd:sequence>
      </xsd:complexType>
</xsd:element>
</xsd:schema>
```

JAVA API for XML Binding (JAXB)

JAXB (2) – XML Processing Model



JAXB XML processing model taken from :
http://java.sun.com/developer/technicalArticles/WebServices/jaxb/

JAXB

- JAXB : API and tools that automate the mapping between XML documents and Java objects

- Part of *SUN JWSDP* package

Binding Schema to Java Class

You can compile the edited XML schema into Java classes using XJC.

You can find XJC at : .../jwsdp/jaxb/bin/xjc

Assume Purchase Order Schema

(http://www.w3.org/TR/xmlschema-0/#po.xsd)

```
> xjc -p primer.po -d src po.xsd

  parsing a schema...
  compiling a schema...
  primer\po\impl\CommentImpl.java
  primer\po\impl\ItemsImpl.java
  primer\po\impl\JAXBVersion.java
  primer\po\impl\PurchaseOrderImpl.java
  primer\po\impl\USAddressImpl.java
  ...........
```

Marshaling (Java Object to XML) - 1

```
    // create a JAXBContext
       JAXBContext jc = JAXBContext.newInstance( "primer.po" );
       // create an ObjectFactory instance.
       ObjectFactory objFactory = new ObjectFactory();
       // create an empty PurchaseOrder
       PurchaseOrder po = objFactory.createPurchaseOrder();
       // manipulate   "po"  object
       ........


// create a Marshaller and marshal to System.out
 Marshaller m = jc.createMarshaller();

 m.setProperty( Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE );

m.marshal( po, System.out );
```

Java Object to XML (Unmarshaling)

Read yourself:

http://java.sun.com/developer/technicalArticles/WebServices/jaxb/

XSLT

XPath

XPath (1)

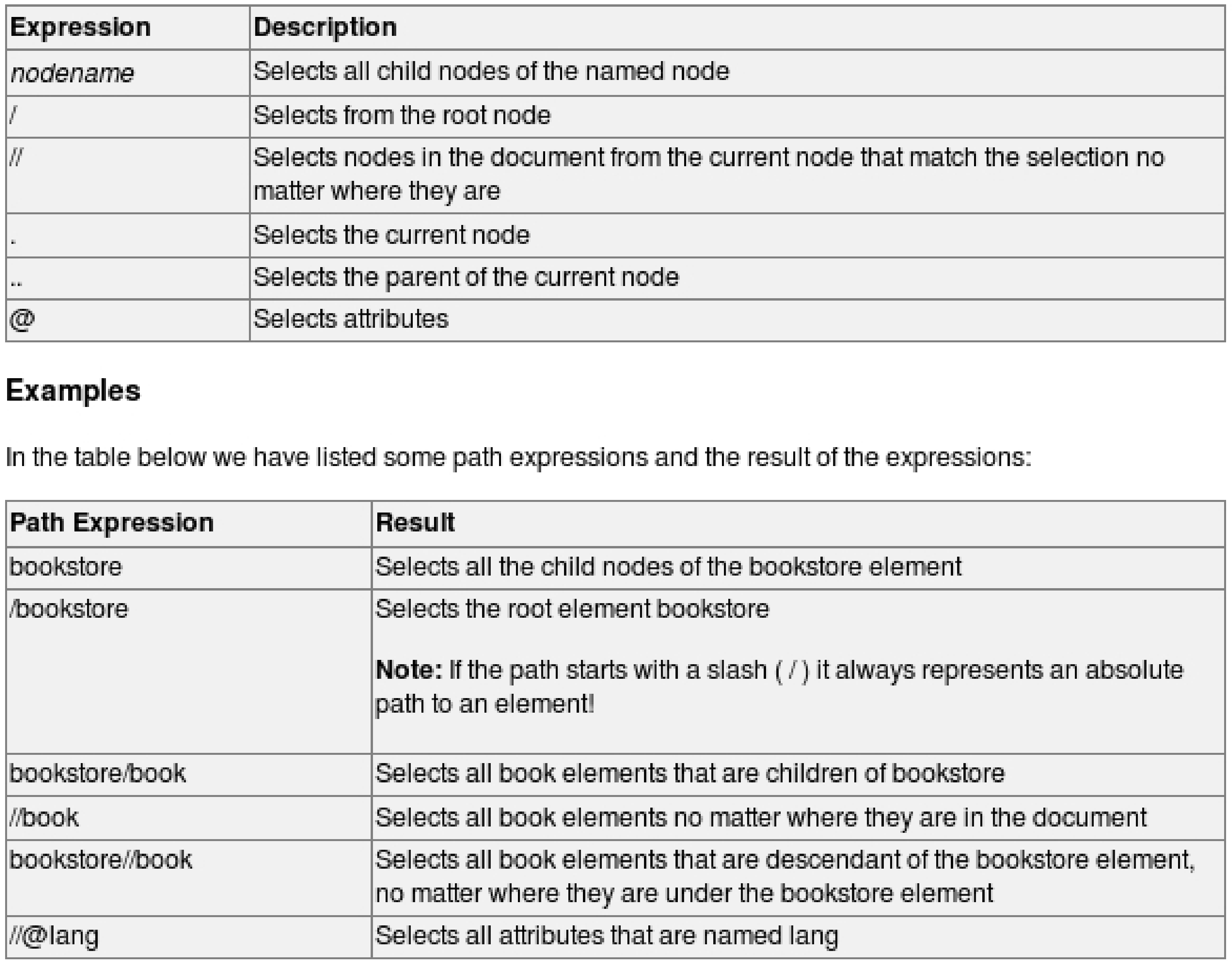
XPath is a language for finding information and to navigate through elements and attributes in an XML document .

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
<book>
    <title lang="eng">Harry Potter</title>
    <price>29.99</price>
</book>
<book>
        <title lang="eng">Learning XML</title>
    <price>39.95</price>
</book>
</bookstore>
```

Some materials for this presentation is taken from : (http://www.w3schools.com/xpath/default.asp)

XPath (2)

XPath: Path expressions to select nodes in an XML document.



Expression	Description
nodename	Selects all child nodes of the named node
/	Selects from the root node
//	Selects nodes in the document from the current node that match the selection no matter where they are
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

Examples

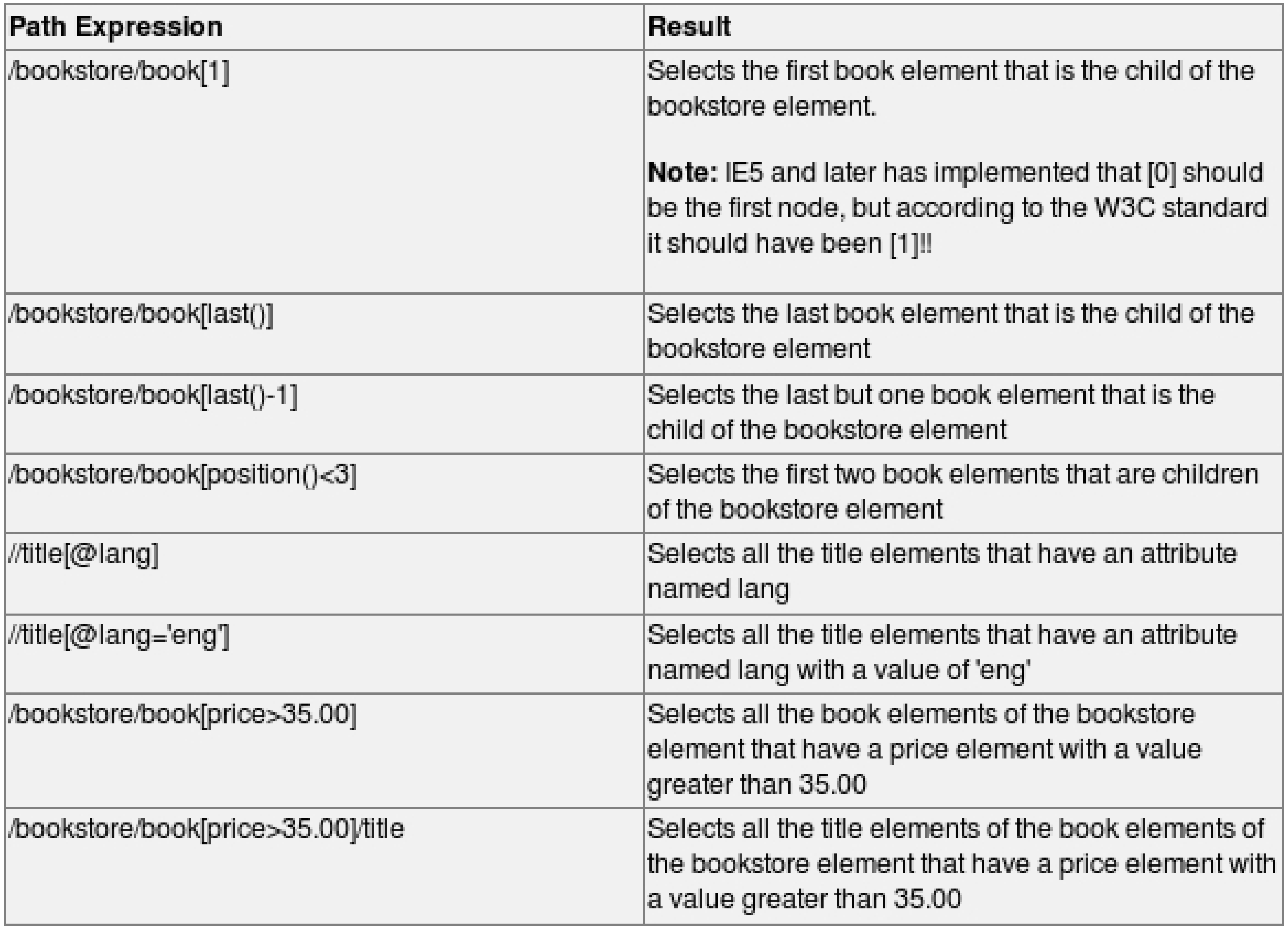
In the table below we have listed some path expressions and the result of the expressions:

Path Expression	Result
bookstore	Selects all the child nodes of the bookstore element
/bookstore	Selects the root element bookstore **Note:** If the path starts with a slash (/) it always represents an absolute path to an element!
bookstore/book	Selects all book elements that are children of bookstore
//book	Selects all book elements no matter where they are in the document
bookstore//book	Selects all book elements that are descendant of the bookstore element, no matter where they are under the bookstore element
//@lang	Selects all attributes that are named lang

XPath (4)

XPath follows a hierarchical pattern to select elements.

Use predicates ([ ])to find a specific node or a node that contains a specific value. http://www.w3schools.com/



Path Expression	Result
/bookstore/book[1]	Selects the first book element that is the child of the bookstore element. **Note:** IE5 and later has implemented that [0] should be the first node, but according to the W3C standard it should have been [1]!!
/bookstore/book[last()]	Selects the last book element that is the child of the bookstore element
/bookstore/book[last()-1]	Selects the last but one book element that is the child of the bookstore element
/bookstore/book[position()<3]	Selects the first two book elements that are children of the bookstore element
//title[@lang]	Selects all the title elements that have an attribute named lang
//title[@lang='eng']	Selects all the title elements that have an attribute named lang with a value of 'eng'
/bookstore/book[price>35.00]	Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00
/bookstore/book[price>35.00]/title	Selects all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00

XSLT

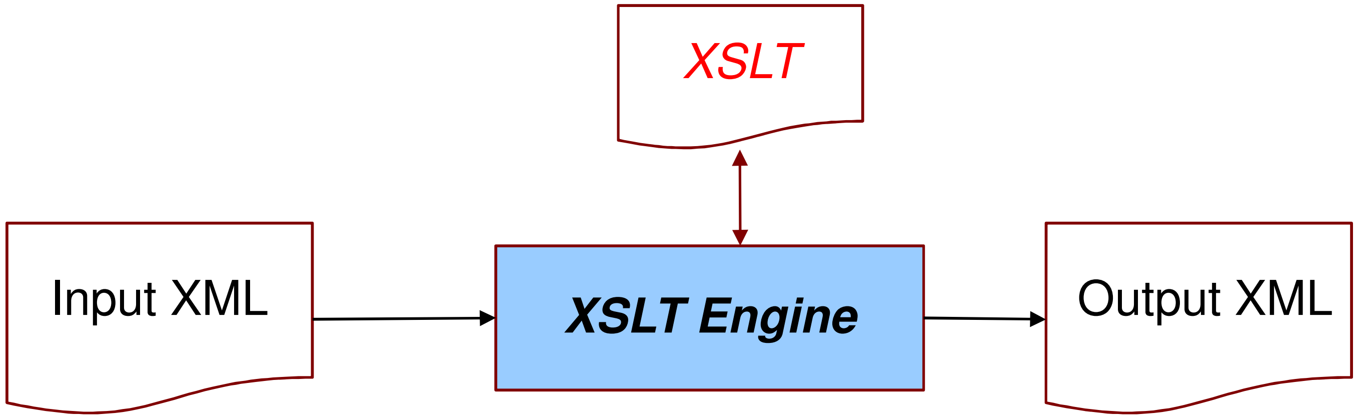
XSLT

Extensible Stylesheet Language Transformation (XSLT) is originally develop an XML-based Stylesheet Language.

In this homework, We use XSLT+ XPTH to specify conversion from one XML document to another format XML.

But XSLT is much more! If you want to know more ,look at: http://www.w3schools.com/xsl/default.asp

Applying XSLT for XML Transformation



XSLT Basics

A Rule based language.

A Rule (template rule) consists of:

1-A "*matching* pattern", to match against XML elements specified using XPath expressions. Example:

```
<xsl:template match="XPath Expression">
```

2. A "template" which defines format of output document whenever an XML element fits to the matching pattern. Example:

```
<xsl:element name="....">
 </xsl:element>
OR...
<xsl:value-of select="...."/>
OR....
```

Input XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<priceList>
  <coffee>
    <name> Santos</name>
    <price>11.95</price>
    <producer>Brazil</producer>
  </coffee>
  <coffee>
    <name>Colombia</name>
    <price>12.50</price>
    <producer>JuanValdez</producer>
  </coffee>
</priceList>
```

Traget XML

```
<Coffee >
    <CoffeePrice> coffe price<CoffeePrice>
    <CoffeeProducer> coffee produce <CoffeeProducer>
  <Coffee>
```

Designed XSLT

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

 <xsl:template match="/">
    <xsl:element name="ns:Coffee">

      <xsl:element name="ns:CoffeePrice">
           <xsl:value-of select="/priceList/coffee/price"/>
          </xsl:element>

      <xsl:element name="ns:CoffeeProducer">
      <xsl:value-of select="/priceList/coffee/producer"/>
      </xsl:element>
    </xsl:element>

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

Homeworks!

Problem Description -1

- We would like to simulate an “Employment Service Company” like *ManPower, AcademicWork ,Komet* ... The main task of such companies is to create a profiles of job seekers and match them with the advertised jobs by different companies.

Problem Description -2

The profile is made of CV, relevant academic degree(s) and previous working experiences, information about companies the applicant worked for before, motivation letter,places desire to work, type of job (permanent, part time, contract,...) , references and other relevant qualifications (e.g. driving license).

Problem Description -3

- The idea is to design a machinery way to collect the required information/documents from different sources to create applicant profile automatically.
  - All documents are provided in XML format.
- The content of the profile can be obtained from following sources:
  - Degree and Transcript issued by “University”
  - Employment Records from “Employment Office”
  - Information of Companies from an Online Service (database of Companies)
  - Short CV and other materials provided by the applicant while registering in “Employment Service Company”

Tasks (1)

- Create appropriate schema (XSD) for each XML document (Transcript, Employment Record, Company Info, short CV and Applicant Profile).
- Generate sample documents (XMLs) out of those schema and populate the content (assume that applicant have at least one previous working experience , one academic degree, ...) and Validate them against your schema .
- Use *Namespace* in your Schemas

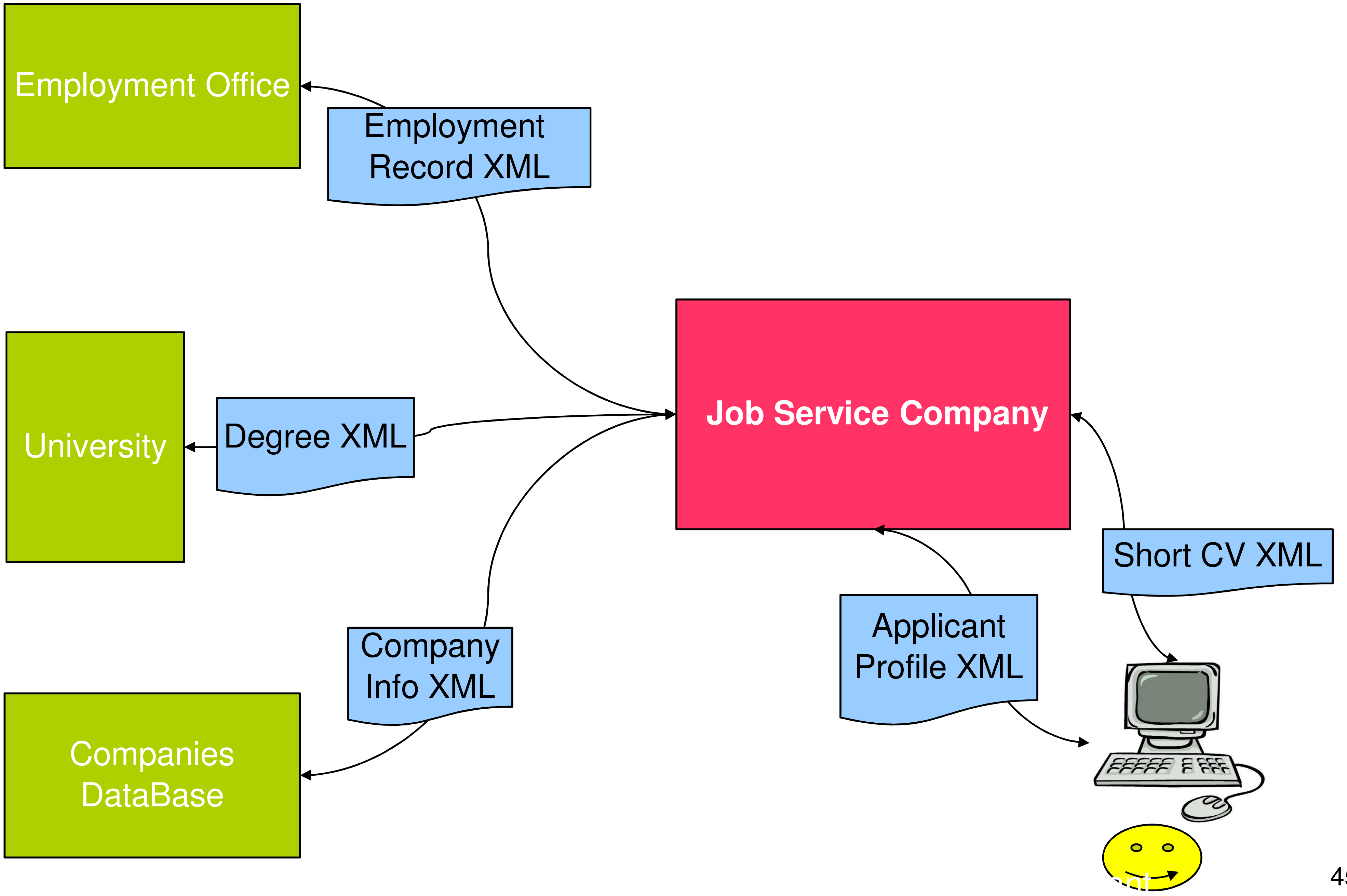
Tasks (2)

- Write programs to map the relevant piece of information from collected documents into Applicant Profile through ALL FOUR different mechanisms (4 different programs in total).
  - Document Object Model (DOM)
  - Simple API for XML (SAX)
  - Extensible Stylesheet Language Transformations (XSLT)
  - JAXB

Tasks (3)

- As a part of program functionality, it should able to calculate the GPA form Transcript and put it in appropriate place in User Profile, while mapping academic records to User Profile.
- The output of the above mentioned programs will be the complete User Profile in XML format.

Interactions



XML Proceeing librareis in JWSDP

- Download and install Java Web Service Developer Pack (JWSDP 2.0) from: http://java.sun.com/webservices/downloads/previous/webservicespack.jsp
- Unzip
  You can find required libraries for XML processing in .../jwsdp/jaxb/lib and .../jwsdp/jaxp/lib folder in the installed directories.

Development Environments

You are free to use whatever IDE (NetBeans, Eclipse, ...) you desire.

Recent versions of NetBeans includes XSLT module.

You can find XSLT plug-ins for Eclipse at:

http://eclipsexslt.sourceforge.net/ ,

http://wiki.eclipse.org/XSLT_Project

Deliverables

- The XSDs (5 xsd files)
- The 4 populated XML documents (Transcript, Employment Record,...)
- The source code of the “ mapping programs” including designed “xslt” file.
- The generated Applicant Profile.
- You SHOULD demonstrate your work in a presentation.

In your work, you SHOULD:

- Give suitable and human understandable names to XML tags.
- In the designed schema, we expect to see:
  - Complex and Simple types, Attributes and Elements.
  - Using Restrictions (at least THREE per schema) to narrow the ranges of values or formating the values which an element could take.
  - Using Extension (at least TWO) .

Warning !

- Your are not allowed to use the “tools” for mapping schemas automatically and also those which generate “XSLT” automatically. (It is simple, do it yourself , you will learn more!) To prevent this, you might be asked to make some changes in delivered XSD or XSLTs during presentation and you SHOULD able to do that correctly.

HW #1 - Delivery

- Send your deliverables by e-mail to BOTH:

  shahabm@kth.se and nimad@kth.se

  e-mail subject: PWS10-HW1

  Please add your names in the body of the email

  Attach: source code + instructions how to run your code,

Deadline : 14 Feb. 2010, 11:59 PM CET

- Presentation: 15 Feb 2009 (somewhere in 6th floor)

GOOD LUCK!

Remember Two Principles

- Share your Experience but Not Deliverables
- **KISS**

Keep It Simple Stupid!

KiSS

Useful Materials

- XSLT:

http://www.globalguideline.com/xslt/XSLT_Introduction.php
http://www.w3schools.com/xsl/default.asp

http://www.learn-xslt-tutorial.com/

http://www.zvon.org/xxl/XSLTutorial/Output/contents.html#id2

- SAX , DOM, JAXB:

http://totheriver.com/learn/xml/xmltutorial.html

http://java.sun.com/webservices/docs/2.0/tutorial/doc/index.html

http://java.sun.com/developer/technicalArticles/WebServices/jaxb/

- Sample schema:

http://www.brianjlandau.com/xml/university.xsd.html