

# XML and JSON Processing

## Lecture 7

# XML

- XML (eXtensible Markup Language) has been used in the Java EE platform since the beginning of deployment descriptors and metadata information.

```
<?xml version="1.0" encoding="UTF-8" ?>
<order id="1234" date="05/06/2013">
  <customer first_name="James" last_name="Rorrison">
    <email>j.rorri@me.com</email>
    <phoneNumber>+44 1234 1234</phoneNumber>
  </customer>
  <content>
    <order_line item="H2G2" quantity="1">
      <unit_price>23.5</unit_price>
    </order_line>
    <order_line item="Harry Potter" quantity="2">
      <unit_price>34.99</unit_price>
    </order_line>
  </content>
  <credit_card number="1357" expiry_date="10/13"
  control_number="234" type="Visa"/>
</order>
```

# XSD

- XML Schema Definition (XSD) is an XML-based grammar declaration used to describe the structure and content of an XML document

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema version="1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="order" type="order"/>
  <xs:complexType name="creditCard">
    <xs:sequence>
      <xs:attribute name="number" type="xs:string"/>
      <xs:attribute name="expiry_date" type="xs:string"/>
      <xs:attribute name="control_number" type="xs:int"/>
      <xs:attribute name="type" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType ... />
</xs:schema>
```

# Parsing with SAX and DOM

- The Document Object Model (DOM) API requires reading the entire XML structure and holding the object tree in memory.
- The Simple API for XML (SAX) is an event-driven, serial-access mechanism that does element-by-element processing

# Querying in XML

- XPath

```
//content/order_line[unit_price>=20]/unit_price/text()
```

- XQuery

```
for $orderLine in
```

```
doc("order.xml")//content/order_line[@quantity>1]
```

```
where $orderLine/unit_price < 50
```

```
return $orderLine/unit_price/text()
```

# XSLT

#

eXtensible Stylesheet Language Transformations (XSLT ) allows to transform an XML document from one vocabulary to another, in a generic manner

<http://www.w3schools.com/xsl/tryxslt.asp?xmlfile=cdcatalog&xsltfile=cdcatalog>



# JAXB

Java Architecture for XML Binding (JAXB) allows developers to work with Java objects that represent XML documents

- ✓ `@XmlRootElement` annotation
- ✓ `marshaller.marshal(person, writer);`
- ✓ Generate XSD from Java class
  - `jaxbContext.generateSchema(sor);`

# JSON

- JSON (JavaScript Object Notation) originated with JavaScript for representing simple data structures

```
{
  "order": {
    "id": "1234",
    "date": "05/06/2013",
    "customer": {
      "first_name": "James",
      "last_name": "Rorrison",
      "email": "j.rorri@me.com",
      "phoneNumber": "+44 1234 1234"
    }
  }
}
```

# JSON Processing

#

- `javax.json.JsonObject`
- `javax.json.JsonArray`
- `javax.json.Json`

OR

- `org.json.JSONObject`
- `org.json.JSONArray`

# To read

- Beginning Java EE 7. Chapter 12  
(<http://www.goodreads.com/book/show/18979704-beginning-java-ee-7>)

# Homework

Please, select list of Persons (age > 30, name starts with "A") from database via JDBC, save this data to file `person_selection.json` in json format.

Later, please, read this file and transform its content into `List<Person>` and print this list sorted by name and age in ascending order with `System.out.println`