XML & XML PROCESSING

PREREQUISITES JAVA WEB SERVICE

Rajeev Gupta MTech CS
rgupta.mtech@gmail.com

Session-1

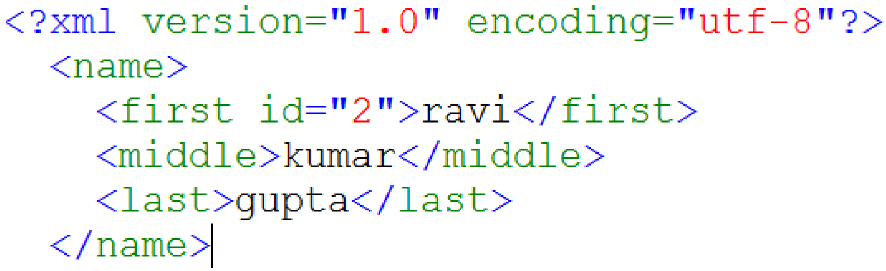
Agenda

- Introduction to XML
- Well formed XML
- NameSpace
- XML validation
  - DTD
  - Schema

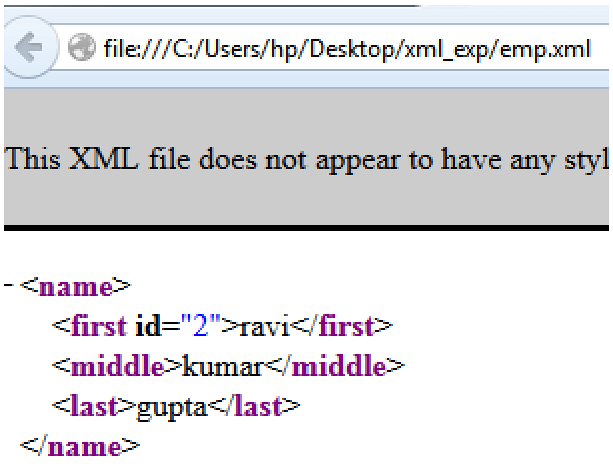
<?xml?>

Introduction to XML

- □ XML vs. HTML
  - Predefined Tags vs. extended (user define) markup language
- □ binary data vs. text data
- □ Idea of universal data format: SGML
  - there were a format that combined the universality of text files with the efficiency and rich information storage capabilities of binary files?
- □ XML parsers
  - parsers read XML syntax and extract the information for us



```
<?xml version="1.0" encoding="utf-8"?>
  <name>
    <first id="2">ravi</first>
    <middle>kumar</middle>
    <last>gupta</last>
  </name>
```



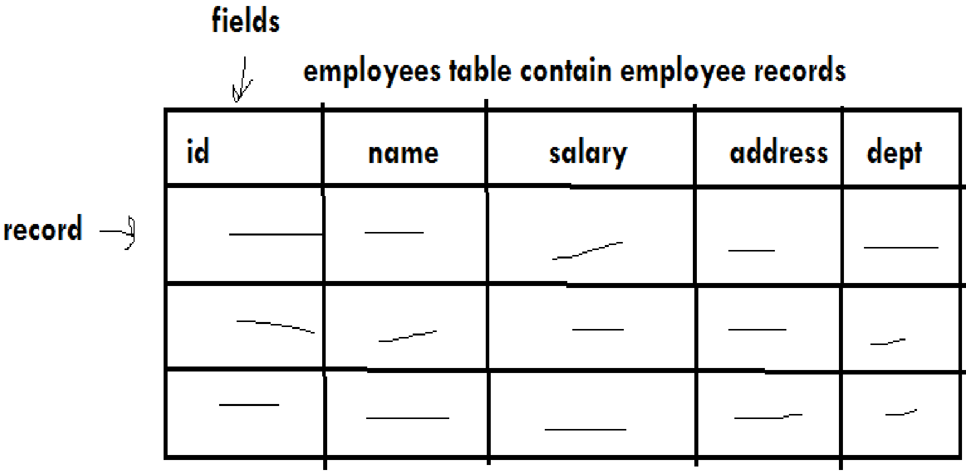
file:///C:/Users/hp/Desktop/xml_exp/emp.xml

This XML file does not appear to have any styl

```
- <name>
    <first id="2">ravi</first>
    <middle>kumar</middle>
    <last>gupta</last>
  </name>
```

XML comparing with RDBMS

- □ XML is very similar to database concept
- □ XML can be used to store , retrieve and transmit data
  - XML databases
  - **Most of the time XML is used for data interchange bw two system in inter-operatable way**


fields

employees table contain employee records

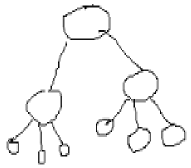
record

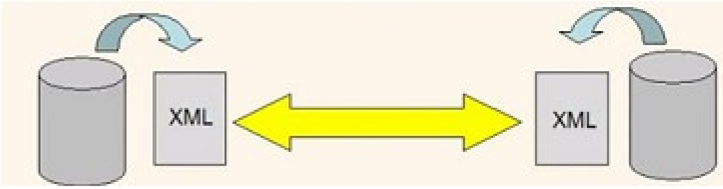
id	name	salary	address	dept

data ?

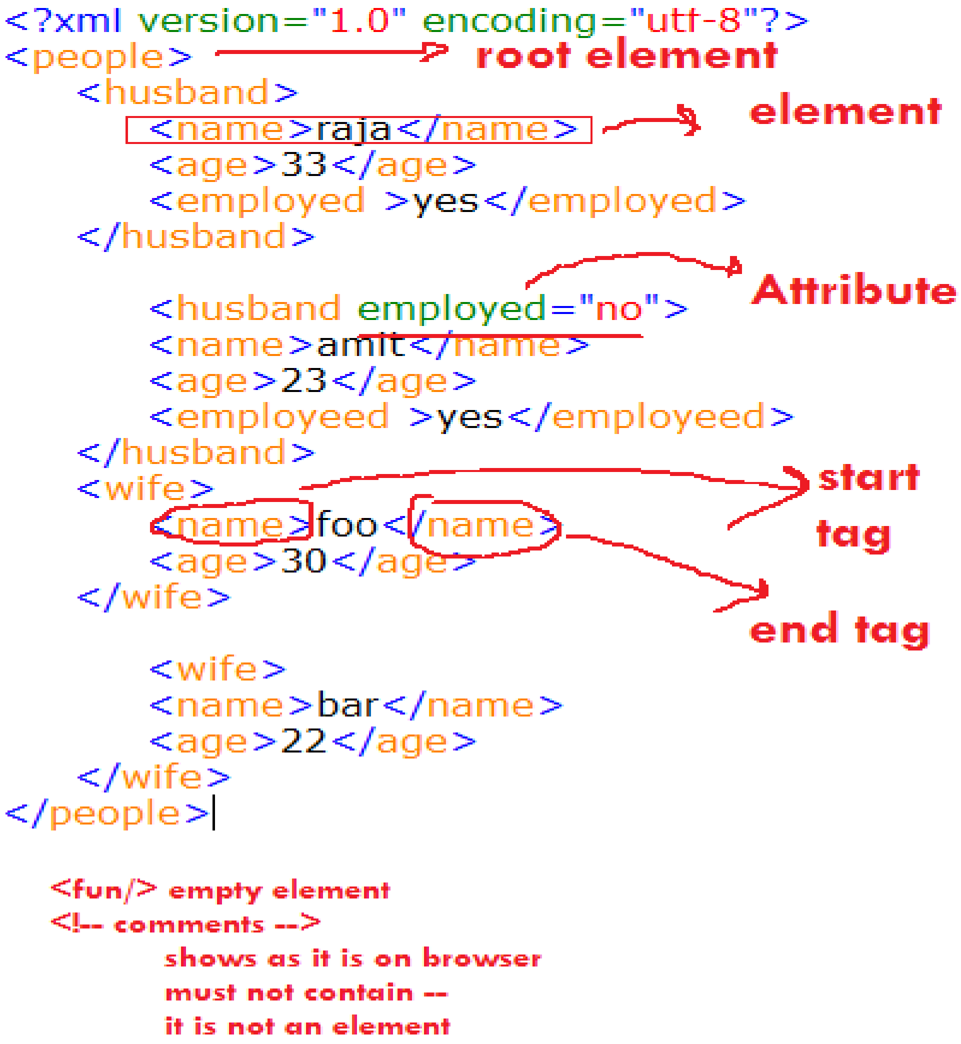
meta data?

constraints on data?





Nut and Bolt of XML



```
<?xml version="1.0" encoding="utf-8"?>
<people>
    <husband>
        <name>raja</name>
        <age>33</age>
        <employed >yes</employed>
    </husband>

        <husband employed="no">
        <name>amit</name>
        <age>23</age>
        <employeed >yes</employeed>
    </husband>
    <wife>
        <name>foo</name>
        <age>30</age>
    </wife>

        <wife>
        <name>bar</name>
        <age>22</age>
    </wife>
</people>
```

<fun/> empty element

<!-- comments -->

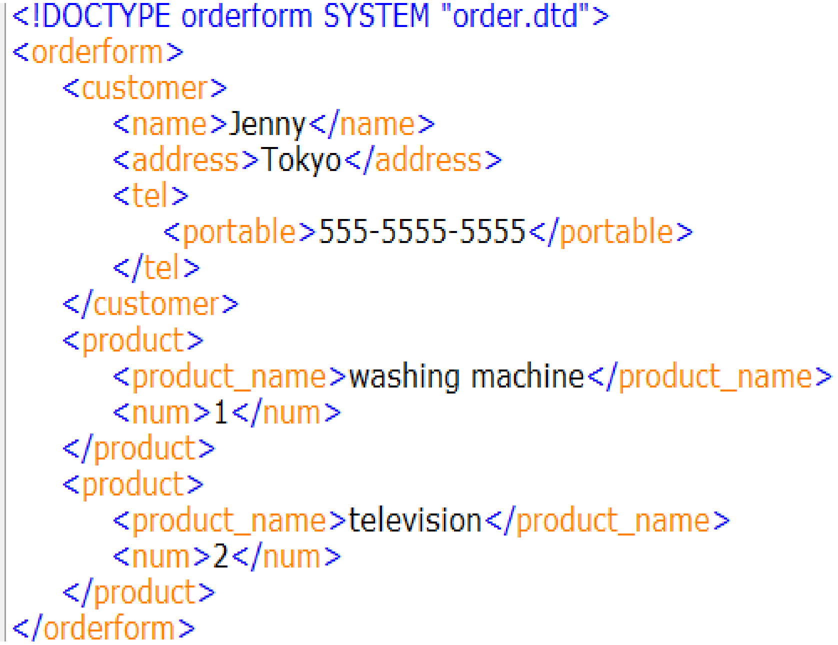
shows as it is on browser

must not contain --

it is not an element

Well formed XML

- Every start-tag must have a matching end-tag, or be a self-closing tag.
- Tags can’t overlap; elements must be properly nested.
- XML documents can have only one root element
- Element names must obey XML naming conventions
- XML is case sensitive
- XML will keep whitespace in your PCDATA ie In XML, the whitespace stays
- Example : Order.xml



```
<!DOCTYPE orderform SYSTEM "order.dtd">
<orderform>
    <customer>
        <name>Jenny</name>
        <address>Tokyo</address>
        <tel>
            <portable>555-5555-5555</portable>
        </tel>
    </customer>
    <product>
        <product_name>washing machine</product_name>
        <num>1</num>
    </product>
    <product>
        <product_name>television</product_name>
        <num>2</num>
    </product>
</orderform>
```

Entity Reference

- Well formed XML can not have symbol like > < &, we need to use Entity in place of them
  - <comparison>6 is &lt; 7 &amp; 7 &gt; 6 </comparison>
  - &amp;—the & character
  - &lt; —the < character
  - &gt;—the > character
  - &apos;—the ‘ character
  - &quot;—the “ character

CDATA character data

- If you have a lot of < and & characters that need escaping, you may find that your document become ugly
- Using CDATA sections, you can tell the XML parser not to parse the text, but to let it all go by until it gets to the end of the section.
- CDATA sections look like this:
- <comparison><![CDATA[6 is < 7 & 7 > 6]]></comparison>

```
<script language=’JavaScript’><![CDATA[
function myFunc()
{
if(0 < 1 && 1 < 2)
alert(“Hello”);
}
]]></script>
```

DTD valid xml

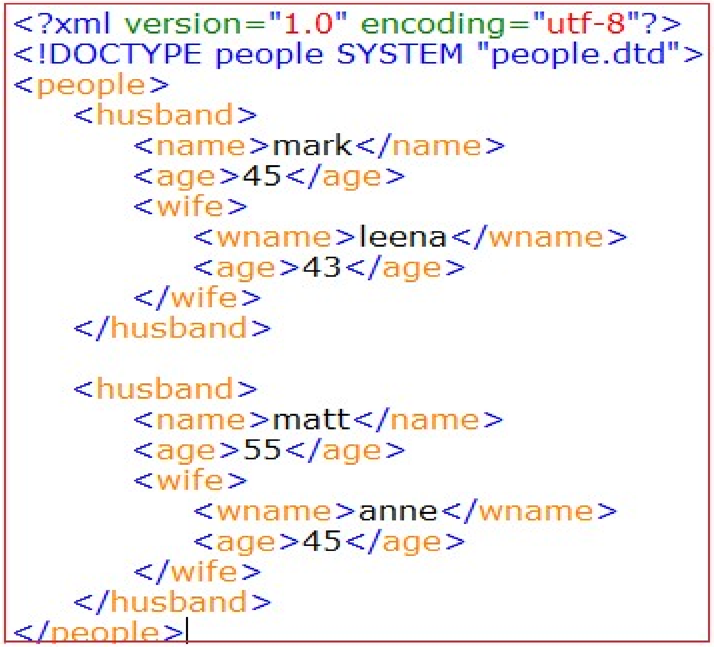
- Need of Schema?
  - Two parties must agree on some agreement
  - Aka contract bw two software systems
- **What schema tell us?**
  - What data is allowed?
  - What data is required?
  - How data is organized?
- An Valid XML document must be valid but reverse may not be true

XML
Validation
XML
All Schema Languages
All Major Validation Engines

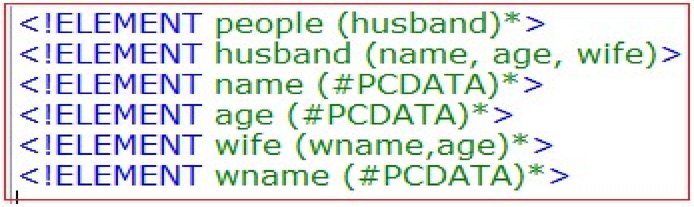
DTD (Document Type Definition)

- A way to validate xml document.
  - What DTD can tell us
    - What element can appear/must appear?
    - what should be order of element?
    - Information about data that contain?
    - attribute information required?
  - What DTD do not tell?
    - DTD do not support namespace concept
    - data type
    - Semantic meaning of an element(whether data is name or date?)

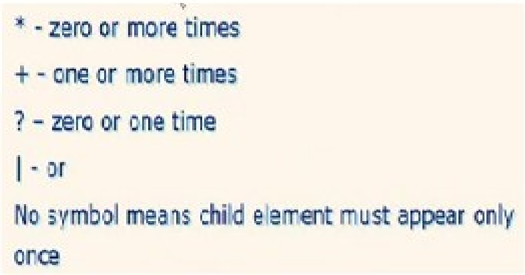
Understanding DTD



```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE people SYSTEM "people.dtd">
<people>
    <husband>
        <name>mark</name>
        <age>45</age>
        <wife>
            <wname>leena</wname>
            <age>43</age>
        </wife>
    </husband>

    <husband>
        <name>matt</name>
        <age>55</age>
        <wife>
            <wname>anne</wname>
            <age>45</age>
        </wife>
    </husband>
</people>
```



```
<!ELEMENT people (husband)*>
<!ELEMENT husband (name, age, wife)>
<!ELEMENT name (#PCDATA)*>
<!ELEMENT age (#PCDATA)*>
<!ELEMENT wife (wname,age)*>
<!ELEMENT wname (#PCDATA)*>
```



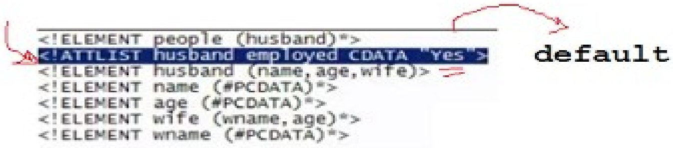
* - zero or more times

+ - one or more times

? – zero or one time

| - or

No symbol means child element must appear only once

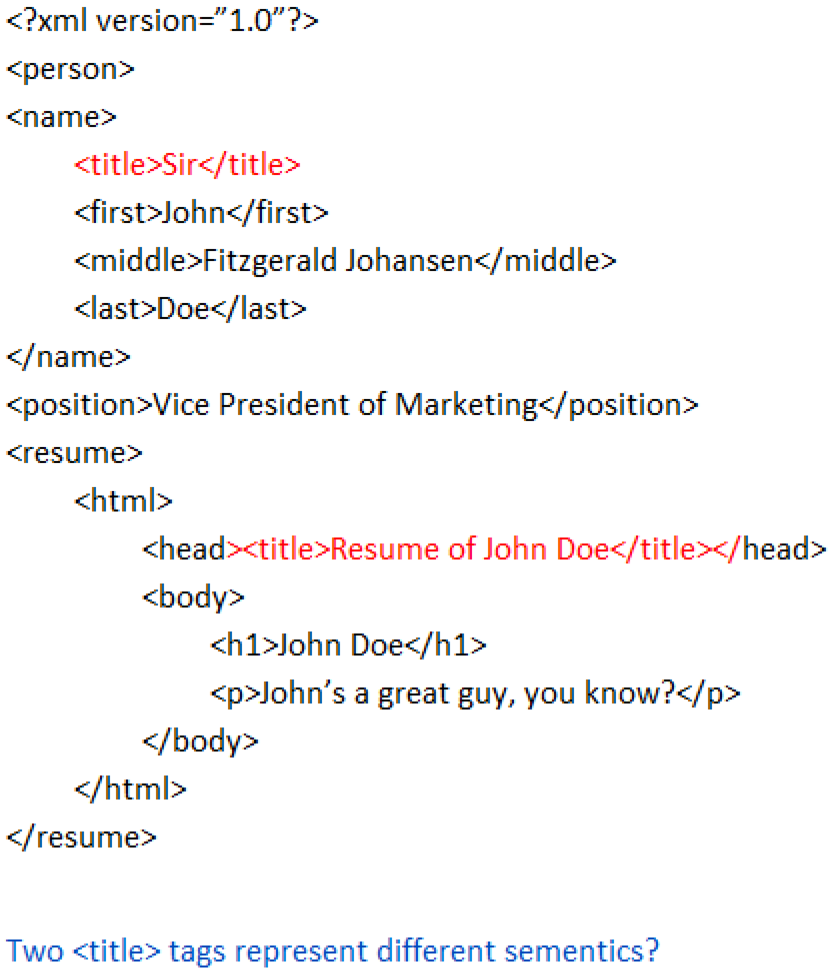


```
<!ELEMENT people (husband)*>
<!ATTLIST husband employed CDATA "Yes">
<!ELEMENT husband (name,age,wife)>
<!ELEMENT name (#PCDATA)*>
<!ELEMENT age (#PCDATA)*>
<!ELEMENT wife (wname,age)*>
<!ELEMENT wname (#PCDATA)*>
```

- Attributes in XML
  - <!ATTLIST contact person ID #REQUIRED>
  - In the document you could add the unique ID: <contact person=”Jeff_Rafter”>

XML Namespaces

- A namespace is a purely abstract entity, aka java package
- It’s nothing more than a group of names that belong with each other conceptually
- **Two <title> tags represent different semantics?**

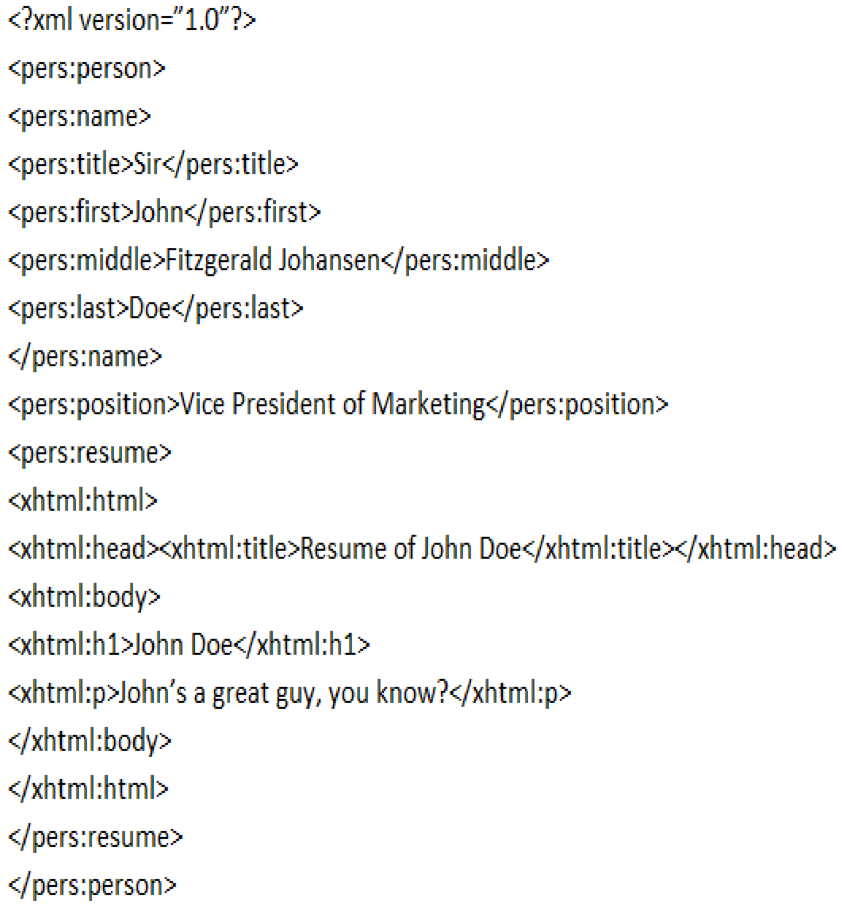


```
<?xml version=”1.0”?>
<person>
<name>
    <title>Sir</title>
    <first>John</first>
    <middle>Fitzgerald Johansen</middle>
    <last>Doe</last>
</name>
<position>Vice President of Marketing</position>
<resume>
    <html>
        <head><title>Resume of John Doe</title></head>
        <body>
            <h1>John Doe</h1>
            <p>John’s a great guy, you know?</p>
        </body>
    </html>
</resume>
```

Two <title> tags represent different sementics?

Approach 1 Using prefix

- **Problem with approach 1**
  - Who will monitor the prefixes?
  - The whole reason for using them is to distinguish names from different document types, but if it is going to work, then the prefixes themselves also have to be unique



```
<?xml version=”1.0”?>
<pers:person>
<pers:name>
<pers:title>Sir</pers:title>
<pers:first>John</pers:first>
<pers:middle>Fitzgerald Johansen</pers:middle>
<pers:last>Doe</pers:last>
</pers:name>
<pers:position>Vice President of Marketing</pers:position>
<pers:resume>
<xhtml:html>
<xhtml:head><xhtml:title>Resume of John Doe</xhtml:title></xhtml:head>
<xhtml:body>
<xhtml:h1>John Doe</xhtml:h1>
<xhtml:p>John’s a great guy, you know?</xhtml:p>
</xhtml:body>
</xhtml:html>
</pers:resume>
</pers:person>
```

Approach 2 using URI

- A URI (Uniform Resource Identifier) is a string of characters that identifies a resource, URI may be an valid URL

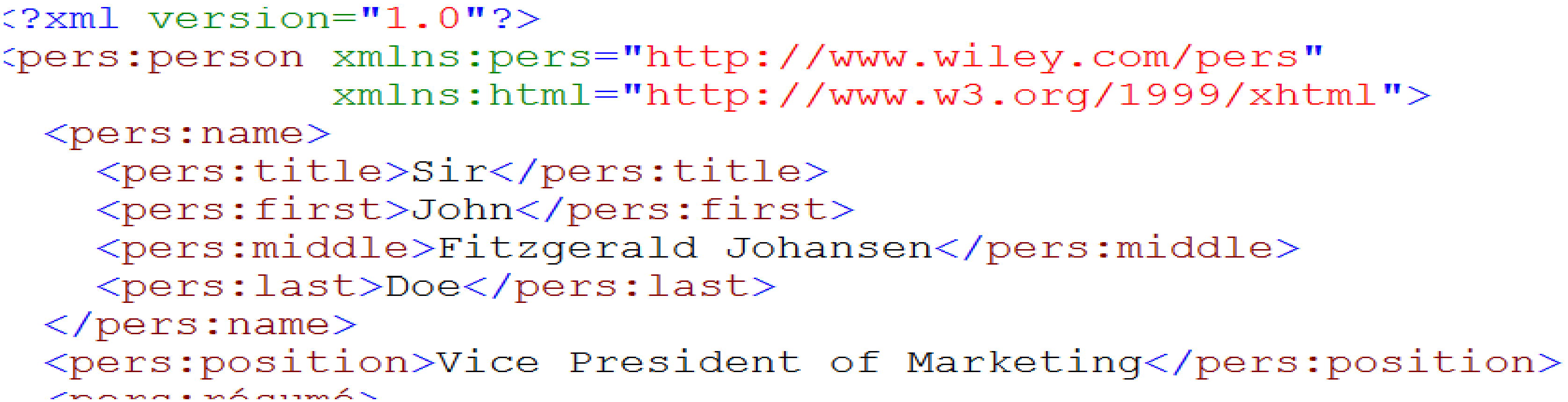
  <?xml version=”1.0”?>

  <{http://www.wiley.com/pers}person>

  <{http://www.wiley.com/pers}name>

  <{http://www.wiley.com/pers}title>

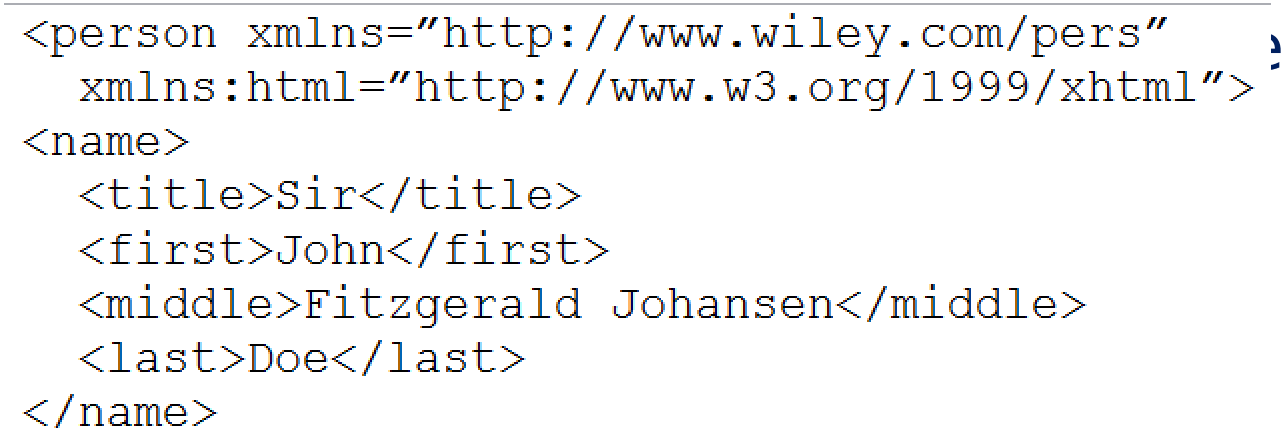
-



```
<?xml version="1.0"?>
<pers:person xmlns:pers="http://www.wiley.com/pers"
             xmlns:html="http://www.w3.org/1999/xhtml">
  <pers:name>
    <pers:title>Sir</pers:title>
    <pers:first>John</pers:first>
    <pers:middle>Fitzgerald Johansen</pers:middle>
    <pers:last>Doe</pers:last>
  </pers:name>
  <pers:position>Vice President of Marketing</pers:position>
  <pers:résumé>
```

Default namespace

- We can use default namespace to simplify last approach
- We can make one of the namespace as default Here xmlns=”http://www.wiley.com/pers” is default namespace
- A espace in a



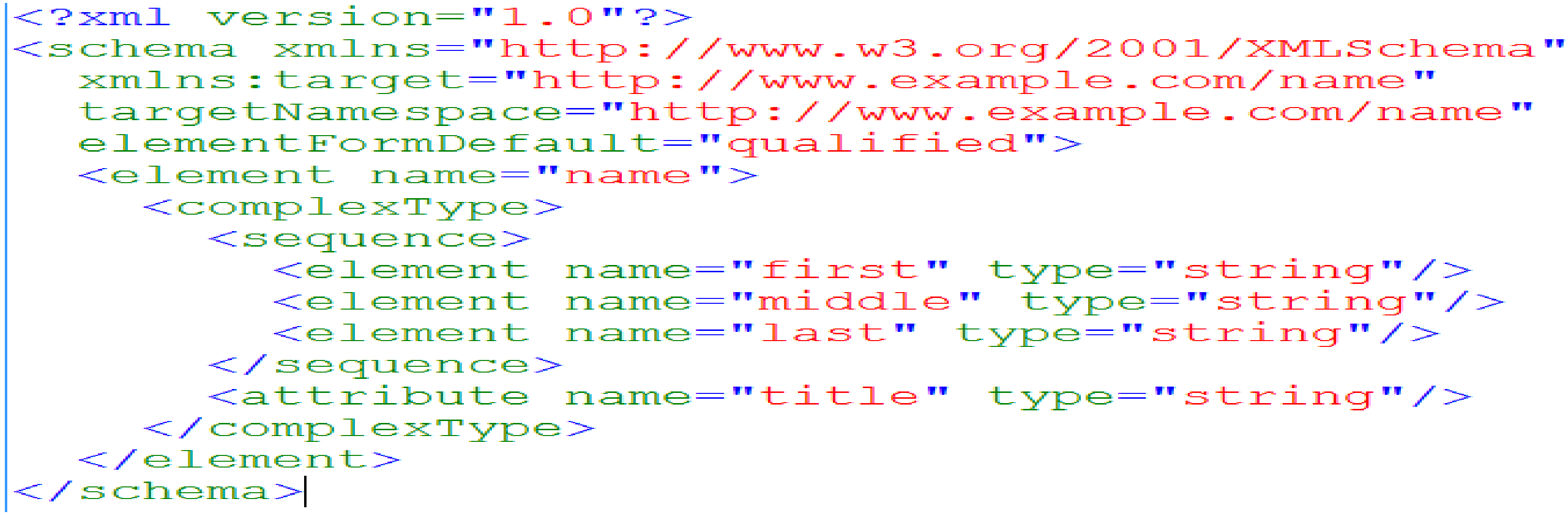
```
<person xmlns=“http://www.wiley.com/pers“
  xmlns:html=“http://www.w3.org/1999/xhtml“>
<name>
  <title>Sir</title>
  <first>John</first>
  <middle>Fitzgerald Johansen</middle>
  <last>Doe</last>
</name>
```

xml Schema

- **Why XML Schema?**
  - XML Schemas are created using basic XML, while DTDs utilize a separate syntax.
  - XML Schemas fully support the Namespace Recommendation.
  - XML Schemas enable you to validate text element content based on built-in and user-defined data types
  - XML Schemas enable you to more easily create complex and reusable content models
  - XML Schemas enable the modelling of programming concepts such as object inheritance and type substitution

Example: name.xsd

```
<?xml version="1.0"?>
<name xmlns="http://www.example.com/name"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.example.com/name name.xsd"
  title="Mr.">
   <first>John</first>
   <middle>Fitzgerald Johansen</middle>
   <last>Doe</last>
</name>
```



```
<?xml version="1.0"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:target="http://www.example.com/name"
  targetNamespace="http://www.example.com/name"
  elementFormDefault="qualified">
  <element name="name">
    <complexType>
      <sequence>
        <element name="first" type="string"/>
        <element name="middle" type="string"/>
        <element name="last" type="string"/>
      </sequence>
      <attribute name="title" type="string"/>
    </complexType>
  </element>
</schema>
```

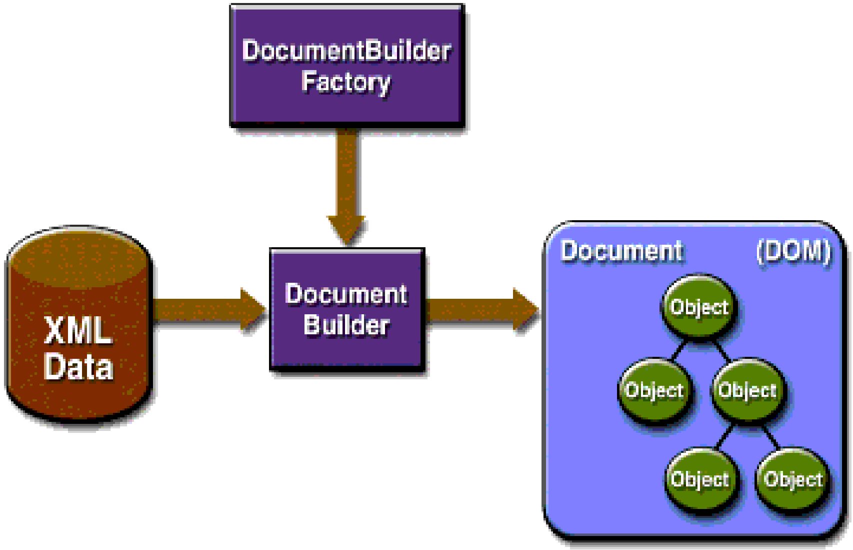
Session-2

Agenda

- Introduction to XML data processing
- XML processing techniques
  - DOM
  - SAX
  - JAXB
  - JSON

DOM XML Parser

- The DOM is the easiest to use Java XML Parser.
- It parses an entire XML document and load it into memory, modelling it with Object for easy model traversal.
- DOM Parser is slow and consume a lot memory if it load a XML document which contains a lot of data.



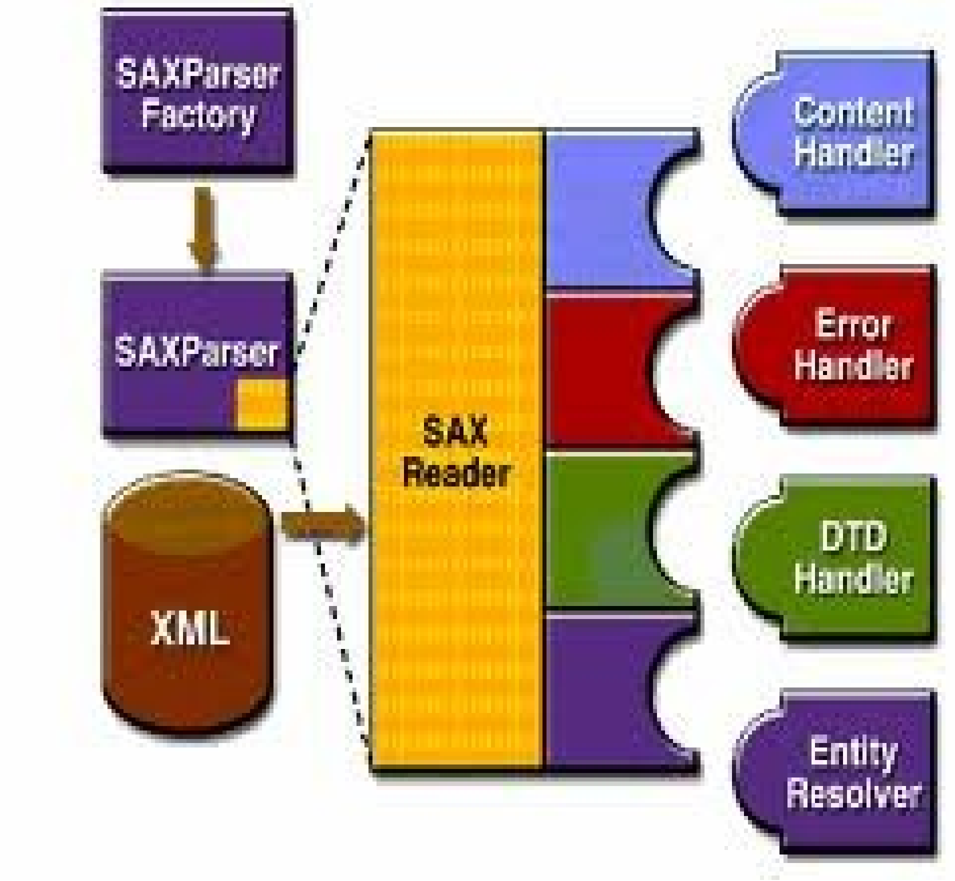
XML DOM processing

- Reading XML file in Java using DOM
- DOM parser to modify an existing XML file
  - Add a new element
  - Update existing element attribute
  - Update existing element value
  - Delete existing element

http://www.mkyong.com/java/how-to-read-xml-file-in-java-dom-parser/

SAX XML Parser

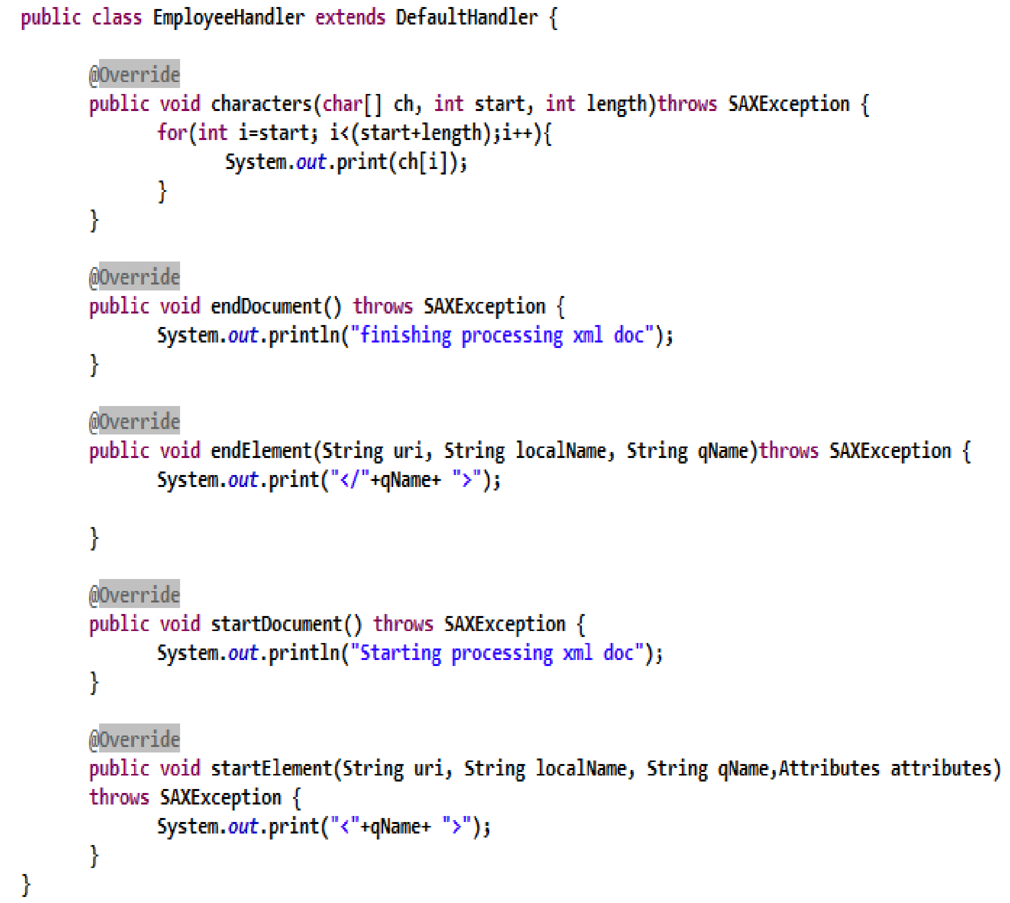
- SAX parser use callback functions (org.xml.sax.helpers.DefaultHandler) to informs clients about the XML document structure
- It is faster then DOM
- Do not require complete xml document in advance



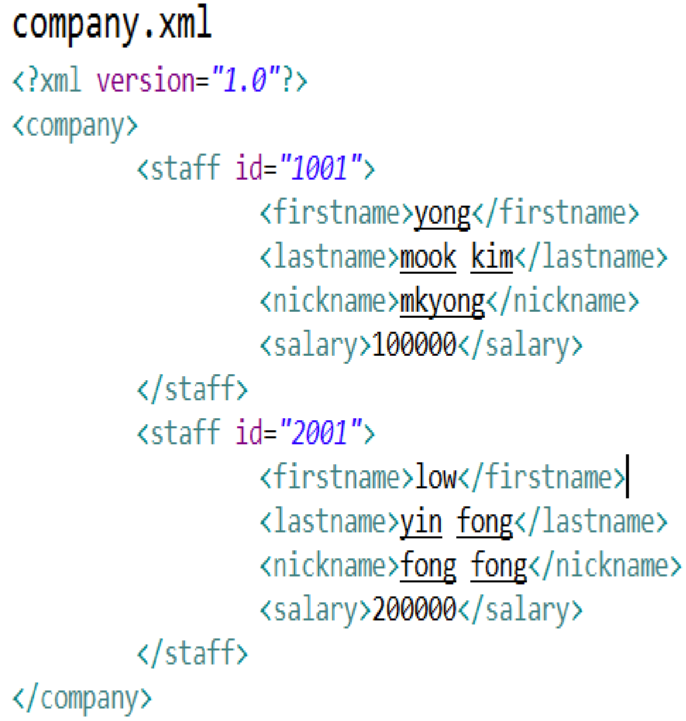
Reading XML file in using SAX

- **Following are SAX callback methods :**
  - **startDocument() and endDocument()**
    - Method called at the start and end of an XML document
  - **startElement() and endElement()**
    - Method called at the start and end of a document element
  - **characters()**
    - Method called with the text contents in between the start and end tags of an XML document element

Hello World SAX



```
public class EmployeeHandler extends DefaultHandler {

	@Override
	public void characters(char[] ch, int start, int length)throws SAXException {
		for(int i=start; i<(start+length);i++){
			System.out.print(ch[i]);
		}
	}

	@Override
	public void endDocument() throws SAXException {
		System.out.println("finishing processing xml doc");
	}

	@Override
	public void endElement(String uri, String localName, String qName)throws SAXException {
		System.out.print("</"+qName+ ">");

	}

	@Override
	public void startDocument() throws SAXException {
		System.out.println("Starting processing xml doc");
	}

	@Override
	public void startElement(String uri, String localName, String qName,Attributes attributes)
	throws SAXException {
		System.out.print("<"+qName+ ">");
	}
}
```



company.xml

```
<?xml version="1.0"?>
<company>
	<staff id="1001">
		<firstname>yong</firstname>
		<lastname>mook kim</lastname>
		<nickname>mkyong</nickname>
		<salary>100000</salary>
	</staff>
	<staff id="2001">
		<firstname>low</firstname>
		<lastname>yin fong</lastname>
		<nickname>fong fong</nickname>
		<salary>200000</salary>
	</staff>
</company>
```

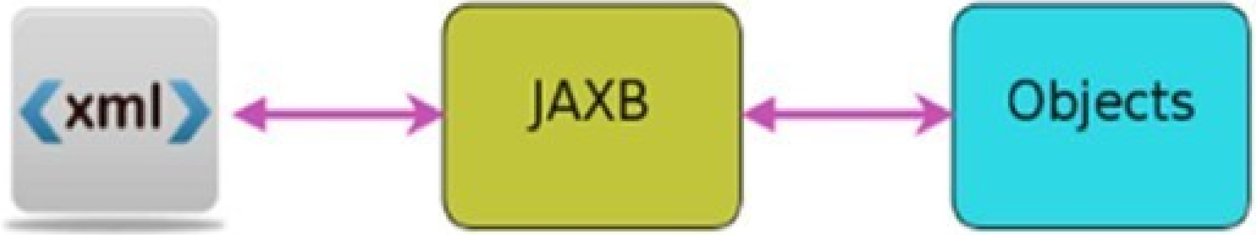
```
XMLReader reader=XMLReaderFactory.createXMLReader();
reader.setContentHandler(new EmployeeHandler());
reader.parse("company.xml");
```

What is JAXB ?

- **JAXB is an acronym of Java Architecture for XML Binding.**

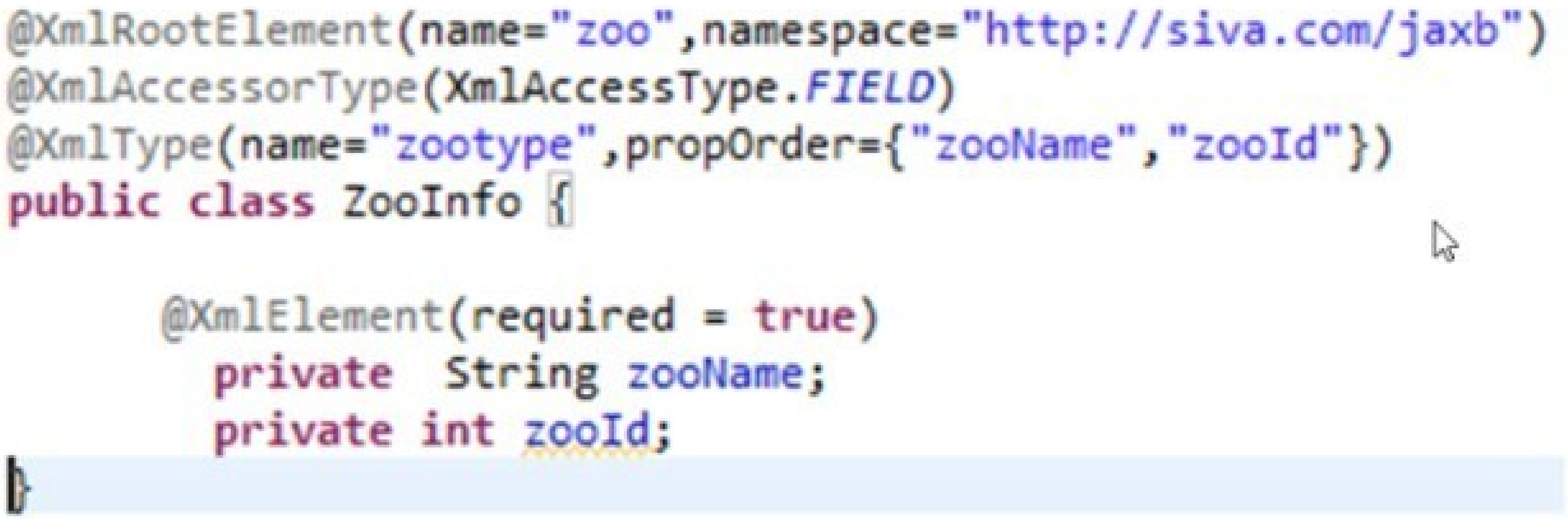
- **JAXB provides API to access and process a XML document.**

- **We can read or write XML files using JAXB.**



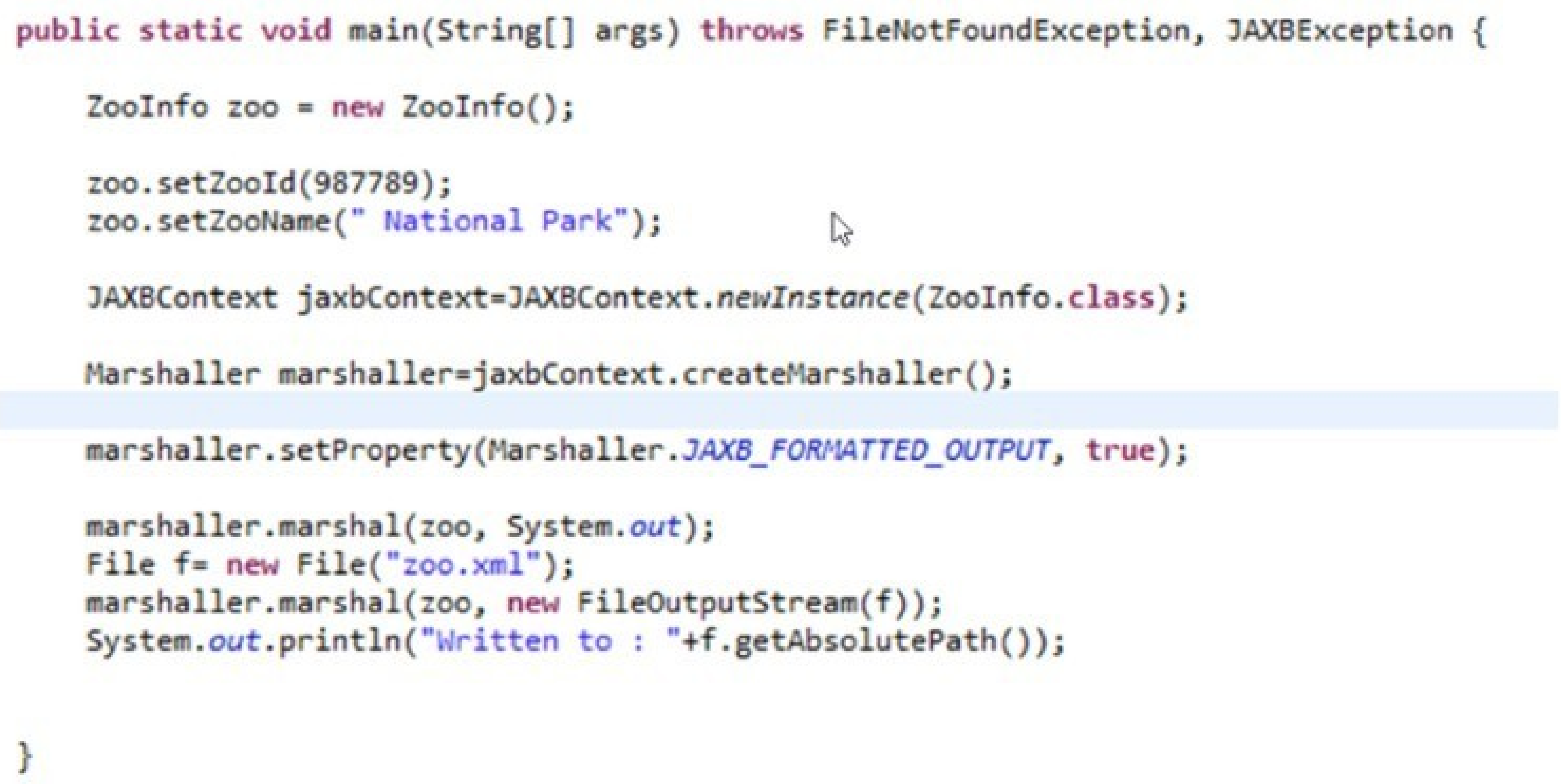
- **Unlike SAX, DOM parsers, to use JAXB the developer need not be aware of XML parsing techniques.**

Using Annotations

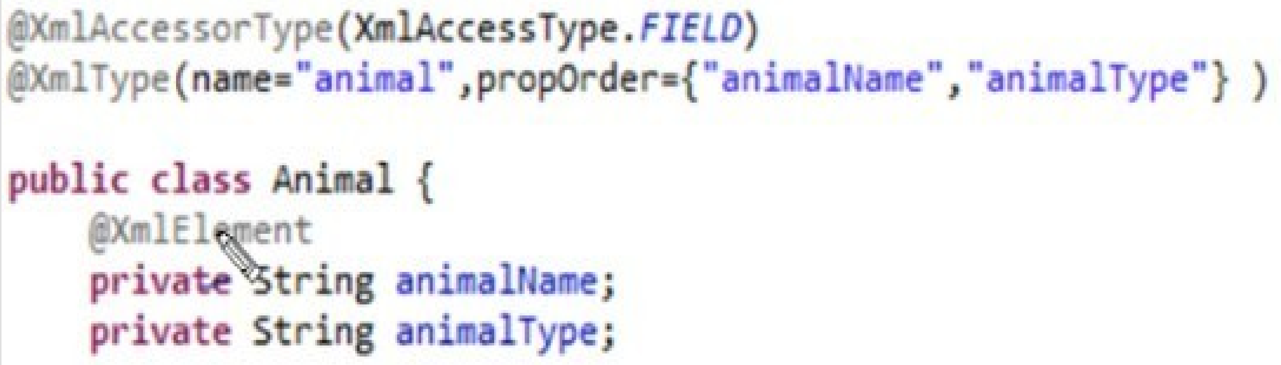


```
@XmlRootElement(name="zoo",namespace="http://siva.com/jaxb")
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name="zootype",propOrder={"zooName","zooId"})
public class ZooInfo {

     @XmlElement(required = true)
      private  String zooName;
      private int zooId;
}
```

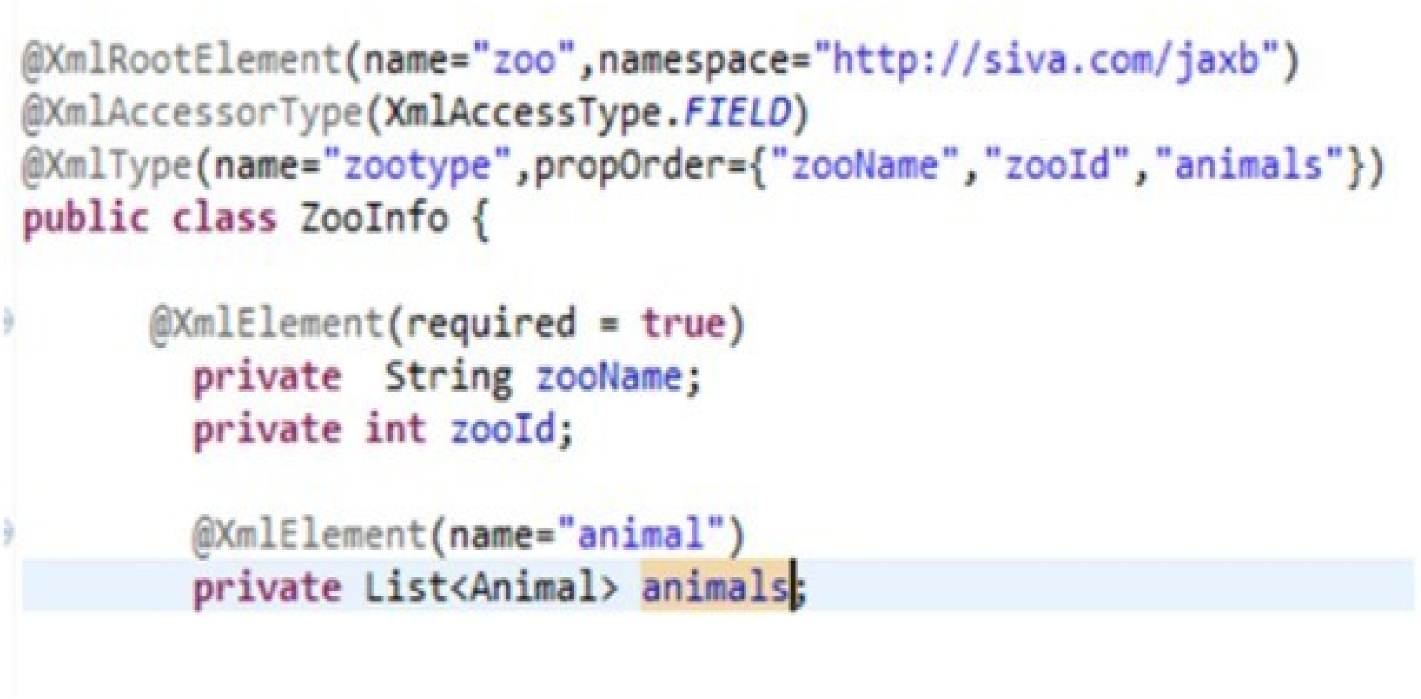
Marshalling Example



```
public static void main(String[] args) throws FileNotFoundException, JAXBException {

    ZooInfo zoo = new ZooInfo();

    zoo.setZooId(987789);
    zoo.setZooName(" National Park");

    JAXBContext jaxbContext=JAXBContext.newInstance(ZooInfo.class);

    Marshaller marshaller=jaxbContext.createMarshaller();

    marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);

    marshaller.marshal(zoo, System.out);
    File f= new File("zoo.xml");
    marshaller.marshal(zoo, new FileOutputStream(f));
    System.out.println("Written to : "+f.getAbsolutePath());


}
```

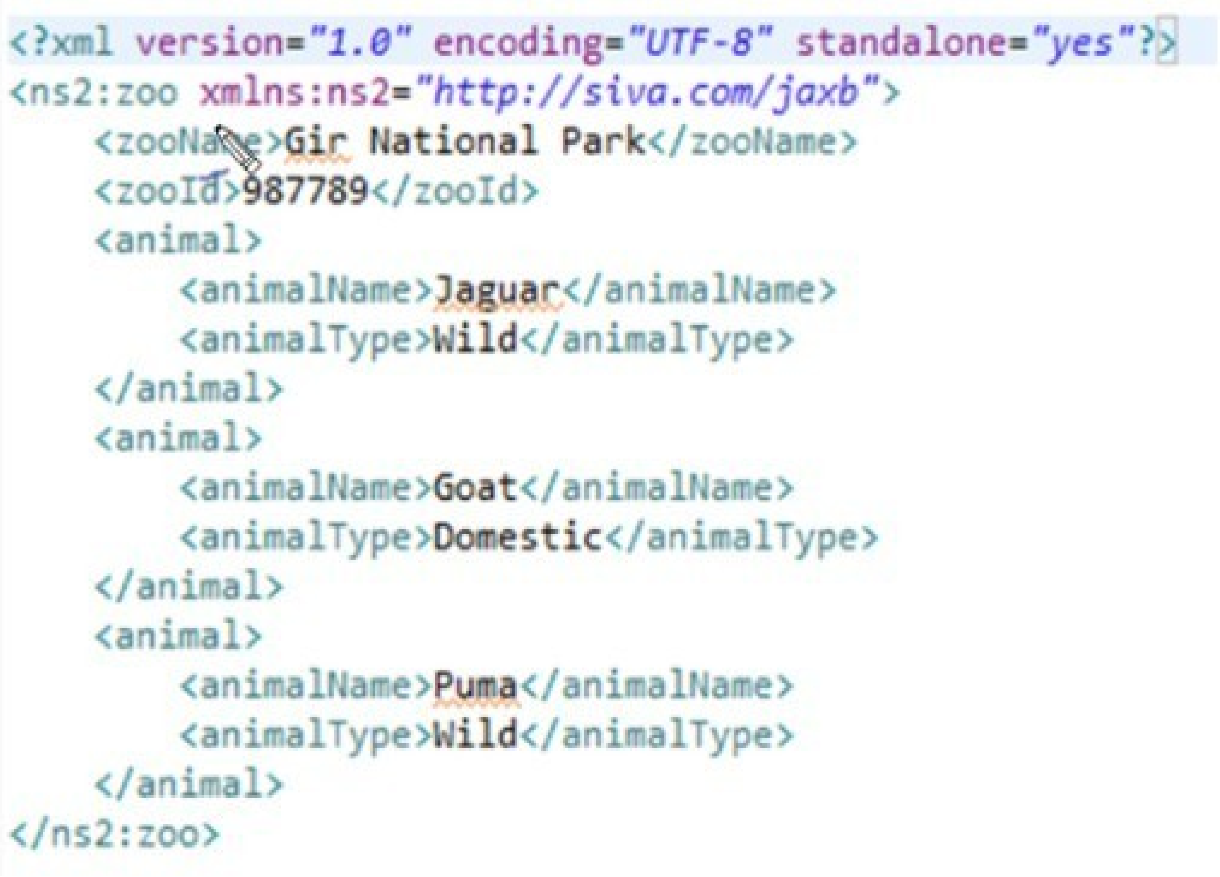


```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name="animal",propOrder={"animalName","animalType"} )

public class Animal {
    @XmlElement
    private String animalName;
    private String animalType;
```



```
@XmlRootElement(name="zoo",namespace="http://siva.com/jaxb")
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name="zootype",propOrder={"zooName","zooId","animals"})
public class ZooInfo {

      @XmlElement(required = true)
        private  String zooName;
        private int zooId;

        @XmlElement(name="animal")
        private List<Animal> animals;
```

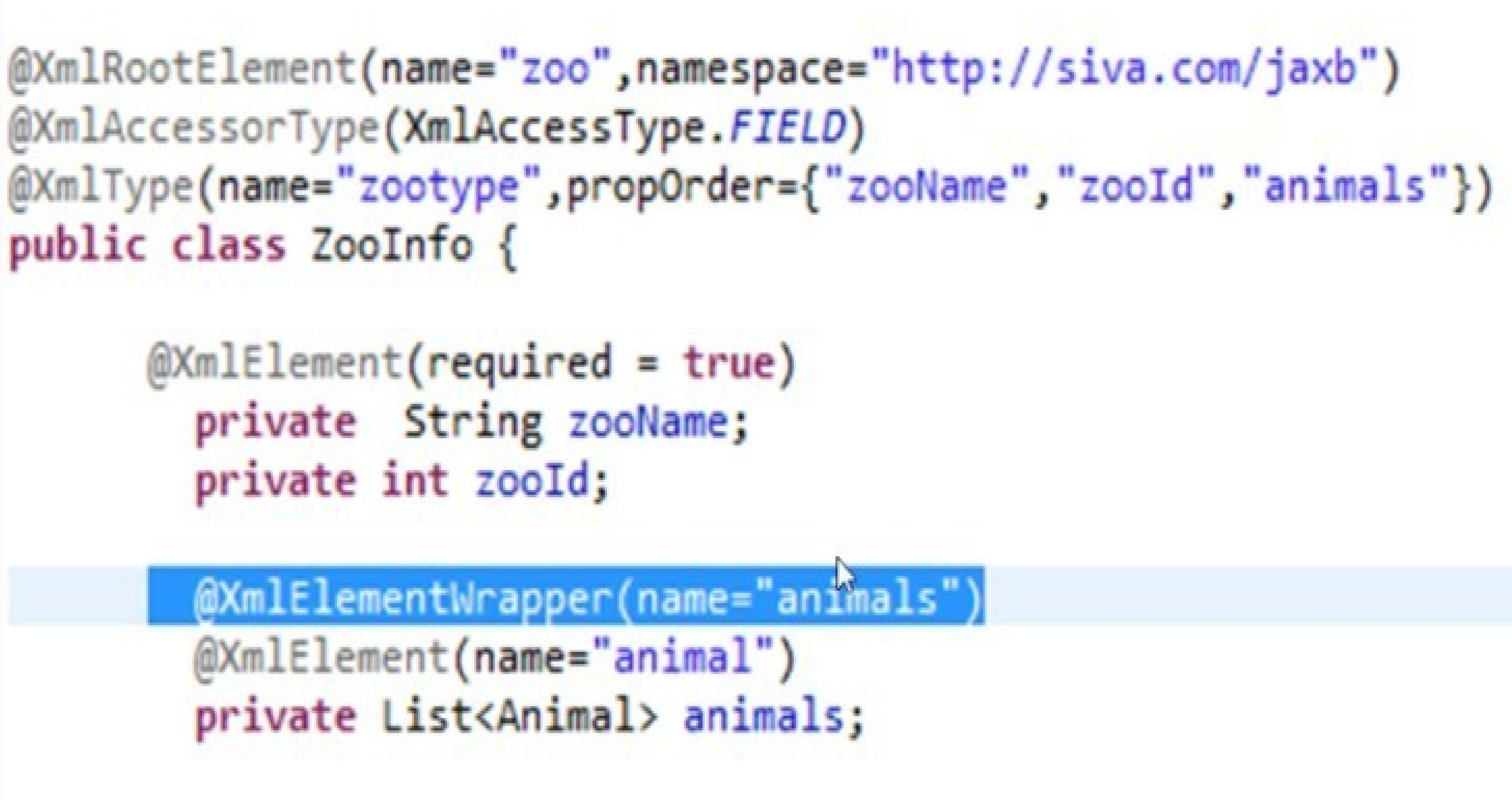
Output after marshalling



```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:zoo xmlns:ns2="http://siva.com/jaxb">
    <zooName>Gir National Park</zooName>
    <zooId>987789</zooId>
    <animal>
        <animalName>Jaguar</animalName>
        <animalType>Wild</animalType>
    </animal>
    <animal>
        <animalName>Goat</animalName>
        <animalType>Domestic</animalType>
    </animal>
    <animal>
        <animalName>Puma</animalName>
        <animalType>Wild</animalType>
    </animal>
</ns2:zoo>
```

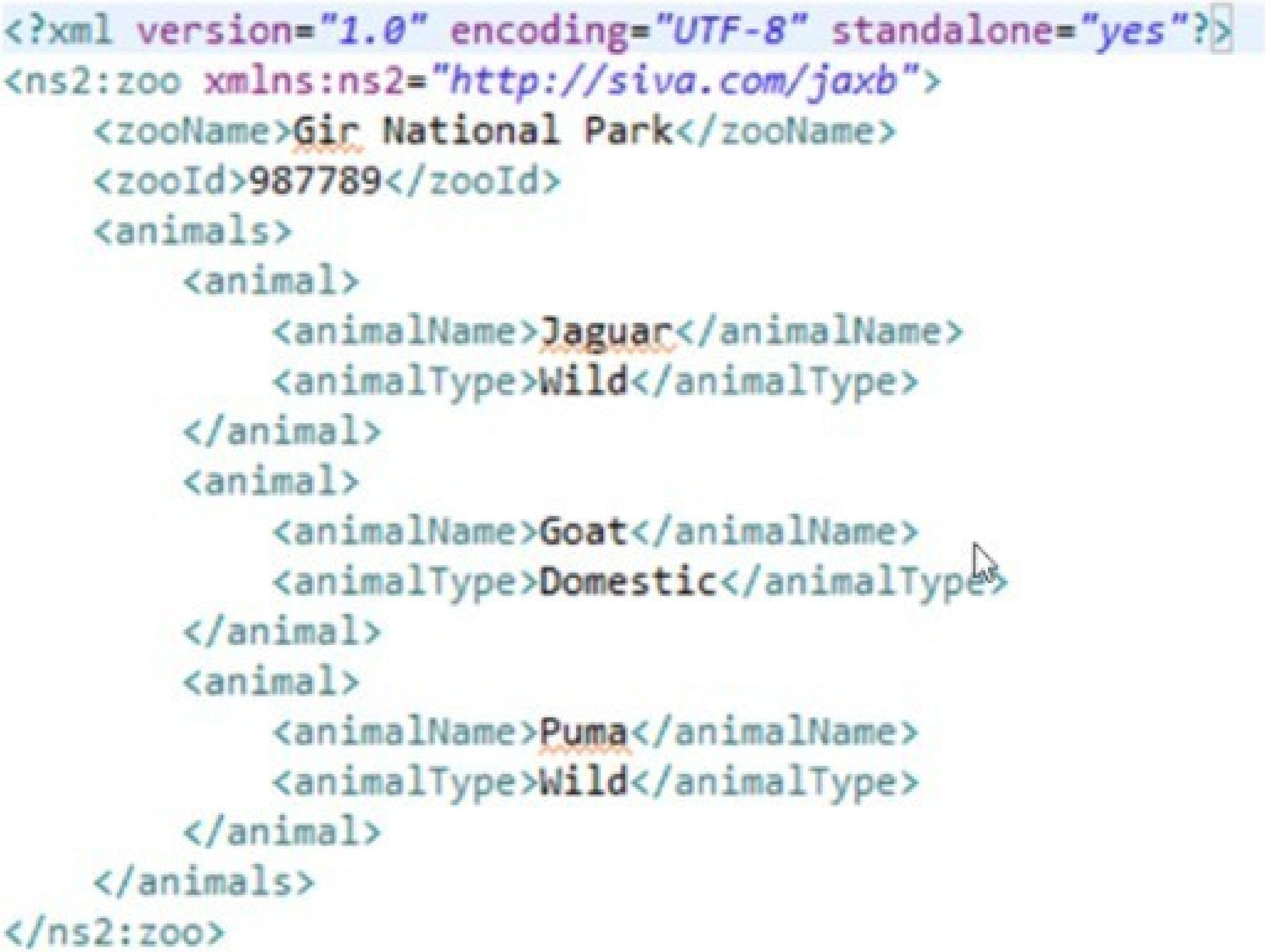
- It would look better if all animal tags are wrapped inside a tag like <animals>

Using @XmlElementWrapper



```
@XmlRootElement(name="zoo",namespace="http://siva.com/jaxb")
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name="zootype",propOrder={"zooName","zooId","animals"})
public class ZooInfo {

    @XmlElement(required = true)
      private  String zooName;
      private int zooId;

      @XmlElementWrapper(name="animals")
      @XmlElement(name="animal")
      private List<Animal> animals;
```

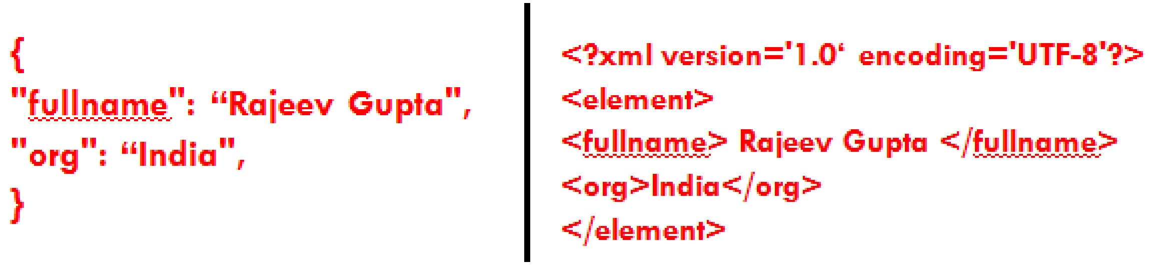
Output after using @XmlElementWrapper



```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:zoo xmlns:ns2="http://siva.com/jaxb">
    <zooName>Gir National Park</zooName>
    <zooId>987789</zooId>
    <animals>
        <animal>
            <animalName>Jaguar</animalName>
            <animalType>Wild</animalType>
        </animal>
        <animal>
            <animalName>Goat</animalName>
            <animalType>Domestic</animalType>
        </animal>
        <animal>
            <animalName>Puma</animalName>
            <animalType>Wild</animalType>
        </animal>
    </animals>
</ns2:zoo>
```

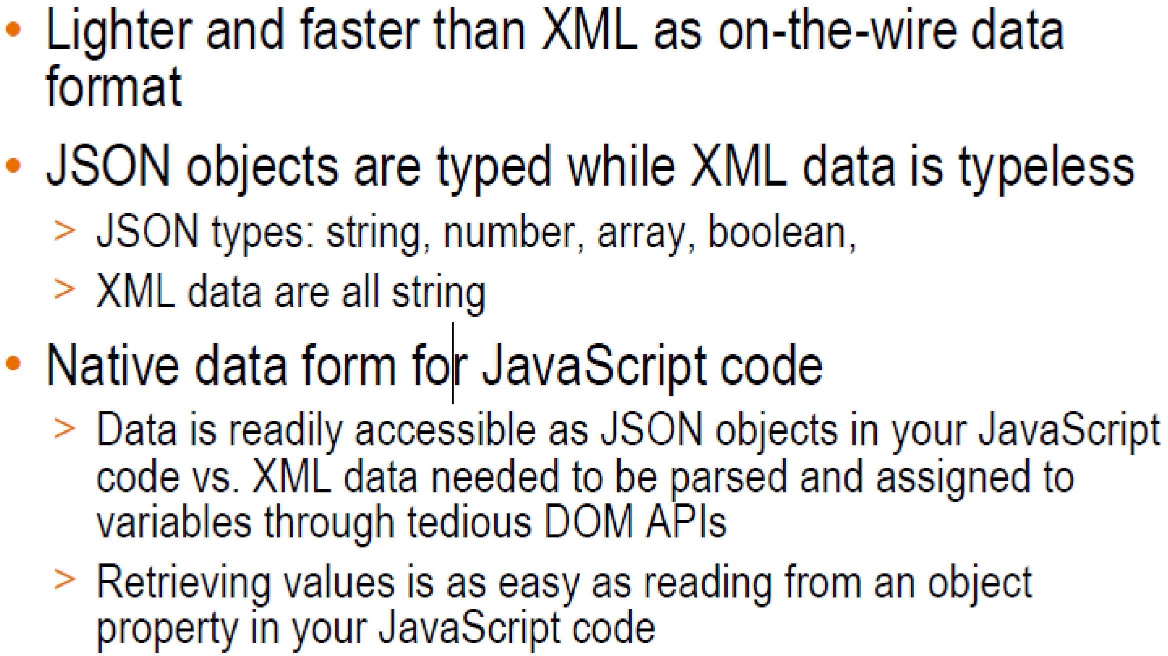
Why JSON is used in AJAX (optional topic)

□ JSON is widely used in AJAX, WS ?



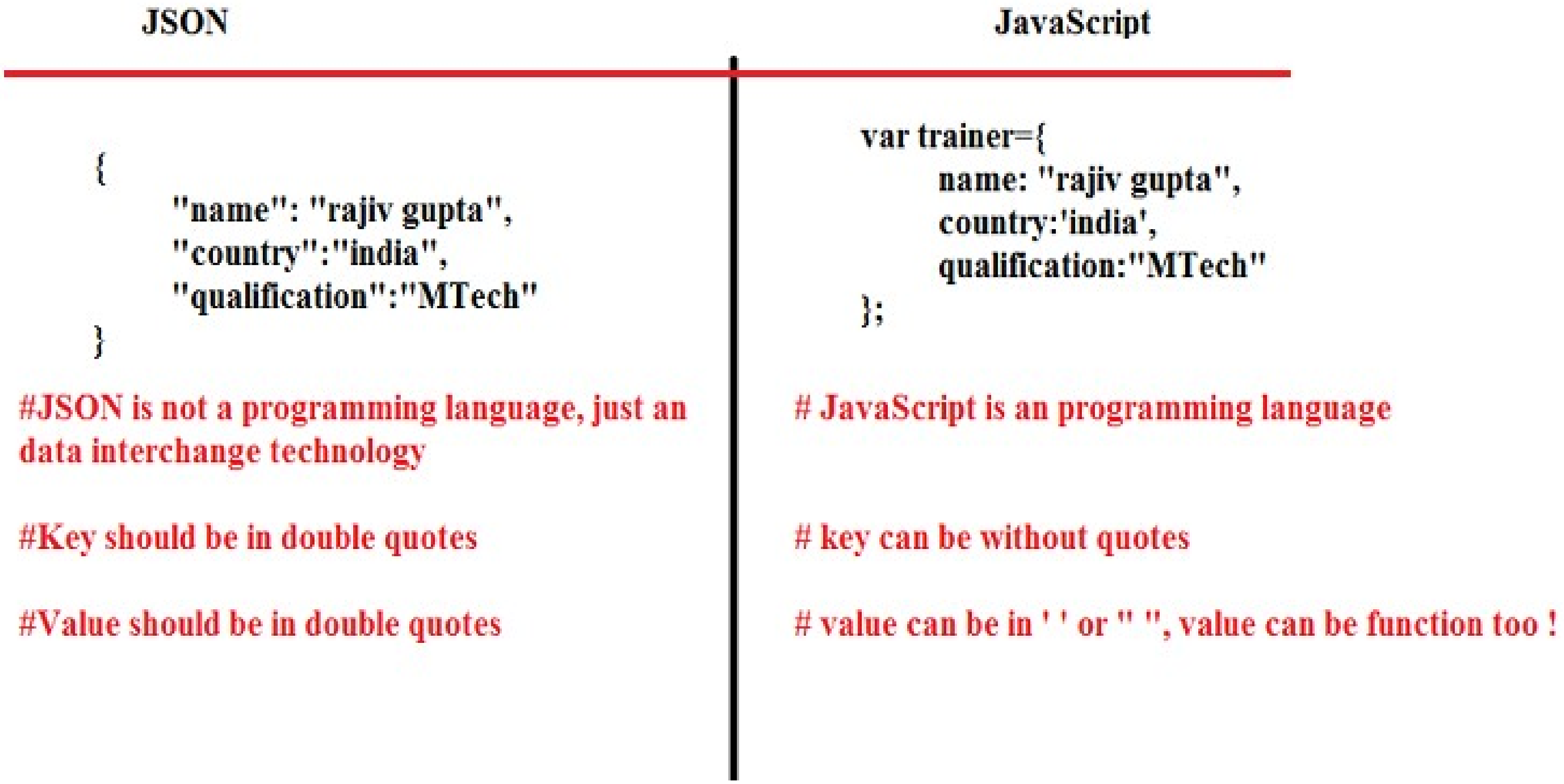
```
{
"fullname": “Rajeev Gupta",
"org": “India",
}
```

```
<?xml version='1.0‘ encoding='UTF-8'?>
<element>
<fullname> Rajeev Gupta </fullname>
<org>India</org>
</element>
```



- Lighter and faster than XML as on-the-wire data format
- JSON objects are typed while XML data is typeless
  - > JSON types: string, number, array, boolean,
  - > XML data are all string
- Native data form for JavaScript code
  - > Data is readily accessible as JSON objects in your JavaScript code vs. XML data needed to be parsed and assigned to variables through tedious DOM APIs
  - > Retrieving values is as easy as reading from an object property in your JavaScript code

JSON vs. JavaScript



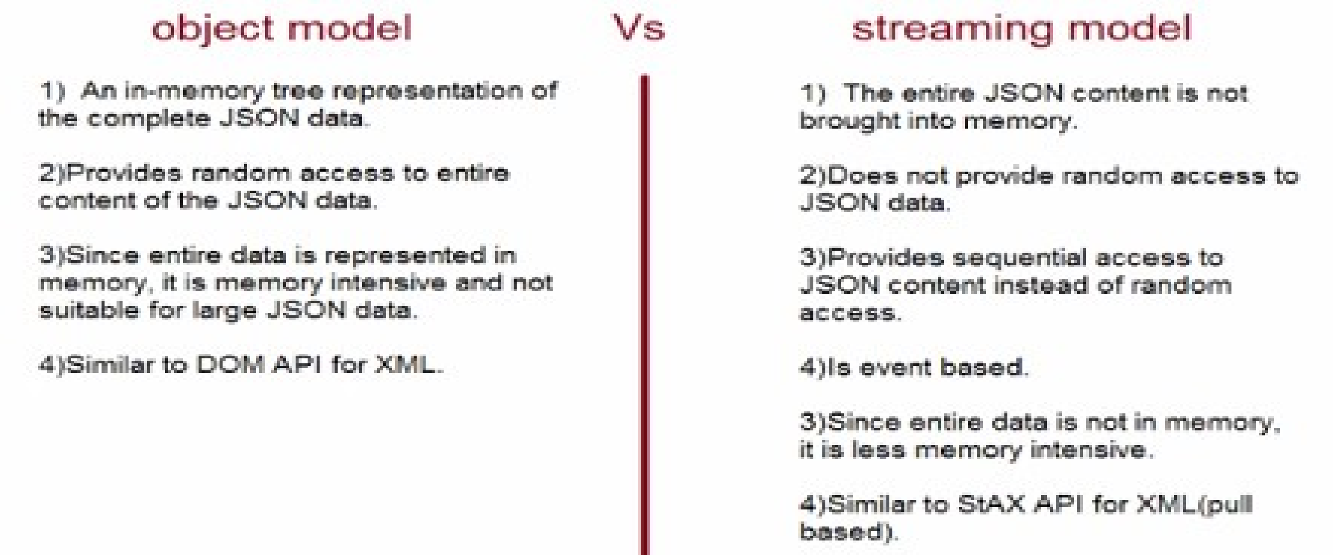
JSON	JavaScript

{
        "name": "rajiv gupta",
        "country":"india",
        "qualification":"MTech"
}
	var trainer={
	        name: "rajiv gupta",
	        country:'india',
	        qualification:"MTech"
	};

#JSON is not a programming language, just an data interchange technology	# JavaScript is an programming language
#Key should be in double quotes	# key can be without quotes
#Value should be in double quotes	# value can be in ' ' or " ", value can be function too !

JSON processing

- **JSON API provides two ways for JSON processing:**
  - Object Model API
    - It’s similar to DOM Parser and good for small objects.
  - Streaming API
    - It’s similar to StaX Parser and good for large objects where you don’t want to keep whole object in memory.

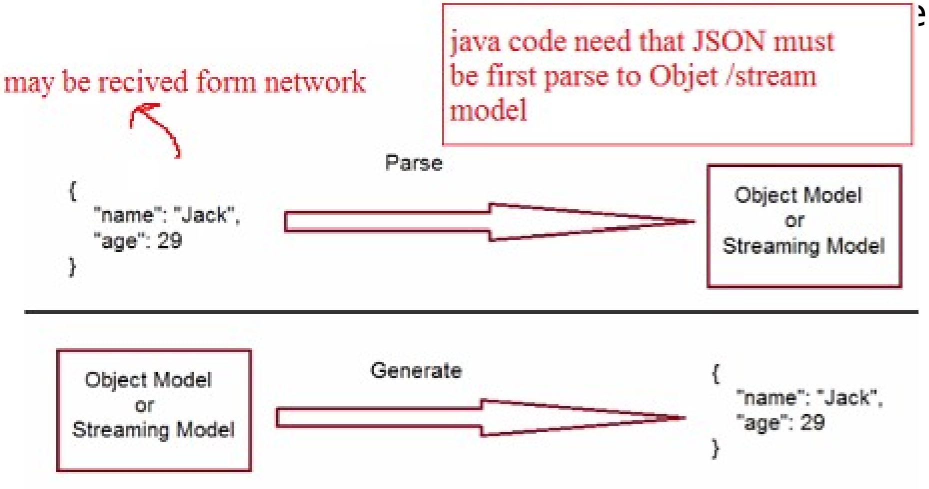


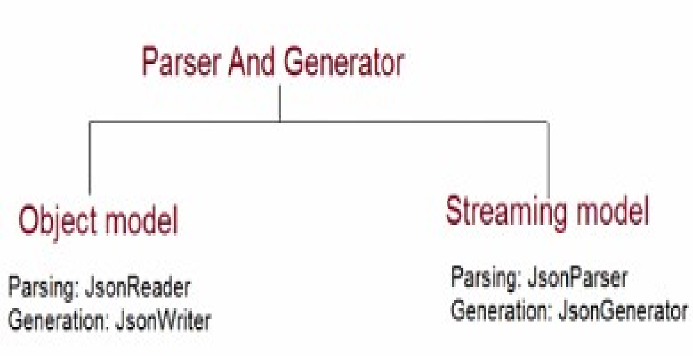
object model	Vs	streaming model
1) An in-memory tree representation of the complete JSON data.		1) The entire JSON content is not brought into memory.
2)Provides random access to entire content of the JSON data.		2)Does not provide random access to JSON data.
3)Since entire data is represented in memory, it is memory intensive and not suitable for large JSON data.		3)Provides sequential access to JSON content instead of random access.
4)Similar to DOM API for XML.		4)Is event based.
		3)Since entire data is not in memory, it is less memory intensive.
		4)Similar to StAX API for XML(pull based).

JSON API (Important Interface)

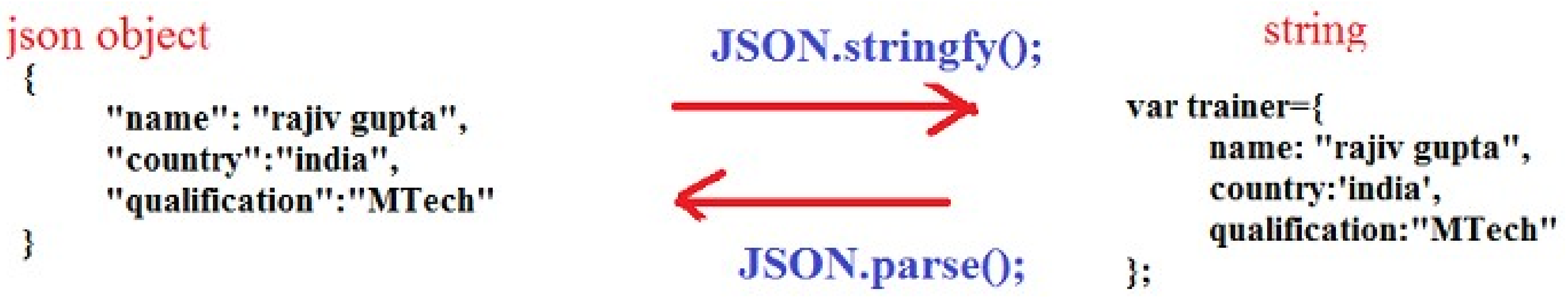
- **javax.json.JsonReader**: We can use this to read JSON object or an array to JsonObject. We can get JsonReader from Json class or JsonReaderFactory.
- **javax.json.JsonWriter**: We can use this to write JSON object to output stream.
- **javax.json.stream.JsonParser**: This works as a pull parser and provide streaming support for reading JSON objects.
- **javax.json.stream.JsonGenerator**: We can use this to write JSON object to output source in streaming way.
- **javax.json.Json**: This is the factory class for creating JSON processing objects. This class provides the most commonly used methods for creating these objects and their corresponding factories. The factory classes provide all the various ways to create these objects.

esents an immutable JSON object value


may be recived form network
java code need that JSON must be first parse to Objet /stream model
Parse
{
"name": "Jack",
"age": 29
}
Object Model or Streaming Model
Object Model or Streaming Model
Generate
{
"name": "Jack",
"age": 29
}



JSON to JavaScript notation interchange



json object

```
{
        "name": "rajiv gupta",
        "country":"india",
        "qualification":"MTech"
}
```

JSON.stringfy();

JSON.parse();

string

```
var trainer={
        name: "rajiv gupta",
        country:'india',
        qualification:"MTech"
};
```