XML Processing

Exporting and Importing Data from XML Format

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Questions

sli.do

#Java-DB

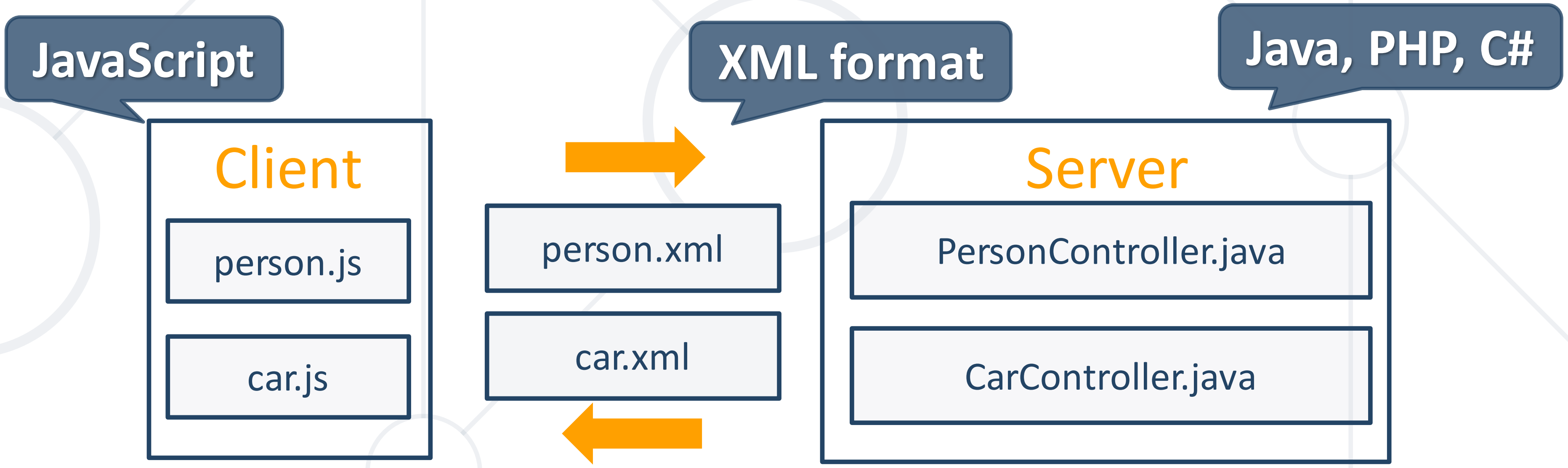
Table of Contents

XML Processing

Exporting and Importing Data from XML Format

XML Specifics

- EXtensible Mark-up Language
  - Lightweight format that is used for **data interchanging**
  - XML is language independent
- Primarily used to transmit data between a server and web application



XML Markup and Content

- An XML document consists of strings that:
  - Constitute **markup** – usually begin with **<** and end with **>**
  - Are **content** – placed between markup(**tags**)

person.xml

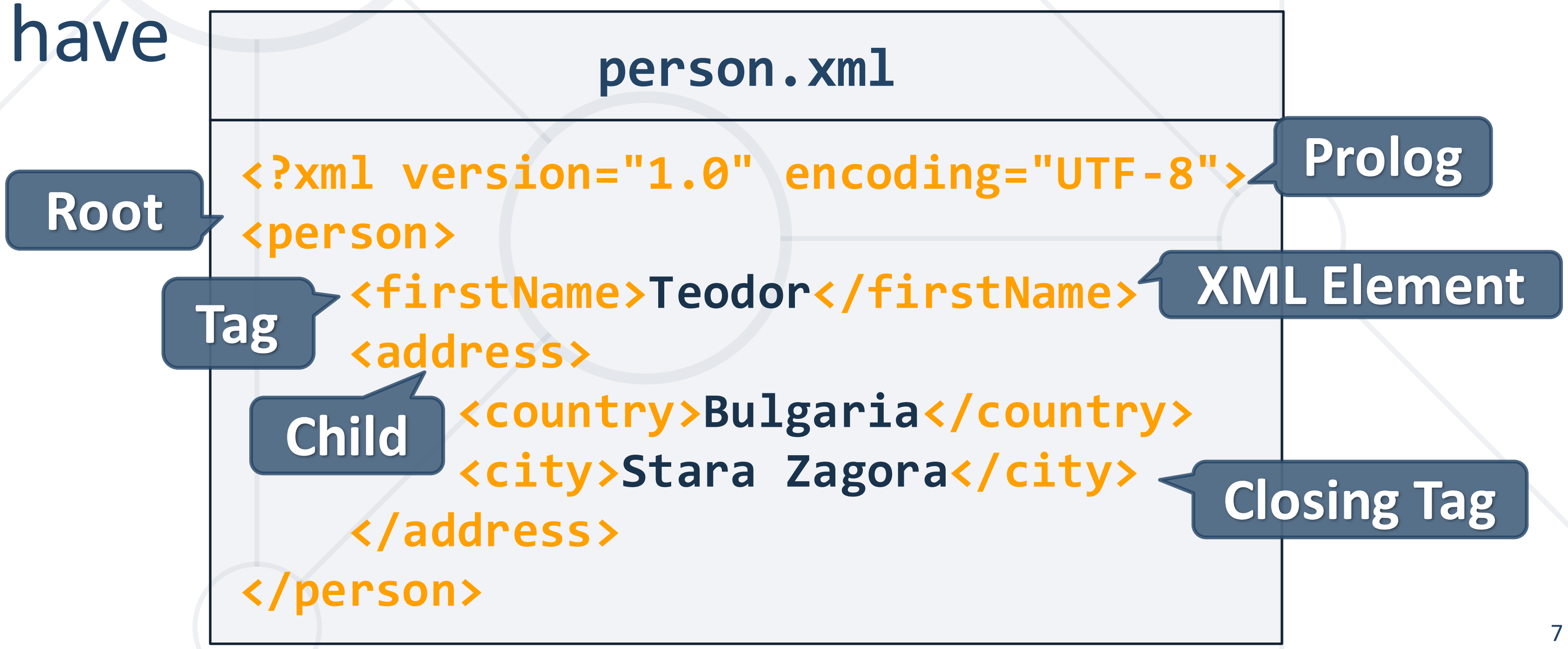
```
<?xml version="1.0" encoding="UTF-8">
<person>
    <firstName>Teodor</firstName>
</person>
```

Markup tags for Person Object

Content (Person Name)

XML Structure

- XML documents are formed as **element trees**
- An XML tree starts at a **root element** and branches from the root to **sub elements**
  - All elements can have child elements:



person.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<person>
    <firstName>Teodor</firstName>
    <address>
        <country>Bulgaria</country>
        <city>Stara Zagora</city>
    </address>
</person>
```

XML Structure

person.xml

```
<?xml version="1.0" encoding="UTF-8">
<person>
     <phoneNumbers>
        <phoneNumber>
            <number>08983248798</number>
        </phoneNumber>
        <phoneNumber>
            <number>08983243143</number>
        </phoneNumber>
    </phoneNumbers>
</person>
```

Wrapper

JAXB

Parsing XML to Java Objects

JAXB

- Processes the schema of the XML **document into a set of Java classes** that represent it
- Generates compact and readable XML output

pom.xml

```
<dependency>
    <groupId>javax.xml.bind</groupId>
    <artifactId>jaxb-api</artifactId>
</dependency>
```

JAXB Basics

- **Marshalling** - converting a Java Object to XML
- **Unmarshalling** - converting XML to Java Object
- We need to annotate the Java Object to provide instructions for XML creation:

AddressDto.java

```java
@XmlRootElement(name = "address")
@XmlAccessorType(XmlAccessType.FIELD)
public class AddressDto implements Serializable {
    @XmlAttribute(name = "country")
    private String country;

    @XmlElement(name = "city")
    private String city;
}
```

JAXB Annotations

- `@XmlRootElement`
  - Defines XML root object
- `@XmlAccessorType`
  - XmlAccessType.**FIELD**
  - XmlAccessType.**PROPERTY**
  - XmlAccessType.**PUBLIC_MEMBER**
- `@XmlAttribute`
  - Marks the field as an attribute to the object

JAXB Annotations

- `@XmlElement`
  - Marks the field as an element
- `@XmlElementWrapper(name = "…")`
  - Wraps the array of objects
- `@XmlTransient`
  - The field won't be exported/imported

JAXB Initialization

- **JAXBContext** objects are responsible for the XML manipulations
- JAXBContext.newInstance(object.getClass()) - creates an **instance** of JAXBContext
- **object.getClass** is the class that we will export/import
  - E.g. User, Address, Employee…

XMLParser.java

```java
this.jaxbContext = JAXBContext.newInstance(object.getClass());
```

Export Single Object to XML – Example 1

User.java

```
@XmlRootElement
@XmlAccessorType(XmlAccessType.FIELD)
public class User {
    @XmlElement(name = "name")
    private String name;
    @XmlElement(name = "age")
    private Integer age;
    public String getName() { return name; }
// Constructor, getters, setters
```

users.xml

```
<?xml version="1.0" encoding
="UTF-8" standalone="yes"?>
<user>
    <name>New User</name>
    <age>18</age>
</user>
```

XMLParser.java

```
JAXBContext context = JAXBContext.newInstance(User.class);
Marshaller marshaller = context.createMarshaller();
marshaller.marshal(user, new File("users.xml"));
```

Creates XML file "users.xml"

Export Single Object to XML – Example 2

AddressDto.java

```
@XmlRootElement(name = "address")
@XmlAccessorType(XmlAccessType.FIELD)
public class AddressDto implements Serializable {

    @XmlAttribute(name = "country")
    private String country;

    @XmlElement(name = "city")
    private String city;
}
```

Object attribute

address.xml

```
<?xml version="1.0"
encoding="UTF-8"?>
<address country="Bulgaria">
    <city>Sofia</city>
</address>
```

XMLParser.java

```
Marshaller jaxbMarshaller = jaxbContext.createMarshaller();
jaxbMarshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
OutputStream outputStream = new FileOutputStream(fileName);
BufferedWriter bfw =
    new BufferedWriter(new OutputStreamWriter(outputStream));
jaxbMarshaller.marshal(object, bfw);
```

Format XML output (Analogically to setPrettyPrinting in JSON parsing)

Export Single Object to XML

AddressDto.java

```java
@XmlRootElement(name = "address")
@XmlAccessorType(XmlAccessType.FIELD)
public class AddressJsonDto
            implements Serializable {
    @XmlAttribute(name = "country")
    private String country;

    @XmlElement(name = "city")
    private String city; }
```

address.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<address country="Bulgaria">
    <city>Sofia</city>
</address>
```

Export Multiple Objects to XML

AddressesDto.java

```java
@XmlRootElement(name = "addresses")
@XmlAccessorType(XmlAccessType.FIELD)
public class AddressesDto {

    @XmlElement(name = "address")
    private List<AddressDto> addressJsonDtos;
}
```

XMLParser.java

```java
AddressesDto addressDtos = new AddressesDto();
jaxbMarshaller.marshal(addressesDto, bfw);
```

Export Multiple Objects to XML

XMLParser.java

```java
AddressesDto addressDtos = new AddressesDto();
jaxbMarshaller.marshal(addressesDto, bfw);
```

addresses.json

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<addresses>
    <address country="Bulgaria">
        <city>Sofia</city>
    </address>
    <address country="Spain">
        <city>Barcelona</city>
    </address>
</addresses>
```

Import Single Object from XML

AddressDto.java

```
@XmlRootElement(name = "address")
@XmlAccessorType(XmlAccessType.FIELD)
public class AddressDto implements Serializable {

    @XmlAttribute(name = "country")
    private String country;

    @XmlElement(name = "city")
    private String city;
}
```

XMLParser.java

```
JAXBContext jaxbContext = JAXBContext.newInstance(AddressDto.class);
InputStream inputStream = getClass().getResourceAsStream("/files/input/xml/
address.xml");
BufferedReader bfr = new BufferedReader(new InputStreamReader(inputStream));
Unmarshaller unmarshaller = jaxbContext.createUnmarshaller();
AddressDto addressDto = (AddressDto) unmarshaller.unmarshal(bfr);
```

Creates Object

Import Single Object from XML

AddressDto.java

```java
@XmlRootElement(name = "address")
@XmlAccessorType(XmlAccessType.FIELD)
public class AddressDto implements
Serializable {

    @XmlAttribute(name = "country")
    private String country;

    @XmlElement(name = "city")
    private String city;
}
```

address.xml

```xml
<?xml version="1.0" encoding=
"UTF-8" standalone="yes"?>
<address country="Bulgaria">
    <city>Sofia</city>
</address>
```

Import Multiple Objects to XML

XMLParser.java

```java
JAXBContext jaxbContext = JAXBContext.newInstance(AddressesDto.class);
InputStream inputStream = getClass().getResourceAsStream("/files/input/xml/addresses.xml");
BufferedReader bfr = new BufferedReader(new InputStreamReader(inputStream));
Unmarshaller unmarshaller = jaxbContext.createUnmarshaller();
AddressesDto addressesDto = (AddressesDto) unmarshaller.unmarshal(bfr);
```

addresses.xml

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<addresses>
    <address country="Bulgaria">
        <city>Sofia</city>
    </address>
    <address country="Spain">
        <city>Barcelona</city>
    </address>
</addresses>
```

Summary

- XML is another way to transfer data besides JSON
- XML document's format consists of mark-up and content elements
- JAXB is a library which helps us to read XML files and parse them to Java objects

Software University

Questions?

SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

Finance Academy

SoftUni Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SOFTWARE GROUP

DXC TECHNOLOGY

createX

Postbank

Решения за твоето утре

SUPER HOSTING .BG

DRAFT KINGS

VIVACOM

INDEAVR

Serving the high achievers

AMBITIONED

PHAR VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License