XML Processing

Exporting and Importing Data from XML Format

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Table of Contents

Questions

Software University

sli.do

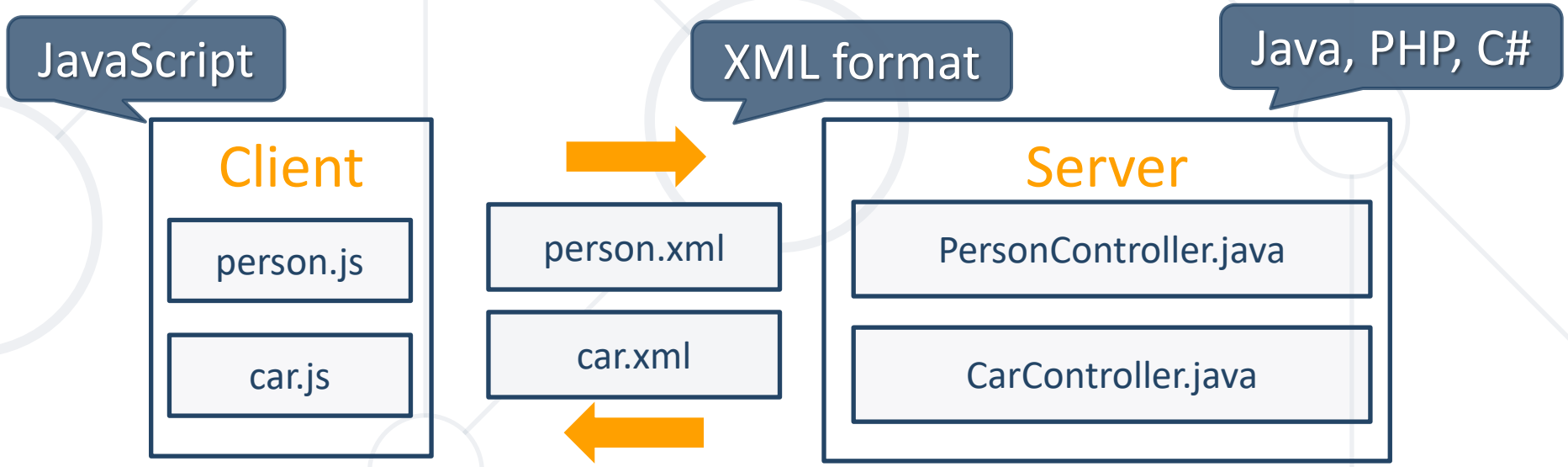
#Java-DB

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

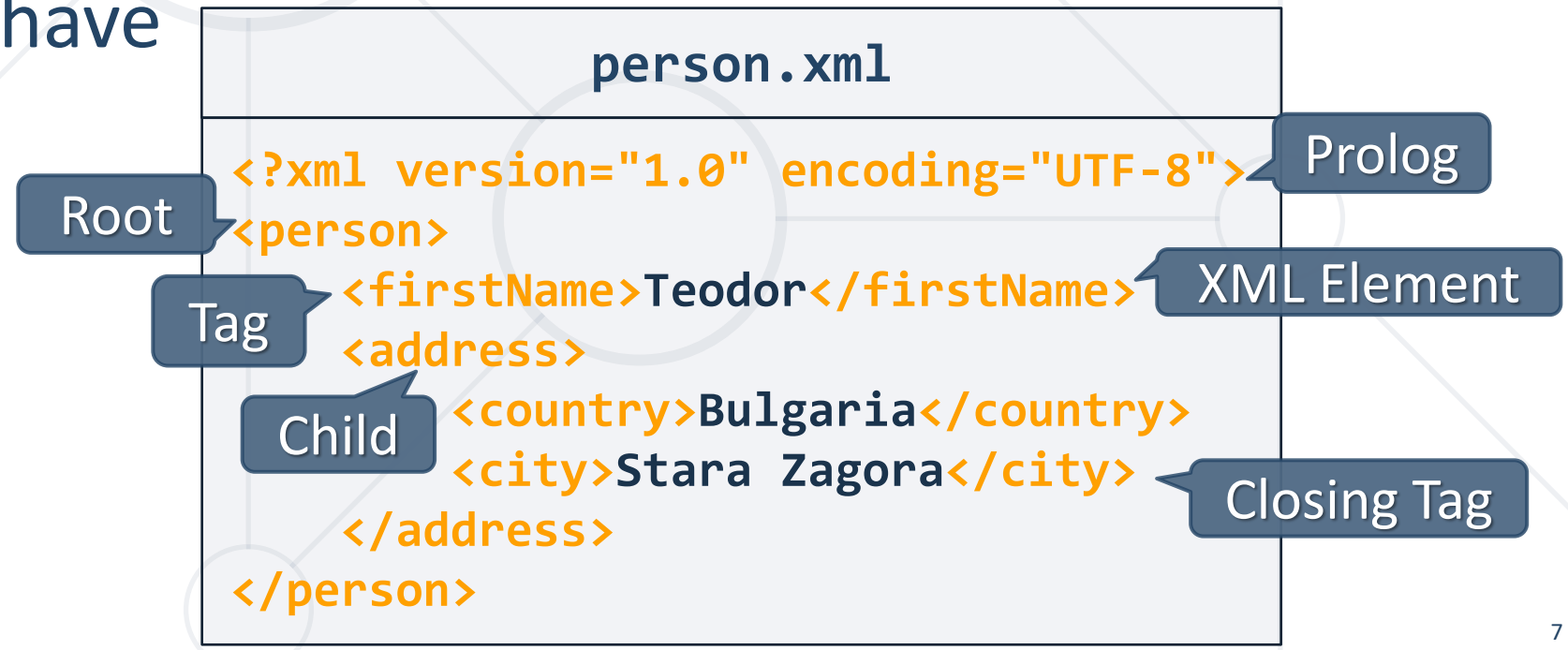
```
<?xml version="1.0" encoding="UTF-8">
<person>
    <firstName>Teodor</firstName>
</person>
```

Markup tags for Person Object

Content (Person Name)

XML Structure (1)

- 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:



XML Structure (2)

Software University

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

```
@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 (1)

- **@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 (2)

- `@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

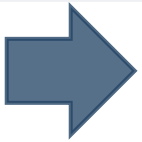
```
this.jaxbContext = JAXBContext.newInstance(object.getClass());
```

Export Single Object to XML – Example 1

Software University

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

Software University

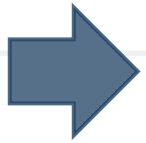
AddressDto.java

```
@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

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

Export Multiple Objects to XML (1)

Software University

AddressesDto.java

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

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

XMLParser.java

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

Export Multiple Objects to XML (2)

Software University

XMLParser.java

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

addresses.json

```
<?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 (1)

Software University

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 (2)

Software University

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;
}
```

address.xml

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

Import Multiple Objects to XML

Software University

XMLParser.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);
```

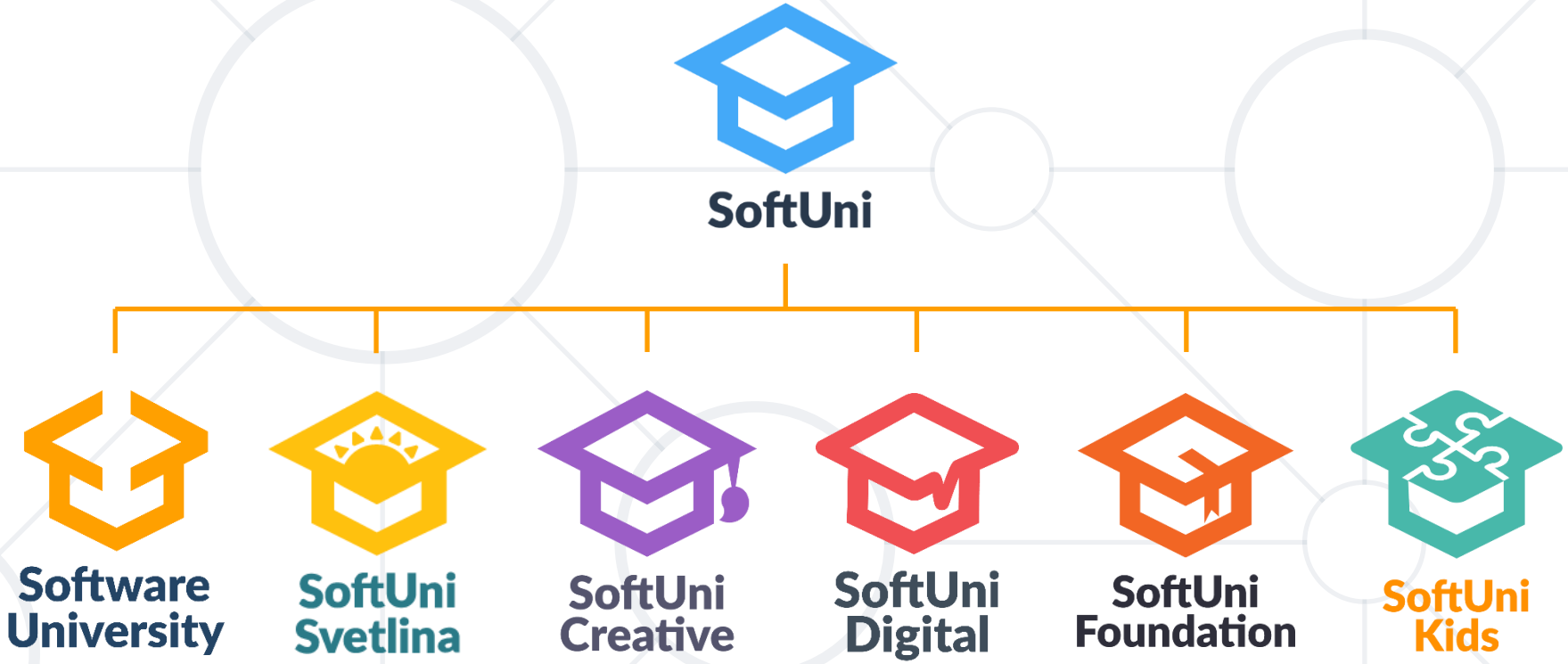
addresses.xml

```
<?xml 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

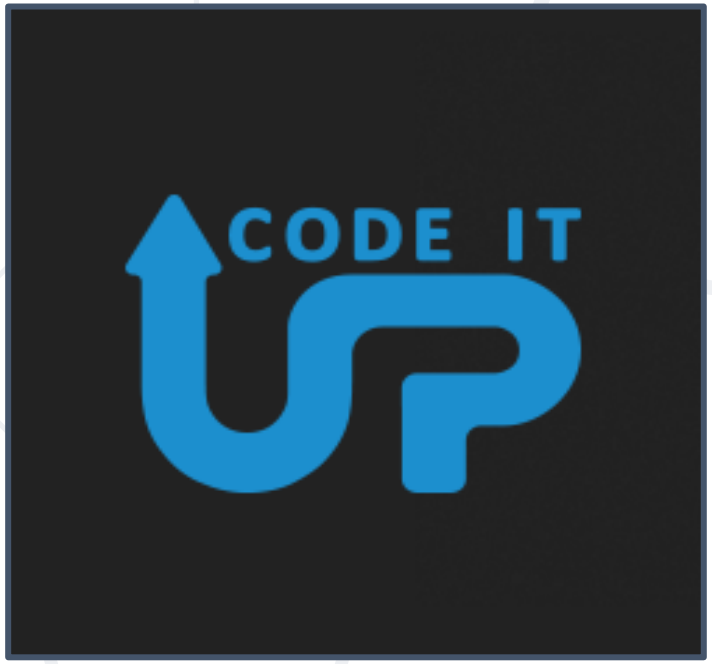
Questions?



SoftUni Diamond Partners

Software University

Educational Partners



VIRTUAL RACING SCHOOL

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
- Software University Forums
  - forum.softuni.bg

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg