Интернет технологии

Технологии за работа на Java приложения с XML.

JAXB.

Същност

- Персистиране на информация
  - Малък обем – настройки, конфигурации, регистри
- Преминаване от код (обект, модел) в xml структура
  - И обратно

Инструменти

- Java API for XML Processing (JAXP)
  - Парсър процес
- Java Architecture for XML Binding (JAXB)
  - Работа с DOM

- До версия 1.8 и след това!

Принцип на работа на JAXB

- Конвертиране от java обект в xml - marshaling.
- Конвертиране от xml в java обект – un-marshaling.

- Стъпки на процеса
  - Мапинг
  - Инициализация на jaxb marshaler / un-marshaler
  - Marshaling / un-marshaling

JAXB (Анотации)

Анотация	SCOPE	Описание
@XmlRootElement	Class, Enum	Определя XML коренния елемент. Основните Java класове трябва да бъдат регистрирани в контекста на JAXB, когато са създадени.
@XmlAccessorType	Package, Class	Определя полетата и свойствата на вашите Java класове, които JAXB използва за свързване. Той има четири стойности: PUBLIC_MEMBER, FIELD, PROPERTY и NONE
@XmlAccessorOrder	Package, Class	Определя последователния ред на дъщерните елементи
@XmlType	Class, Enum	Свързва Java клас с тип схема. Той определя името на типа и реда на дъщерните елементи.
@XmlElement	Field	Съпоставя поле или свойство с XML елемент
@XmlAttribute	Field	Съпоставя поле или свойство с XML елемент

JAXB (Анотации)

@XmlTransient	Field	Предотвратява съпоставянето на поле или свойство в XML схемата
@XmlValue	Field	Съпоставя поле или свойство с XML елемент
@XmlList	Field, Parameter	Съпоставя колекция със списък от стойности, разделени с интервал.
@XmlElementWrapper	Field	Свързва Java колекция с XML колекция

JAXB пример

public class Country {

Атрибути - "name", "capital", "foundation", "continent" , "population,,

Setters,getters

public void setPopulation(int population) {

this.population = population;

}

public void setName(String name) {

this.name = name;

}

...

JAXB пример

```
<?xml version="1.0" encoding="UTF-8"?>
<Country …>
<Country_Name>…</Country_Name>
<Country_Capital>….</Country_Capital>
<Country_Foundation_Date>…</Country_Foundation_Date>
<Country_Continent>…</Country_Continent>
<Country_Population>…</Country_Population>
</Country>
```

JAXB пример (Анотации)

```
@XmlType( propOrder = { "name", "capital", "foundation", "continent" , "population"} )
@XmlRootElement( name = "Country" )
public class Country {
@XmlElement (name = "Country_Population")
public void setPopulation( int population ) {
this.population = population;
}
@XmlElement( name = "Country_Name" )
public void setName( String name ) {
this.name = name;
}
@XmlElement( name = "Country_Capital" )
public void setCapital( String capital ) {
this.capital = capital;
}
@XmlAttribute( name = "importance", required = true )
public void setImportance( int importance ) {
this.importance = importance;
}
...
```

JAXB пример marshaling

```
Country spain = new Country();
spain.setName( "Spain" );
spain.setCapital( "Madrid" );
spain.setContinent( "Europe" );
spain.setImportance( 1 );
spain.setFoundation( LocalDate.of( 1469, 10, 19 ) );
spain.setPopulation( 45000000 );
JAXBContext jaxbContext = JAXBContext.newInstance( Country.class );
Marshaller jaxbMarshaller = jaxbContext.createMarshaller();
jaxbMarshaller.setProperty( Marshaller.JAXB_FORMATTED_OUTPUT, true );
jaxbMarshaller.marshal( spain, new File( "country.xml" ) );
//jaxbMarshaller.marshal( spain, System.out );
```

JAXB пример marshaling

```
<?xml version="1.0" encoding="UTF-8"?>
<Country importance="1">
<Country_Name>Spain</Country_Name>
<Country_Capital>Madrid</Country_Capital>
<Country_Foundation_Date></Country_Foundation_Date>
<Country_Continent>Europe</Country_Continent>
<Country_Population>45000000</Country_Population>
</Country>
```

Работа с колекции

- Не може директно списък – необходимост от контейнер
  - Отделен клас
- @XmlRootElement(name = "Countries")

```
public class Countries {
List countries;
@XmlElement( name = "Country" )
public void setCountries( List countries ) {
this.countries = countries;
}
```

Работа с колекции

```
Country spain = new Country();
…..
Countries countries = new Countries();
countries.add( spain );
countries.add( usa );

JAXBContext jaxbContext = JAXBContext.newInstance( Countries.class );
Marshaller jaxbMarshaller = jaxbContext.createMarshaller();
jaxbMarshaller.setProperty( Marshaller.JAXB_FORMATTED_OUTPUT, true );
jaxbMarshaller.marshal( countries, new File( "list_countries.xml" ) );
```

JAXB Un-marshal

```
File file = new File( "countries.xml" );
JAXBContext jaxbContext = JAXBContext.newInstance( Countries.class );
Unmarshaller jaxbUnmarshaller = jaxbContext.createUnmarshaller();
Countries countres = (Countries)jaxbUnmarshaller.unmarshal( file );
```

JAXB Адаптери

- При използване на сложни обекти
  - java.time.LocalDate.

JAXB Адаптери

```
public class DateAdapter extends XmlAdapter<String, LocalDate> {

public LocalDate unmarshal( String date ) throws Exception {
        return LocalDate.parse( date );
    }

public String marshal( LocalDate date ) throws Exception {
        return date.toString();
    }
}
```

JAXB Адаптери

- @XmlElement(name = "Country_Foundation_Date")
  @XmlJavaTypeAdapter(**DateAdapter**.class)
  public void setFoundation(LocalDate foundation)

- country.setFoundation(LocalDate.of(1469, 10, 19));

- <Country_Foundation_Date>1469-10-19</Country_Foundation_Date>

Валидиране на XML данните

- Валидиране на файл – XSD схема
- <?xml version="1.0" encoding="UTF-8"?>

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="Country">
        <xs:complexType>
                <xs:sequence>
                        <xs:element name="Country_Name" type="xs:string" />
                        <xs:element name="Country_Capital" type="xs:string" />
                        <xs:element name="Country_Foundation_Date" type="xs:string" />
                        <xs:element name="Country_Continent" type="xs:string" />
                        <xs:element name="Country_Population" type="xs:integer" />
                </xs:sequence>
        </xs:complexType>
</xs:element>
</xs:schema>
```

Валидиране на XML данните

- Валидиране на обект - – XSD схема
- <?xml version="1.0" encoding="UTF-8"?>

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:simpleType name="continentType">
                <xs:restriction base="xs:string">
                        <xs:pattern value="Asia|Europe|America|Africa|Oceania"/>
                </xs:restriction>
        </xs:simpleType>
<xs:element name="Country">
        <xs:complexType>
                <xs:sequence>
                        <xs:element name="Country_Name" type="xs:string" minOccurs='1' />
                        <xs:element name="Country_Capital" type="xs:string" minOccurs='1' / >
                        <xs:element name="Country_Foundation_Date" type="xs:string"  minOccurs='1' />
                        <xs:element name="Country_Continent" type="continentType" minOccurs  ='1' />
                        <xs:element name="Country_Population" type="xs:integer" />
                </xs:sequence>
        </xs:complexType>
</xs:element>
</xs:schema>
```

Валидиране на XML данните

- SchemaFactory sf = **SchemaFactory.newInstance**(XMLConstants.W3C_XML_SCHEMA_NS_URI);
  Schema schema = sf.**newSchema**(new File("countries_validation.xsd"));

Прехващане на грешки

- org.xml.sax.ErrorHandler
- public class MyErrorHandler implements ErrorHandler {
  @Override
  public void warning(SAXParseException exception) throws SAXException {
  throw exception;
  }
  ...

Валидиране на XML данните

```
JAXBContext jaxbContext = JAXBContext.newInstance( Country.class );
JAXBSource sourceSpain = new JAXBSource( jaxbContext, spain );
JAXBSource sourceAustralia = new JAXBSource( jaxbContext, australia );
Validator validator = schema.newValidator();
validator.setErrorHandler( new MyErrorHandler() );
        try {
                validator.validate( sourceSpain );
                System.out.println( "spain has no problems" );
        } catch( SAXException ex ) {
                ex.printStackTrace();
                System.out.println( "spain has problems" );
                …
```