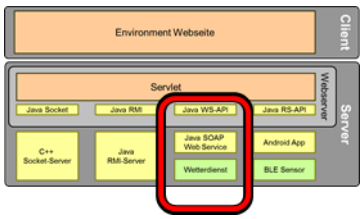
Verteilte Informationssysteme

Person:

Name:

Matrikelnummer:



Übungszettel 3

JAXB (XML und JSON)
Web Services (SOAP und REST)

Aufgabe 1a

„M“ **(15%)**

Erstellen Sie eine einfache Klasse **`Pet.java`**, die die folgenden internen Felder enthält:

- **`String mName;`**
- **`String mNickName;`**
- **`Date mBirthday;`**
- **`Type mTyp; // enum cat, dog, mouse und bird`**
- **`String[] mVaccinations;`**
- **`String mID;`**

Erstellen Sie auf Basis dieser Klasse ein Objekt mit folgenden Inhalten:

- Thomas
- Tom
- 10.02.1940
- CAT
- „Katzenschnupfen“, „Katzenseuche“, „Tollwut“, „Leukose“
- „22.455.465“

Erstellen Sie auf Basis der Annotationen von JAXB eine XML-Serialisierung des Objektes, wobei:

- der Nickname als Attribut zu unterstützen ist
- die Reihenfolge der Elemente vorgegeben ist (Name, Typ, ID, GebDatum und Impfungen)
- die Elemente ohne den Header „m“ zu serialisieren sind
- die Impfungen unter einem Tag „vacinations“ zusammengefasst sind und die jeweilige Impfung als „vacination“ gekennzeichnet ist
- der Haustier-Typus über einen eigenen Namespace verfügt

Speichern Sie das XML in einen String und deserialisieren Sie den Inhalt in ein neues **`Pet`**-Objekt. Überprüfen Sie über eine entsprechende Ausgabe, ob im neuen **`Pet`**-Objekt, die gleichen Daten enthalten sind wie im Ausgangsobjekt.

Testen Sie danach die (De-)Serialisierung über JSON unter Verwendung der MOXy-Bibliotheken.

Aufgabe 2a „O“ (15%)

Erstellen Sie mittels JAX-WS einen einfachen Web Service **HelloWorld**, der die folgenden beiden Methoden als Web Service-Methoden anbietet:

```
public String saySomething() { … }
public DummyData getData(String _name) { … }
```

Der Inhalt vom **saySomething**-String und **DummyData** ist dabei frei wählbar.

Stellen Sie den Web Service mittels der **Endpoint** Klasse online und testen Sie die Erreichbarkeit mittels eines Browsers.

Implementieren Sie in weiterer Folge einen Client, der über – mittels **wsimport** – generierte Schnittstellen-Klassen mit dem Web Service kommunizieren kann und testen Sie die Funktionsfähigkeit in dem Sie beide Methoden aufrufen und die Rückgabewerte ausgeben.

Implementieren Sie als nächstes einen Client, der über die **Service** Klasse, die die Schnittstellen-Klassen dynamisch generiert, mit dem Web Service kommunizieren kann und testen Sie die Funktionsfähigkeit in dem Sie beide Methoden aufrufen und die Rückgabewerte ausgeben.

Aufgabe 2b „M“ (30%)

Erstellen Sie analog zu den beiden ersten Übungszetteln einen „Environment Data“-Server, der seine Daten mittels SOAP (JAX-WS) zur Verfügung stellt.

Der SOAP Environment Data Service stellt hierfür aktuelle Wetterdaten für einen bestimmten Ort (welche Orte unterstützt werden ergibt sich aus der Rückmeldung der Methode **requestEnvironmentDataTypes()**) zur Verfügung. Verwenden Sie für die Bereitstellung der aktuellen Wetterdaten folgenden REST Web Service:

http://weathers.co/api.php?city=New+York

Verwenden Sie JAXB, um den in JSON zur Verfügung gestellt Wetterinformation in ein Java-Objekt umzuwandeln. Nutzen Sie die numerischen Anteile der Wetterinformation, um ein **EnvData** Objekt entsprechend anzureichern und über die SOAP-Schnittstelle zurückzuliefern.

HINWEIS:
Versuchen Sie zunächst ein Java-Objekt(e) Konstrukt zu erstellen, mit dem Sie die JSON-Nachricht des REST Web Services reproduzieren können, bevor Sie mit dem SOAP-Dienst starten. Im Rahmen der Übung ist es notwendig, den **Unmarschaller** um folgende Codezeilen zu ersetzen/ergänzen:

```
um.setProperty(JAXBContextProperties.JSON_INCLUDE_ROOT, false);
StreamSource json = new StreamSource(new StringReader(dataS));
resp = (WeatherResponse) um.unmarshal(json, WeatherResponse.class).getValue();
```

Testen Sie die Funktionsfähigkeit des Dienstes in dem Sie einen Client implementieren, der alle Methoden aufruft und die Rückgabewerte ausgibt.

Aufgabe 2c „O“ (5%)

Integrieren Sie analog zu den beiden ersten Übungszetteln den SOAP „Environment Data“-Server in das Übersichts-Servlet.

Aufgabe 3a „O“ (10%)

Erstellen Sie mittels JAX-RS einen einfachen Web Service HelloWorld, der die folgenden URLs unterstützt:

```
<Servlet-Context>/HelloWorld       → Übersichts-HTML
<Servlet-Context>/HelloWorld/json  → eine einfache Botschaft in JSON
<Servlet-Context>/HelloWorld/xml   → eine einfache Botschaft in XML
```

Testen Sie die Funktionsfähigkeit des Systems, in dem Sie mit einem Browser die URLs aufrufen und die Ausgaben überprüfen

Aufgabe 3b „M“ (25%)

Erstellen einen REST-basierten „Environment Data“-Server, der seine Daten mittels REST (JAX-RS) zur Verfügung stellt.

Adaptieren Sie hierfür die bisher verwendete „Environment Data“-API wie folgt:

```
<Servlet-Context>/EnvironmentService
→ Übersichts-HTML

<Servlet-Context>/EnvironmentService/sensors
→ eine Liste der unterstützten Sensoren in Form von XML

<Servlet-Context>/EnvironmentService/{Sensor}
→ eine XML Serialisierung des zum Sensor passenden EnvData-Objektes

<Servlet-Context>/EnvironmentService/ALL
→ eine XML Liste entsprechender EnvData-Objekte
```

Der Dienst unterstützt zunächst zufällig genierte Temperatur und Luftfeuchtigkeitswerte, die in weiterer Folge über einen weiteren Dienst zur Verfügung gestellt werden (s. Aufgabenzettel 4).

Testen Sie die Funktionsfähigkeit des Systems, in dem Sie mit einem Browser die URLs aufrufen und die Ausgaben überprüfen.

HINWEIS:
Eine Integration in die Übersichtstabelle erfolgt auf Aufgabenzettel 4