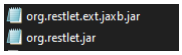
REST

Sinn: Übertragung zwischen Server und Client

Nutzt: **Get**(Lesen) ,**Post**(Hinzufügen) ,**Put**(Update), **Delete**(Löschen)

Brauchen XML um Datenschema zu erstellen -> JAXB **generiert Klassen** aus XSD (Tools/JAXB)

Schritte zum Programmieren:

1. Neues Projekt
2. Libraries einbinden 

erstellen
3. Packages „model“, „server“,“common“ erstellen
4. XSD File erstellen
5. Java Klassen aus XSD erstellen (Tools/Jaxb/Generate Java Code from XSD)
6. Klasse Model erstellen
7. Model in Singelton umwandeln
8. Einlesen im Konstruktor (durch Unmarshaller)
9. Klasse in Datei abspeichern
10. Model ist fertig
11. Interface für alle Model Klassen erstellen
12. Get, Put, Post, Delete benötigte Methoden in allen Interfaces erstellen
13. Für jedes Interface ServerResource erstellen
14. Alle ServerResource Klassen, ServerResource extenden und Interface implementieren und Methoden autom. Implementieren
15. (Generate/Override Methods/DoInit) **Nach Objekt mit ID suchen** -> DoInit implementieren -> als globale Variable speichern
16. Implementieren der Methoden mithilfe der Variable (Model.save vor dem return)
17. Restliche ServerResource implementieren
18. ServerApp erstellen, extends Application
19. (Generate/Override Methos/CreateInbountRoot) überschreiben
20. Aufbauen wie in Bild 17
21. Server erstellen, lt bild 18
22. Client erstellen, Package, Klasse
23. Client Methoden implementieren
24. Testen

24

```
Classroom classroom = new Classroom();
classroom.setId(0);
classroom.setClassname("AHIF");
Student student = new Student();
student.setId(0);
student.setGrade(2);
student.setFirstname("Samuel");
student.setLastname("Pointner");
addStudent(classroom.getId(), student);
```

4

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="school">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="schoolname" type="xs:string"></xs:element>
                <xs:element ref="classroom" maxOccurs="unbounded"></xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="classroom">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="id" type="xs:int"></xs:element>
                <xs:element name="classname" type="xs:string"></xs:element>
                <xs:element ref="student" maxOccurs="unbounded"></xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="student">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="id" type="xs:int"></xs:element>
                <xs:element name="firstname" type="xs:string"></xs:element>
                <xs:element name="lastname" type="xs:string"></xs:element>
                <xs:element name="grade" type="xs:int"></xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

```java
public class Model {
    private static Model instance;

    private Model(){

    }

    public static Model getInstance(){
        if(instance==null){
            instance = new Model();
        }
        return instance;
    }
}
```

7

```java
JAXBContext jaxbContext = null;
try{
    jaxbContext = JAXBContext.newInstance(School.class);
    Unmarshaller jaxbUnmarshaller = jaxbContext.createUnmarshaller();
    this.school = (School) jaxbUnmarshaller.unmarshal(new File(XML_PATH));
}catch (JAXBException e){
    e.printStackTrace();
}
```

8

```java
public void save(){
    JAXBContext jaxbContext = null;
    try{
        jaxbContext = JAXBContext.newInstance(School.class);
        Marshaller marshaller = jaxbContext.createMarshaller();
        marshaller.marshal(school,new File(XML_PATH));
    }catch (JAXBException e){
        e.printStackTrace();
    }
}
```

9

```java
protected void doInit() throws ResourceException {
    super.doInit();
    School school = Model.getInstance().getSchool();
    school.getClassroom().forEach(classroom -> {
        classroom.getStudent().forEach(student -> {
            if(student.getId() == Integer.parseInt(getAttribute( name: "id"))){
                this.student = student;
            }
        });
    });
}
```

15

```java
@Get
public Student getStudent();

@Put
public boolean updateStudent();

@Delete
public boolean deleteStudent();
```

12

```java
public boolean deleteStudent() {
    School school = Model.getInstance().getSchool();
    Classroom studentsClassroom = null;

    for(Classroom classroom : school.getClassroom()){
        for (Student student : classroom.getStudent()){
            if(student.getId() == this.student.getId()){
                studentsClassroom = classroom;
            }
        }
    }
    boolean s = studentsClassroom.getStudent().remove(this.student);
    Model.getInstance().save();
    return s;
}
```

16

19

```java
public Restlet createInboundRoot() {
    Router r = new Router();
    r.attach( pathTemplate: "/School",SchoolServerResource.class);
    r.attach( pathTemplate: "/School/Class/{classID}",ClassServerResource.class);
    r.attach( pathTemplate: "/School/Class/{classID}/Student/{studentID}",StudentServerResource.class);
    return r;
}
```

```java
Component component = new Component();
component.getServers().add(Protocol.HTTP, port: 8888);
component.getDefaultHost().attach( uriPattern: "/api",new SchoolApp());
component.start();
```

18

```java
public static boolean deleteClassroom(int classID){
    ClientResource clientResource = new ClientResource( uri: "http://localhost:8888/api/school/classroom/"+ classID);
    IClassResource resource = clientResource.wrap(IClassResource.class);
    return resource.deleteClassroom();
}

public static boolean addStudent(int studentID, Student student){
    ClientResource clientResource = new ClientResource( uri: "http://localhost:8888/api/school/student/"+ studentID);
    IClassResource resource = clientResource.wrap(IClassResource.class);
    return resource.addStudent(student);
}

public static Student getStudent(int studentID, Student student){
    ClientResource clientResource = new ClientResource( uri: "http://localhost:8888/api/school/student/"+ studentID);
    IStudentResource resource = clientResource.wrap(IStudentResource.class);
    return resource.getStudent();
}
public static boolean updateStudent(int studentID, Student student){
    ClientResource clientResource = new ClientResource( uri: "http://localhost:8888/api/school/student/"+ studentID);
    IStudentResource resource = clientResource.wrap(IStudentResource.class);
    return resource.updateStudent(student);
}
```

23