TGM - HTBLuVA Wien XX
IT Department

Service Oriented Architecture and Restful Webservice

Dezsys 08

Authors: Siegel Hannah & Nachname2 Vorname2

April 11, 2015

Contents

1 Working time

Estimated working time

Task	Person	Time in minutes
Reading into the Topic, Deciding onto Frameworks	Coworker1	120
Creating the Knowledge base	Coworker1	120
Inserting Test-Data into the Knowledge base	Coworker1	60
Testing the Performance of the Knowledge base	Coworker1	60
RestFul CRUD Operations	Coworker1 Coworker2	120 120
Generate SOA Webservice	Coworker1	120
Include SOAP for SOA Webservice	Coworker1	60
Generate WSDL File	Coworker1	60
Generate Website for using RestFul Webservices	Coworker2	120
Generate Client for SOA Webservice	Coworker1	120
Document Datatransfer with SOAP	Coworker2	90
Testing	Coworker1 Coworker2	120 120
Documentation	Coworker1	120
Total	Coworker1 Coworker2	XXX XXX
Total Team		**4 hours**

2 Aufgabenstellung

3 Knowledge Base

For the Knowledge Base we decided to use Hibernate and JPA.
We were using IntelliJ and therefore it was quite easy.
At first, we were making a new project, which has already a Hello World Webservice. We copied the hibernate config file from the *Westbahn* project into the `src` folder.
Also, we had to add all the needed libraries, such as the mysql connector and the hibernate libraries.

The Knowledge Base class can be seen in Listing 1.

Listing 1: KnowledgeBase entity class

```
// import statements

@Entity
public class KnowledgeBase {
    @Id
```

```
    @GeneratedValue
    private Long ID;

    @Column(unique=true)
    @Size(min=10, max=500)
    private String text;

    @Column(unique=true)
    @Size(min=10, max=50)
    private String topic;

        // constructor, getter and setter
}
```

- database connection is not working tho -¿ mocking it - new module : JPA module... (file - new module - JPA, Hibernate, SQL Supp)

brauch nun schon 2 stunden fuer hibernate skipping this

4 Build a RESTful WebService

"RESTful web services are built to work best on the Web. Representational State Transfer (REST) is an architectural style that specifies constraints, such as the uniform interface, that if applied to a web service induce desirable properties, such as performance, scalability, and modifiability, that enable services to work best on the Web. In the REST architectural style, data and functionality are considered resources and are accessed using Uniform Resource Identifiers (URIs), typically links on the Web. The resources are acted upon by using a set of simple, well-defined operations. The REST architectural style constrains an architecture to a client/server architecture and is designed to use a stateless communication protocol, typically HTTP. In the REST architecture style, clients and servers exchange representations of resources by using a standardized interface and protocol."[**?**]

4.1 JAX-RS

Annotations

Resource identification through URI: A RESTful web service exposes a set of resources that identify the targets of the interaction with its clients. Resources are identified by URIs, which provide a global addressing space for resource and service discovery. See The @Path Annotation and URI Path Templates for more information.

Root resource classes are POJOs that are either annotated with @Path or have at least one method annotated with @Path or a request method designator, such as @GET, @PUT, @POST, or @DELETE. Resource methods are methods of a resource class annotated with a request method designator. This section explains how to use JAX-RS to annotate Java classes to create RESTful web services.

References

[1] **The Java EE 6 Tutorial**
http://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html
last used: 06.04.2014, 11:52