JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Dezsys 09

Webservices in Java

Systemtechnik Labor
5BHITT 2015/16

Selina Brinnich

Version 1.0

Note:

Betreuer: Borko

Begonnen am 12. Februar 2016

Beendet am 18. Februar 2016

Inhaltsverzeichnis

1 Einführung

Diese Übung zeigt die Anwendung von mobilen Diensten in Java.

1.1 Ziele

Das Ziel dieser Übung ist eine Webanbindung zur Benutzeranmeldung in Java umzusetzen. Dabei soll sich ein Benutzer registrieren und am System anmelden können.

Die Kommunikation zwischen Client und Service soll mit Hilfe von JAX-RS (Gruppe1) umgesetzt werden.

1.2 Voraussetzungen

- Grundlagen Java und Java EE
- Verständnis über relationale Datenbanken und dessen Anbindung mittels JDBC oder ORM-Frameworks
- Verständnis von Restful Webservices

1.3 Aufgabenstellung

Es ist ein Webservice mit Java zu implementieren, welches eine einfache Benutzerverwaltung implementiert. Dabei soll die Webapplikation mit den Endpunkten /register und /login erreichbar sein.

Registrierung

Diese soll mit einem Namen, einer eMail-Adresse als BenutzerID und einem Passwort erfolgen. Dabei soll noch auf keine besonderen Sicherheitsmerkmale Wert gelegt werden. Bei einer erfolgreichen Registrierung (alle Elemente entsprechend eingegeben) wird der Benutzer in eine Datebanktabelle abgelegt.

Login

Der Benutzer soll sich mit seiner ID und seinem Passwort entsprechend authentifizieren können. Bei einem erfolgreichen Login soll eine einfache Willkommensnachricht angezeigt werden.

Die erfolgreiche Implementierung soll mit entsprechenden Testfällen dokumentiert werden. Es muss noch keine grafische Oberfläche implementiert werden! Verwenden Sie auf jeden Fall ein gängiges Build-Management-Tool.

2 Ergebnisse

2.1 Datenmodell

Als Datenbank für die Persistierung der User wird SQLite verwendet. Dazu wurde mit der SQLite-Shell ein User-Datenbank-File erstellt, in dem Namen, Usernamen und Passwörter für User gespeichert werden können.

2.2 Webservice

Vorbereitung

In Eclipse neues „Dynamic Web Project" erstellen. Jersey downloaden und Jar-Files im Projekt unter WebContent/WEB-INF/lib kopieren. In WebContent/WEB-INF/web.xml folgenden Code innerhalb des „web-app"-Tags hinzufügen:

```
<servlet>
    <servlet-name>Jersey REST Service</servlet-name>
    <servlet-
class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
    <init-param>
      <param-name>com.sun.jersey.config.property.packages</param-name>
      <param-value>mypackage</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
 </servlet>
 <servlet-mapping>
    <servlet-name>Jersey REST Service</servlet-name>
    <url-pattern>/*</url-pattern>
 </servlet-mapping>
```

Den JDBC-Connector für SQLite downloaden und in den lib-Ordner kopieren.

Datenbankverbindung

Unter Java Resources/src eine neue Klasse „DBConnections" hinzufügen, die die Verbindung zur Datenbank und alle Datenbank-Operationen übernimmt.

Unter Java Resources/src eine neue Klasse „Utility" hinzufügen, die Methoden zum Null Check und zum Konstruieren eines Response-JSON enthält.

Register & Login

Eine neue Klasse „Register" erstellen, die die Registrierung von neuen Usern übernimmt. Diese Klasse enthält eine Methode „registerUser", die einen Namen, einen Usernamen und ein Passwort als Parameter bekommt und damit

versucht einen neuen User in der Datenbank anzulegen. Sollte dabei ein Fehler auftreten, wird ein entsprechender Return-Code zurückgegeben, der auf den jeweiligen Fehler hinweist. (0=ok, 1=already_registered, 2=error)
Zudem gibt es in dieser Klasse eine Methode „doRegister", die durch die Annotation @GET Get-Requests auf @Path(„/doregister") empfängt und durch @Produces(MediaType.APPLICATION_JSON) als Response JSON zurückgibt.

Eine neue Klasse „Login" erstellen, die den Login von bereits bestehenden Usern übernimmt. Diese Klasse enthält eine Methode „checkCredentials", die einen Usernamen und ein Passwort als Parameter bekommt und überprüft, ob dieser User in der Datenbank existiert. Zudem gibt es in dieser Klasse eine Methode „doLogin", die durch die Annotation @GET Get-Requests auf @Path(„/dologin") empfängt und durch @Produces(MediaType.APPLICATION_JSON) als Response JSON zurückgibt.

Aufruf

Ruft man nun den Web-Service unter der URL „/register/doregister?name=Name&username=Username&password=pwd" auf, erhält man folgende Erfolgsmeldung in JSON-Format zurück:

```
{"tag":"register","status":true}
```

Abbildung 1: JSON-Response nach erfolgreicher Registrierung

Nun kann man sich mit dem eben erstellten User einloggen. Dazu wird die URL „/login/dologin?username=Username&password=pwd" aufgerufen, die folgende Willkommens-Meldung zurückgibt:

```
{"tag":"login","status":true,"msg":"Willkommen, Testyuser!"}
```

Abbildung 2: JSON-Response nach erfolgreichem Login

3 Quellen

[1] **Android Restful Webservice Tutorial – How to create RESTful webservice in Java – Part 2**
Android Guru
http://programmerguru.com/android-tutorial/android-restful-webservice-tutorial-how-to-create-restful-webservice-in-java-part-2/
Zuletzt besucht am 12.02.2016

4 Abbildungsverzeichnis

5 Zeitaufzeichnung

Datum	Dauer	Beschreibung
12.02.2016	3h	Datenbank Konfiguration MySQL / Webservice erstellt laut Tutorial
16.02.2016	2h	Datenbank von MySQL zu SQLite geändert
18.02.2016	1h	Maven-Konfiguration / Deployment auf Tomcat-Server / Verbessertes Exception-Handling

6 GitHub-Repository

Der gesamte Source-Code ist auf GitHub unter folgendem Link einsehbar:

https://github.com/sbrinnich-tgm/DezSys09-Webservices-in-Java