JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Laborprotokoll

Web Services in Java

Systemtechnik Labor
5BHITT 2015/16, Gruppe Y

Burkhard Hampl

Version 1

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

Ich hab mich an das Tutorial von Spring gehalten und die Grundlegenden Dinge, welche ich nicht mehr wusste von der Übung Dezsys07 angeschaut. Nach dem das User Datenmodell erstellt wurde, konnte mit der Implementierung der Rest-Schnittstelle begonnen werden. Dieser Teil war der Aufwendigste da das Verarbeiten der Requests und das richtige Handling der Verschiedenen Fälle nicht ganz Trivial war.

2.1 Tests

2.1.1 Create User Test

Test-Request (http://localhost:8080/register):

```
{
 "email": "mmustermann@example.com",
 "name": "Max Mustermann",
 "password": "testpassw0rd"
}
```

Erwarteter Wert:

```
{
 "reason": "Created",
 "code": "201",
 "message": "User successfully created!"
}
```

Tatsächlicher Wert:

```
{
 "reason": "Created",
 "code": "201",
 "message": "User successfully created!"
}
```

2.1.2 Login

Test-Request (http://localhost:8080/login):

```
{
 "email": "mmustermann@example.com",
 "password": "testpassw0rd"
}
```

Erwarteter Wert:

```
{
 "reason": "OK",
 "code": "200",
 "message": "Welcome Max Mustermann!"
}
```

Tatsächlicher Wert:

```
{
 "reason": "OK",
 "code": "200",
 "message": "Welcome Max Mustermann!"
}
```

2.1.3 User already exists

Test-Request (http://localhost:8080/register):

```
{
 "email": "mmustermann@example.com",
 "name": "Max Mustermann",
 "password": "testpassw0rd"
}
```

Erwarteter Wert:

```
{
 "reason": "Bad Request",
 "code": "400",
 "message": "User already exists"
}
```

Tatsächlicher Wert:

```
{
 "reason": "Bad Request",
 "code": "400",
 "message": "User already exists"
}
```

2.1.4 Wrong Username

Test-Request (http://localhost:8080/login):

```
{
 "email": "wrong@wrong.wrong",
 "password": "testpassw0rd"
}
```

Erwarteter Wert:

```
{
 "reason": "Forbidden",
 "code": "403",
 "message": "Email or password is wrong"
}
```

Tatsächlicher Wert:

```
{
 "reason": "Forbidden",
 "code": "403",
 "message": "Email or password is wrong"
}
```

2.1.5 Wrong Password

Test-Request (http://localhost:8080/login):

```
{
 "email": "mmustermann@example.com",
 "password": "wrongpassw0rd"
}
```

Erwarteter Wert:

```
{
 "reason": "Forbidden",
 "code": "403",
 "message": "Email or password is wrong"
}
```

Tatsächlicher Wert:

```
{
 "reason": "Forbidden",
 "code": "403",
 "message": "Email or password is wrong"
}
```

2.2 Probleme

- Die meisten Probleme hatte ich dadurch das TGM-Netz mal wieder recht langsam war und ich warten musste bis alle Dependencies heruntergeladen waren.
- Es war mir nicht möglich eine HTTP Error Message (also ein HTTP Status-Code 20x) ohne jeglichen Content zu senden, dann bekam der „Client" immer eine 404 Message.
- Manchmal kam es zu komischen Dependencies Problemen, wenn man zu viele oder fehlende Abhängigkeiten hatte.

2.3 Github Repo Link

Link zum Github Repo: https://github.com/bhampl-tgm/DezSys-09

2.4 Zeitaufzeichung

Datum	Zeit in h	Arbeit
12.02.2016	2	Projekt aufgesetzt und grobe Implementation Durchgeführt
12.02.2016	1	Versucht den „Empty Resonse" zu fixen
18.02.2016	2	Aufgabe Fertiggestellt

2.4.1.13 Quellen

[1] "Android Restful Webservice Tutorial – Introduction to RESTful webservice – Part 1"; Posted By Android Guru on May 1, 2014; online: http://programmerguru.com/android-tutorial/android-restful-webservice-tutorial-part-1/

[2] "REST with Java (JAX-RS) using Jersey - Tutorial"; Lars Vogel; Version 2.5; 15.12.2015; online: http://www.vogella.com/tutorials/REST/article.html

[3] "O Java EE 7 Application Servers, Where Art Thou? Learn all about the state of Java EE app servers, a rundown of various Java EE servers, and benchmarking."; by Antonio Goncalves; Java Zone; Feb. 10, 2016; online: https://dzone.com/articles/o-java-ee-7-application-servers-where-art-thou

[4] „"Bootiful" Java EE Support in Spring Boot 1.2“, by Josh Long, Nov. 23, 2014; online: https://spring.io/blog/2014/11/23/bootiful-java-ee-support-in-spring-boot-1-2