JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Laborprotokoll

Java Webservice

Systemtechnik Labor
5BHITT 2015/16, Gruppe 1

Niklas Hohenwarter

Version 0.2

Note:

Betreuer: Michael Borko

Begonnen am 12. Februar 2016

Beendet am 16. 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 Datenbank

Erstellen einer sqlite Datenbank über die sqlite Konsole.

```
sqlite3 user.db
```

Verwenden des DDL Scripts aus dem Example[1]:

```
CREATE TABLE IF NOT EXISTS `user` (
  `name` varchar(50) NOT NULL,
  `username` varchar(50) NOT NULL,
  `password` varchar(50) NOT NULL,
  `register_dt` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`username`)
)
```

2.2 Tutorial

Zur Realisierung der Aufgabe wurde das Tutorial von Programmerguru.com[1] verwendet.

- Erstellen eines Dynamischen Web Projektes in Eclipse
- Herunterladen von Jersey und Entpacken in den WEB-INF/lib Ordner
- Herunterladen des Mysql Connectors und hinzufügne in den WEB-INF/lib Ordner
- Anpassen der web.xml
- Erstellen der Klasse Constants welche die Parameter für die Datenbankverbindung enthält
- Erstellen der Klasse DBConnection welche eine Verbindung zur Datenbank mittels JDBC aufbaut
- Erstellen der Klasse Utility welche Methoden zum erstellen von JSON Datenstrukturen enthält
- Erstellen der Klasse Register mit welcher man User erstellen kann
- Erstellen der Klasse Login mit welcher man sich einloggen kann

Wenn man das Tutorial befolgt hat dann sollten die beiden Funktionen mit folgenden URL Aufrufen getestet werden können:

- Registrierung: http://localhost:8080/javawebservice/register/doregister?name=erster&username=erster&password=1234
- Login: http://localhost:8080/javawebservice/login/dologin?username=erster1&password=1234

2.3 Anpassungen

Datenbank

Die Klasse Constants wurde entfernt, da sie bei einer sqlite Datenbank nicht benötigt wird. Das Datenbankfile wurde in WEB-INF/database abgelegt. Dannach wurde in der DBConnection Klasse folgende Methode geändert:

```
public static Connection createConnection() throws Exception {

    Connection c = null;
    try {
       Class.forName("org.sqlite.JDBC");
       String path = DBConnection.class.getClassLoader().getResource("").getPath();
       String fullPath = URLDecoder.decode(path, "UTF-8");
       c = DriverManager.getConnection("jdbc:sqlite:"+fullPath+"../database/"+"users.db");
    } catch ( Exception e ) {
       System.err.println( e.getClass().getName() + ": " + e.getMessage() );
       System.exit(0);
    }finally {
       return c;
    }
  }
```

Build Automation

Anstatt die JARs herunterzuladen wurde Maven zur Build Automation verwendet. Dazu wird folgender Teil des pom.xml benötigt:

```
<dependencies>
      <dependency>
            <groupId>org.xerial</groupId>
            <artifactId>sqlite-jdbc</artifactId>
            <version>3.8.11.2</version>
      </dependency>
      <dependency>
            <groupId>com.sun.jersey</groupId>
            <artifactId>jersey-core</artifactId>
            <version>1.19</version>
      </dependency>
      <dependency>
            <groupId>com.sun.jersey</groupId>
            <artifactId>jersey-json</artifactId>
            <version>1.19</version>
      </dependency>
  </dependencies>
```

Das automatische Deployment auf einen embedded Application Server gestaltet sich schwierig(->Nachfragen).

3 Zeitaufzeichnung

Datum	Dauer	Beschreibung
12.02.2016	3h	Tutorial durchführen
16.02.2016	2h	Anpassungen

4 Quellen

[1]Anroid Guru, Android Restful Webservice Tutorial – How to create RESTful webservice in Java – Part 2, http://programmerguru.com/android-tutorial/android-restful-webservice-tutorial-how-to-create-restful-webservice-in-java-part-2/, zuletzt besucht: 12.02.16 (offline version im repo)