
Laborprotokoll

Web Services in Java

**Systemtechnik Labor
5BHITT 2015/16, Gruppe 1**

Stefan Erceg

Version 1.0

Note:

Betreuer: Prof. Borko

Begonnen am 12. Februar 2016

Beendet am 18. Februar 2016

Inhaltsverzeichnis

1	Einführung	3
1.1	Ziele	3
1.2	Voraussetzungen	3
1.3	Aufgabenstellung.....	3
2	Ergebnisse	4
2.1	RESTful Webservice mittels Jersey erstellen	4
2.2	file-basiertes DBMS statt MySQL verwenden	5
2.3	Aufgetretene Probleme beim Tomcat Server.....	6
2.4	Acceptance Tests	7
2.4.1	Registrierung.....	7
2.4.2	Login.....	8
2.5	Starten der Applikation	9
3	Quellen	9

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 Datenbanktabelle 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

Die gesamte Übung wurde in der Eclipse IDE für Java EE Entwickler (Downloadlink: [1]) durchgeführt.

2.1 RESTful Webservice mittels Jersey erstellen

Um ein RESTful Webservice mittels Jersey umzusetzen, wurde ein Tutorial [2] als Hilfestellung verwendet. Voraussetzung hierbei ist jedoch, dass man einen Tomcat- und MySQL-Server, entweder lokal oder auf einer virtuellen Maschine, besitzt.

Das Tutorial besitzt 5 Java-Klassen:

- **Constants**
zur Angabe des DB-Namens, der URL, des Users, dessen Passwort, etc.
- **DBConnection**
zur Verbindung mit der Datenbank und zum Inserten von Datensätzen
- **Register**
damit ein neuer User durch Aufrufen des Pfades
`http://<ip-address>:<port>/<appln-folder-name>/register/doregister?name=<name>&username=<email>&password=<password>`
in die Datenbank gespeichert und somit registriert wird
- **Login**
damit ein bestimmter User durch Aufrufen des Pfades
`http://localhost/<appln-folder-name>/login/dologin?username=abc&password=xyz`
überprüfen kann, ob er erfolgreich eingeloggt werden konnte oder sich noch nicht registriert bzw. eine falsche E-Mail-Adresse und/oder ein falsches Passwort eingegeben hat
- **Utility**
entwickelt ein JSON-Konstrukt mit einem Tag, einem Status und einer Nachricht, z.B. bei einer erfolgreichen Registrierung:

```
{"tag": "register", "status": true, "msg": "You have been successfully registered"}
```

Nach dem Erstellen eines Maven-Projekts konnte die Applikation nach einigen aufgetretenen Problemen beim Tomcat Server (siehe Kapitel 2.3) auf dem Server gestartet werden. Die notwendigen Libraries, die einzubinden waren, wurden im pom.xml – File definiert.

2.2 file-basiertes DBMS statt MySQL verwenden

Damit ein file-basiertes Datenbankmanagementsystem (bei mir fiel die Entscheidung auf SQLite) verwendet wird, müssen einige Änderungen vorgenommen werden:

Zu Beginn kann die Java-Klasse „Constants.java“ gelöscht werden, da dort lediglich der Username, dessen Passwort, der Datenbankname und weitere Eigenschaften definiert wurden und diese bei einem file-basierten DBMS nicht benötigt werden.

Weiters wurde folgendermaßen in der Methode „createConnection“ aus der Klasse „DBConnection“ angegeben, dass der SQLite-Driver verwendet werden soll und wo sich das File „users.db“ befindet:

```
Class.forName("org.sqlite.JDBC");
String path = DBConnection.class.getClassLoader().getResource("").getPath();
String fullPath = URLDecoder.decode(path, "UTF-8");
con=DriverManager.getConnection("jdbc:sqlite:"+fullPath+"../database/"+"users.db");
```

Zu guter Letzt musste man noch in der Methode „registerUser“, welche sich in der Klasse „Register“ befindet, definieren, dass die Meldung „You are already registered“ ausgegeben werden soll, wenn die SQL-Fehlermeldung „UNIQUE constraint failed“ auftaucht. Dies wurde mit folgender if-Unterscheidung umgesetzt:

```
} catch(SQLException sqle) {
    if(sqle.getMessage().startsWith("UNIQUE constraint failed"))
        ...
}
```

2.3 Aufgetretene Probleme beim Tomcat Server

Bevor ich die Applikation starten konnte, sind folgende 2 Probleme beim Tomcat Server aufgetaucht:

Zu Beginn tauchte beim Starten der Applikation die Fehlermeldung `Server Tomcat v6.0 Server at localhost failed to start` auf. Diese wurde folgendermaßen in Eclipse behoben:

- 1.) Run -> Run Configurations
- 2.) Auf Server „Tomcat v6.0 Server at localhost“ klicken
- 3.) Zum Tab „Classpath“ wechseln
- 4.) Button „Add external jar“ klicken
- 5.) File „tomcat-juli.jar“ auswählen (bei mir unter dem Verzeichnis
C:\Program Files\Tomcat\bin)
- 6.) Buttons „Apply“ & „Run“ klicken

[3]

Nach der Behebung dieses Problems bekam ich den HTTP Status 404 nach dem Starten der Applikation. Um diesen Fehler zu beheben, musste wie folgt vorgegangen werden:

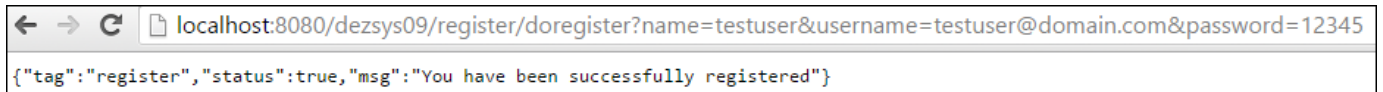
- 1.) Window -> Show view -> Server
- 2.) Rechtsklick auf Server „Tomcat ...“ und „Properties“ auswählen
- 3.) Unter „General“ auf den Button „Switch Location“ klicken
- 4.) Buttons „Apply“ & „Ok“ klicken
- 5.) Doppelklick auf Server „Tomcat ...“
- 6.) Unter „Server locations“ „Use Tomcat installation“ auswählen
- 7.) Speichern & Server restarten

[4]

2.4 Acceptance Tests

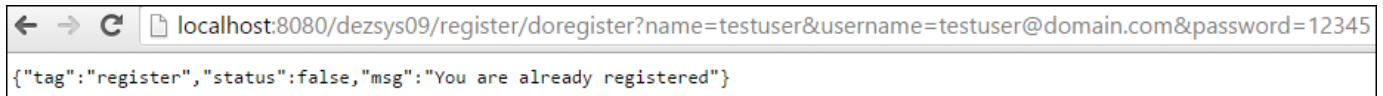
2.4.1 Registrierung

Testfall	Erfolgreich	Abbildungsverzeichnis
neuer User registriert sich, Meldung „You have been successfully registered“ erscheint	✓	Abb1
User will sich registrieren, verwendet bereits vorhandene E-Mail, Meldung „You are already registered“ erscheint	✓	Abb2



A screenshot of a web browser window. The address bar shows the URL: `localhost:8080/dezsys09/register/doregister?name=testuser&username=testuser@domain.com&password=12345`. The main content area displays a JSON response: `{"tag": "register", "status": true, "msg": "You have been successfully registered"}`.

Abbildung 1: User registriert sich

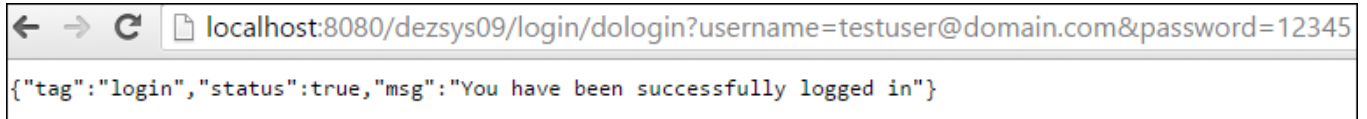


A screenshot of a web browser window. The address bar shows the URL: `localhost:8080/dezsys09/register/doregister?name=testuser&username=testuser@domain.com&password=12345`. The main content area displays a JSON response: `{"tag": "register", "status": false, "msg": "You are already registered"}`.

Abbildung 2: User verwendet bereits vorhandene E-Mail

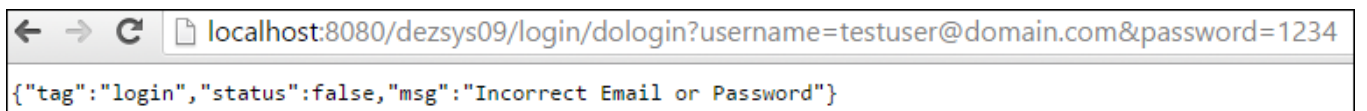
2.4.2 Login

Testfall	Erfolgreich	Abbildungsverzeichnis
User, welcher sich vorher registriert hat, versucht sich einzuloggen, Meldung „You have been successfully logged in“ erscheint	✓	Abb3
User, welcher sich vorher registriert hat, gibt falsches Passwort ein, Meldung „Incorrect Email or Password“ erscheint	✓	Abb4
User, welcher sich vorher registriert hat, gibt falsche E-Mail-Adresse ein, Meldung „Incorrect Email or Password“ erscheint	✓	Abb5



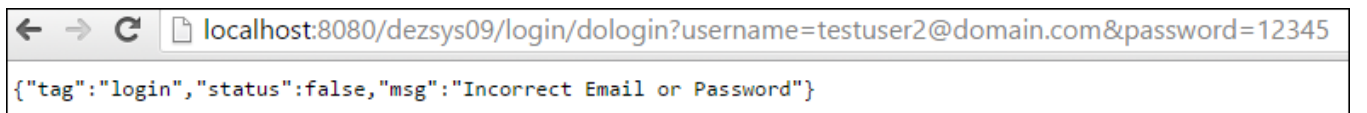
```
localhost:8080/dezsys09/login/dologin?username=testuser@domain.com&password=12345
{"tag":"login","status":true,"msg":"You have been successfully logged in"}
```

Abbildung 3: User loggt sich erfolgreich ein



```
localhost:8080/dezsys09/login/dologin?username=testuser@domain.com&password=1234
{"tag":"login","status":false,"msg":"Incorrect Email or Password"}
```

Abbildung 4: User gibt falsches Passwort ein



```
localhost:8080/dezsys09/login/dologin?username=testuser2@domain.com&password=12345
{"tag":"login","status":false,"msg":"Incorrect Email or Password"}
```

Abbildung 5: User gibt falsche E-Mail-Adresse ein

2.5 Starten der Applikation

Um mein WAR-File, welches ich abgegeben habe, ausführen zu können, ist es notwendig diesen auf einem Tomcat Server (ich habe die Version 6.0.45 verwendet) zu deployen. Dieser kann auf folgender Seite heruntergeladen werden: [5].

Mein Github-Repository, welches alle notwendigen Sources und das pom.xml – File beinhaltet, ist unter folgender URL aufrufbar: [6].

3 Quellen

- [1] The Eclipse Foundation (2016). Eclipse IDE for Java EE Developers [Online]. Available at: <http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/keplerr> [zuletzt abgerufen am 18.02.2016]
- [2] Android Guru (Mai 2014). Android Restful Webservice Tutorial – How to create RESTful webservice in Java – Part 2 [Online]. Available at: <http://programnerguru.com/android-tutorial/android-restful-webservice-tutorial-how-to-create-restful-webservice-in-java-part-2/> [zuletzt abgerufen am 18.02.2016]
- [3] User „PVince81“ (April 2009). tomcat6 + eclipse not working [Online]. Available at: <https://forums.opensuse.org/showthread.php/391114-tomcat6-eclipse-not-working> [zuletzt abgerufen am 18.02.2016]
- [4] User „Ask De“ (2011). http Status 404 error in tomcat [Online]. Available at: <http://www.coderanch.com/t/87666/Tomcat/HTTP-Status-error-tomcat> [zuletzt abgerufen am 18.02.2016]
- [5] The Apache Software Foundation (1999, 2016). Tomcat 6 Downloads [Online]. Available at: <https://tomcat.apache.org/download-60.cgi> [zuletzt abgerufen am 18.02.2016]
- [6] Stefan Erceg (Februar 2016). Github-Repository „DezSys09_WebServices“ [Online]. Available at: https://github.com/serceg-tgm/DezSys09_WebServices [zuletzt abgerufen am 18.02.2016]