

**Laborprotokoll**

**Web Services in Java**

**Systemtechnik Labor**

**5BHITT 2015/16, Gruppe X**

**Andreas Ernhofer**

|  |  |
| --- | --- |
|  | **Version 1.0** |
| **Note:** | **Begonnen am 12. Februar 2016** |
| **Betreuer: M. Borko** | **Beendet am 19. Februar 2016** |

Inhaltsverzeichnis

[1 Einführung 3](#_Toc443504406)

[1.1 Ziele 3](#_Toc443504407)

[1.2 Voraussetzungen 3](#_Toc443504408)

[1.3 Aufgabenstellung 3](#_Toc443504409)

[1.4 Bewertung 3](#_Toc443504410)

[2 Ergebnisse 4](#_Toc443504411)

[2.1 Datenbank 4](#_Toc443504412)

[2.2 Webservice-Schnittstelle 6](#_Toc443504413)

[2.3 AcceptanceTests 14](#_Toc443504414)

[3 Probleme 15](#_Toc443504415)

[4 Zeitaufzeichnung 19](#_Toc443504416)

[5 Quellen 20](#_Toc443504417)

**Abbildungsverzeichnis**

[Abbildung 1: Anlegen eines neuen Projektes 6](#_Toc443504371)

[Abbildung 2: URL zusammensetzung beim Registrieren [ANDR2] 9](#_Toc443504372)

[Abbildung 3: URL Zusammensetzung beim Login [ANDR2] 11](#_Toc443504373)

[Abbildung 4: Manuelles einfügen eines Adapters 16](#_Toc443504374)

# Einführung

Diese Übung zeigt die Anwendung von mobilen Diensten in Java.

## 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.

## Voraussetzungen

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

## 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.

## Bewertung

- Aufsetzen einer Webservice-Schnittstelle (4 Punkte)  
- Registrierung von Benutzern mit entsprechender Persistierung (4 Punkte)  
- Login und Rückgabe einer Willkommensnachricht (3 Punkte)  
- AcceptanceTests (3 Punkte)  
- Protokoll (2 Punkte)

# Ergebnisse

## Datenbank

Für mein Webservice benötige ich eine Datenbank um meine Userdaten zu speichern. Dabei habe ich mich für eine MySQL-Datenbank entschieden. Um diese Datenbank zur Zur Verfügung stelle ich diese Datenbank mittels vagrant. Dazu habe ich in meinem Verzeichnis einen vagrant Ordner angelegt. In diesem Ordner habe ich mittels *vagrant init* ein Vagrantfile erzeugt. Dieses habe ich dann ähnlich an ein zur Verfügung gestelltem [VAGM] angepasst:

# -\*- mode: ruby -\*-

# vi: set ft=ruby :

VAGRANTFILE\_API\_VERSION = "2"

Vagrant.configure(VAGRANTFILE\_API\_VERSION) do |config|

config.vm.box = "hashicorp/precise64"

config.vm.provider "virtualbox" do |v|

v.name = "dezsys09\_mysql"

end

config.vm.network :forwarded\_port, guest: 3306, host: 12345

config.vm.provision :shell, :path => "install.sh"

config.vm.synced\_folder ".", "/vagrant", :mount\_options => ["dmode=777", "fmode=666"]

config.vm.network "private\_network", ip: "192.168.56.56"

end

Beim Starten wird ein File namens instll.sh ausgeführt um ein Update auszuführen, sowie mysql zu installieren und konfigurieren. Dieses beinhaltet folgenden Inhalt:

#!/usr/bin/env bash

sudo apt-get update

sudo debconf-set-selections <<< 'mysql-server mysql-server/root\_password password root'

sudo debconf-set-selections <<< 'mysql-server mysql-server/root\_password\_again password root'

sudo apt-get install -y vim curl python-software-properties

sudo apt-get update

sudo apt-get -y install mysql-server

sed -i "s/^bind-address/#bind-address/" /etc/mysql/my.cnf

mysql -u root -proot < /vagrant/create.sql

mysql -u root -proot -e "GRANT ALL PRIVILEGES ON users.\* TO 'user'@'%' IDENTIFIED BY 'user' WITH GRANT OPTION; FLUSH PRIVILEGES;"

sudo /etc/init.d/mysql restart

In die Datenbank wird eine Datei namens *create.sql* geladen. Ihr Inhalt sieht folgendermaßen aus:

USE mysql;

CREATE USER user IDENTIFIED BY 'user';

CREATE DATABASE IF NOT EXISTS users;

USE users;

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`)

)

Mit dem Befehl *vagrant up* kann nun eine VM erzeugt/gestartet werden. Diese installiert automatisch eine MySQL Datenbank mit dem Namen *users*. Es ist erlaubt mit dem Benutzer *user* auf diese Datenbank zuzugreifen. Dazu muss folgender Befehl verwendet werden:

mysql -h 192.168.56.56 -u user -puser

## Webservice-Schnittstelle

Zu Beginn habe ich in IntelliJ ein neues maven-Projekt angelegt. Habe ich meine aktuelle Java jdk Installation und das zu verwendende Archtype angegeben. Dabei habe ich folgende Einstellungen getroffen: Meine Java Version ist 1.8 und Archtype habe ich einen neuen hinzugefügt.

GroupId: org.codehause.mojo.archetype

ArtifactId: webapp-j2ee14,

Version: 1.1.

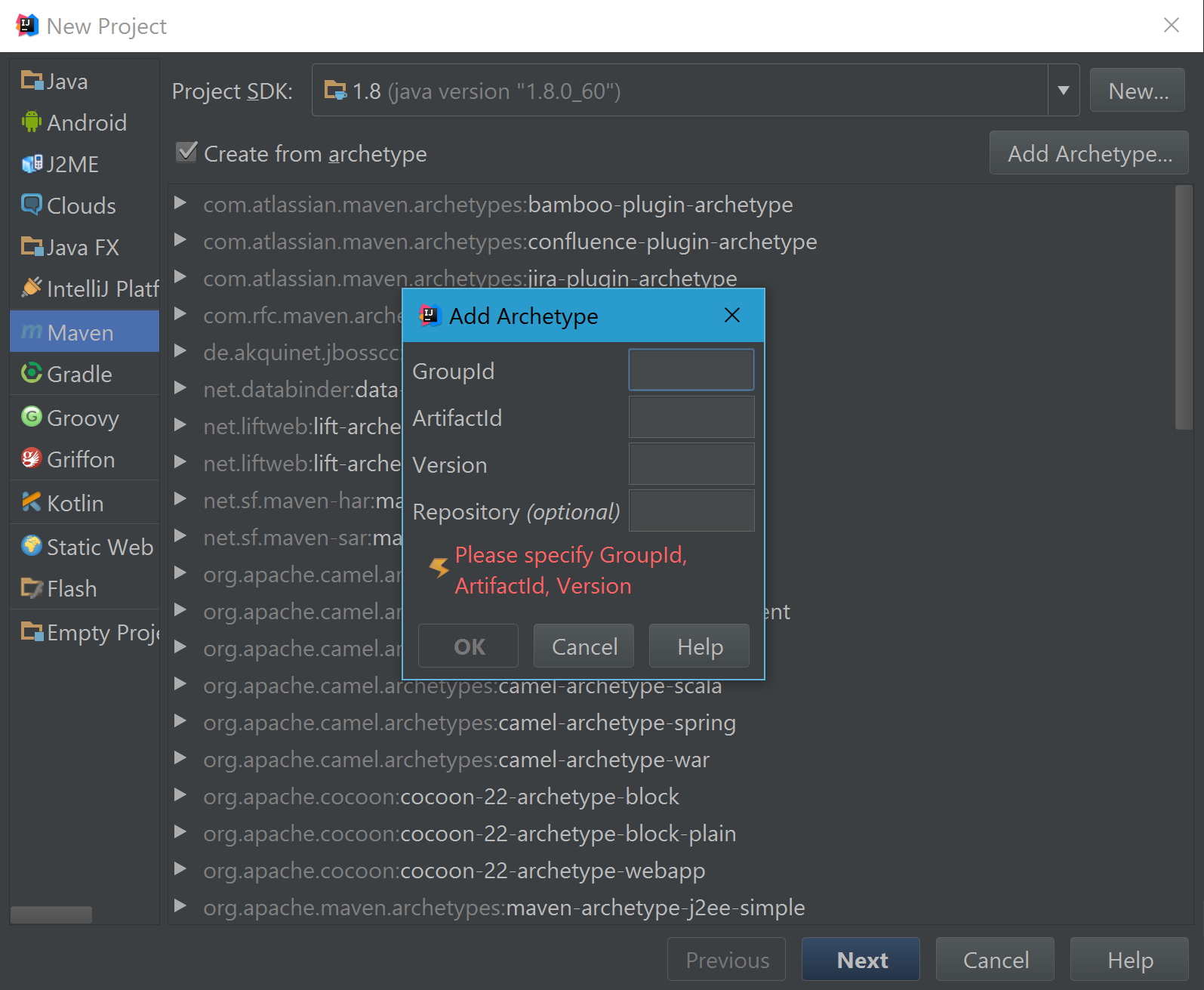


Abbildung : Anlegen eines neuen Projektes

Anschließend habe ich noch eine GroupId, sowie eine ArtefactId für mein Projekt vergeben und habe es erzeugt.

Danach mit Hilfe des Tutorials[ANDR2] ein Restful Webservice erstellt. Zuerst habe ich meine *web.xml* Datei geöffnet und den Inhalt durch den Folgenden ersetzt. Web.xml befindet sich unter src/main/webapp/WEB-INF/web.xml

*<?***xml version="1.0" encoding="UTF-8"***?>*<**web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app\_2\_4.xsd"**>  
 <**display-name**>WebServicesInJava</**display-name**>  
 <**welcome-file-list**>  
 <**welcome-file**>index.jsp</**welcome-file**>  
 </**welcome-file-list**>  
 <**session-config**>  
 <**session-timeout**>  
 30  
 </**session-timeout**>  
 </**session-config**>  
 <**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**>DEZSYS09.WebServicesInJava.ernhofer</**param-value**>  
 </**init-param**>  
 <**load-on-startup**>1</**load-on-startup**>  
 </**servlet**>  
 <**servlet-mapping**>  
 <**servlet-name**>Jersey REST Service</**servlet-name**>  
 <**url-pattern**>/ernhofer/\*</**url-pattern**>  
 </**servlet-mapping**>  
</**web-app**>

Anschließend habe ich in dem Package DEZSYS09.WebServicesInJava.ernhofer meine ersten Java Dateien angelegt Dabei habe ich mich zuerst bei dem Tutorial[ANDR2] von Android Guru erkundigt und anschließend ähnliche Klassen erstellt. Begonnen habe ich mit der *Constants*.*java*. Diese Datei beinhaltet alle meine Konstanten. Im Konkreten, meine Werte um auf die Datenbank zugreifen zu können.

Allein durch das Festlegen der Verbindungsdaten besteht jedoch noch keine Verbindung. Daher habe ich eine Datei *DBConnection.java* erstellt. Diese ist dafür zuständig eine Verbindung mit der Datenbank herzustellen und die Daten zu verwalten.

Die dritte Datei trägt den Namen *Utility.java*. Diese ist dafür zuständig JSON-Objekte zu erzeugen und zu verwalten.

Die nächsten beiden Klassen, Register und Login, beinhalten den wichtigsten Teil für das Web Service. Sie kümmern sich um das Layout der URL und was anschließend passieren soll.

Alle diese Dateien finden sich in meinem Repository wieder. [REPD]

Als Erste der Beiden habe ich die Register Klasse erzeugt, da Benutzer logischerweise zuerst registriert werden müssen, bevor sie sich einloggen können.

*//Path: http://localhost/<appln-folder-name>/register*@Path(**"/register"**)  
**public class** Register {  
 *// HTTP Get Method* @GET   
 *// Path: http://localhost/<appln-folder-name>/register/doregister* @Path(**"/doregister"**)   
 *// Produces JSON as response* @Produces(MediaType.***APPLICATION\_JSON***)   
 *// Query parameters are parameters: http://localhost[:port]/<appln-folder-name>/register/doregister?name=pqrs&username=abc&password=xyz* **public** String doLogin(@QueryParam(**"name"**) String name, @QueryParam(**"username"**) String uname, @QueryParam(**"password"**) String pwd){  
 String response = **""**;**int** retCode = registerUser(name, uname, pwd);  
 **if**(retCode == 0){  
 response = Utility.*constructJSON*(**"register"**,**true**);  
 }**else if**(retCode == 1){  
 response = Utility.*constructJSON*(**"register"**,**false**, **"You are already registered"**);  
 }**else if**(retCode == 2){  
 response = Utility.*constructJSON*(**"register"**,**false**, **"Special Characters are not allowed in Username and Password"**);  
 }**else if**(retCode == 3){  
 response = Utility.*constructJSON*(**"register"**,**false**, **"Error occured"**);  
 }  
 **return** response;  
 }  
 **private int** registerUser(String name, String uname, String pwd){   
 **int** result = 3;  
 **if**(Utility.*isNotNull*(uname) && Utility.*isNotNull*(pwd)){  
 **try** {  
 **if**(DBConnection.*insertUser*(name, uname, pwd)){   
 result = 0;  
 }  
 } **catch**(SQLException sqle){   
 *//When Primary key violation occurs that means user is already registered* **if**(sqle.getErrorCode() == 1062){  
 result = 1;  
 }   
 *//When special characters are used in name,username or password* **else if**(sqle.getErrorCode() == 1064){  
 System.***out***.println(sqle.getErrorCode());  
 result = 2;  
 }  
 }  
 **catch** (Exception e) {  
 result = 3;  
 }  
 }**else**{  
 result = 3;  
 }  
 **return** result;  
 }  
}

Diese Klasse kümmert sich nun darum einen Benutzer zu registrieren Die URL um einen Benutzer anzumelden stellt sich nun wie folgt zusammen, bzw. wird sie in dieser Reihenfolge abgearbeitet. Gut veranschaulicht wird dies in einem Bild des Tutorials von Android Guru, auch wenn der Code nicht zu 100% meinem Entspricht.



Abbildung : URL zusammensetzung beim Registrieren [ANDR2]

Als Antwort auf die Anfrage wird von der Klasse ein JSON erzeugt. Dieses beinhaltet jedenfalls den Inhalt *„register“*. Zusätzlich wird *true* angehängt, sollte die Registrierung erfolgreich gewesen sein. Sollte die Registrierung fehlschlagen, so wird dem JSON *false* angehängt, sowie eine Beschreibung des aufgetretenen Fehlers. Wenn der Benutzer beispielsweiße schon registriert ist und versucht sich erneut zu registrieren, ist dies nicht möglich und es wird folgendes JSON zurückgegeben:

{"tag":"register","status":false,"error\_msg":"You are already registered"}

Sollte alles in Ordnung und die Registrierung gelungen sein, so wird folgendes JSON zurückgegeben:

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

Die URL um einen Benutzer zu registrieren sieht beispielsweiße wie folgt aus:

<http://localhost:8080/WebServicesInJava/ernhofer/register/doregister?name=Admin&username=admin@example.com&password=password>.

Bei der Anwendung dieses Links ist es wichtig, dass die IP-Adresse sowie der Port an die Verwendeten Werte angepasst werden.

Um das Einloggen von Benutzern kümmert sich eine Klasse namens *Login*. Der Inhalt der Datei sieht folgendermaßen aus:

*//Path: http://localhost/<appln-folder-name>/login*@Path(**"/login"**)  
**public class** Login {  
 *// HTTP Get Method* @GET   
 *// Path: http://localhost/<appln-folder-name>/login/dologin* @Path(**"/dologin"**)  
 *// Produces JSON as response  
 //@Produces(MediaType.APPLICATION\_JSON)* @Produces(MediaType.***TEXT\_PLAIN***)  
 *// Query parameters are parameters: http://localhost/<appln-folder-name>/login/dologin?username=abc&password=xyz* **public** String doLogin(@QueryParam(**"username"**) String uname, @QueryParam(**"password"**) String pwd){  
 String response = **""**;  
 **if**(checkCredentials(uname, pwd)){  
 *//response = Utility.constructJSON("login",true);* response = **"Login erfolgreich! Willkommen, sie sind als "** + uname + **" angemeldet!"**;  
 }**else**{  
 *//response = Utility.constructJSON("login", false, "Incorrect Email or Password");* response = **"Login fehlgeschlagen! Email und Password stimmen nicht ueberein!"**;  
 }  
 **return** response;   
 }  
   
 */\*\*  
 \* Method to check whether the entered credential is valid  
 \*   
 \** ***@param uname*** *\** ***@param pwd*** *\** ***@return*** *\*/* **private boolean** checkCredentials(String uname, String pwd){  
 System.***out***.println(**"Inside checkCredentials"**);  
 **boolean** result = **false**;  
 **if**(Utility.*isNotNull*(uname) && Utility.*isNotNull*(pwd)){  
 **try** {  
 result = DBConnection.*checkLogin*(uname, pwd);  
 *//System.out.println("Inside checkCredentials try "+result);* } **catch** (Exception e) {  
 *//System.out.println("Inside checkCredentials catch");* result = **false**;  
 }  
 }**else**{  
 *//System.out.println("Inside checkCredentials else");* result = **false**;  
 }  
   
 **return** result;  
 }  
}

Die URL um einen Benutzer anzumelden stellt sich nun wie folgt zusammen, bzw. wird sie in dieser Reihenfolge abgearbeitet. Gut veranschaulicht wird dies in einem Bild des Tutorials von Android Guru, auch wenn der Code nicht zu 100% meinem Entspricht.

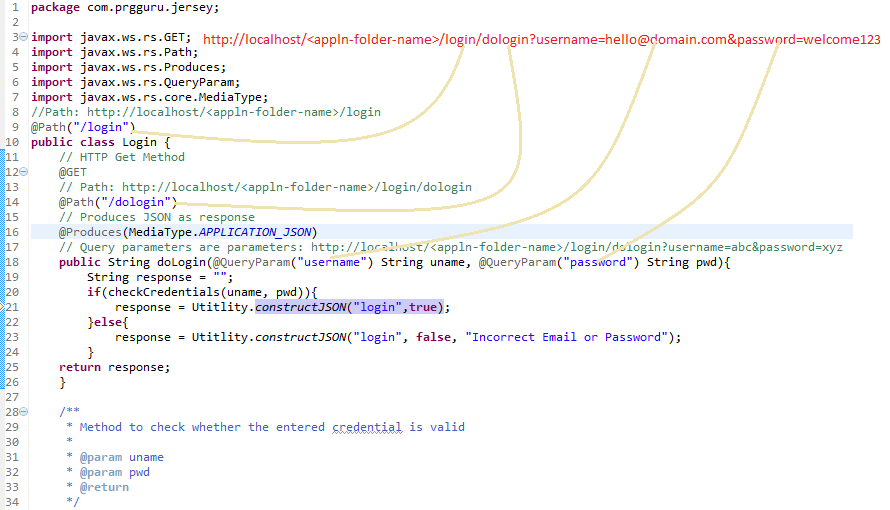


Abbildung : URL Zusammensetzung beim Login [ANDR2]

Ich habe den Inhalt im Gegensatz zur Vorlage folgendermaßen Angepasst. Um der Aufgabenstellung „Bei einem erfolgreichen Login soll eine einfache Willkommensnachricht angezeigt werden.“ Folge zu leisten habe ich die Art des Producers von MediaType.APPLICATION\_JSON auf MediaType.TEXT\_PLAIN geändert. Dies ermöglicht es mir anstatt eines JSON-Objektes einen Einfachen String zurückzugeben. Dadurch kann ich nach einem erfolgreichen Login eine Willkommensnachricht ausgeben. Diese sieht wie folgt aus:

*Login erfolgreich! Willkommen, sie sind als [username] angemeldet!*

Sollte das einloggen Fehlschalgen erscheint folgende Ausgabe:

*Login fehlgeschlagen! Email und Password stimmen nicht ueberein!*

Die URL um einen Benutzer zu registrieren sieht beispielsweiße wie folgt aus:

<http://localhost:8080/WebServicesInJava/ernhofer/login/dologin?username=admin@example.com&password=password>

Bei der Anwendung dieses Links ist es wichtig, dass die IP-Adresse sowie der Port an die Verwendeten Werte angepasst werden.

Um das Projekt Compilier- und Ausführbar zu machen, musste ich anschließend noch Änderungen in der *pom.xml* Datei vornehmen. Wichtig war es dabei keine der verwendeten Ressourcen zu vergessen. Wenn auch nur Eine fehlt funktioniert das Programm entweder gar nicht mehr, oder nicht mehr zuverlässig. Die wichtigsten Inhalte sind die Jersey Abhängigkeiten, sowie das automatische ausführen auf einem Tomcat-Server. Zuletzt genanntes wird durch folgende Codezeilen realisiert. Es handelt sich dabei aber nur um einen Ausschnitt aus der *pom.xml* Datei und nicht um den gesamten Inhalt!

<**build**>  
 <**defaultGoal**>clean install tomcat:run</**defaultGoal**>  
 <**plugins**>  
 <**plugin**>  
 <**groupId**>org.apache.tomcat.maven</**groupId**>  
 <**artifactId**>tomcat7-maven-plugin</**artifactId**>  
 <**version**>2.0</**version**>  
 </**plugin**>  
 <**plugin**>  
 <**artifactId**>maven-war-plugin</**artifactId**>  
 <**version**>2.3</**version**>  
 </**plugin**>  
 </**plugins**>  
 </**build**>

Um eine Verbindung mit der Datenbank aufzubauen war weiters Folgende Ergänzung notwendig:

<**dependency**>  
 <**groupId**>mysql</**groupId**>  
 <**artifactId**>mysql-connector-java</**artifactId**>  
 <**version**>5.1.17</**version**>  
 </**dependency**>

Die gesamte *pom.xml* Datei findet sich in meinem Repository wieder. [REPD]

Nach diesen Schritten habe ich in IntelliJ ein unter Run -> *edit configurations* eine neue Konfiguration zum Ausführen eines Mavenprojektes angelegt. Anschließend habe ich diese gestartet. Wenn alles geklappt hat kann man unter *localhost:8080/WebServicesInJava* folgendes finden:



Abbildung : Startseite des Webservice

Das Registrieren und Einloggen kann mit folgenden Links durchgeführt werden:

<http://localhost:8080/WebServicesInJava/ernhofer/register/doregister?name=Admin&username=admin@example.com&password=password>

<http://localhost:8080/WebServicesInJava/ernhofer/login/dologin?username=admin@example.com&password=password>

## Acceptance Tests

**Startseite**

**ID: S01**

*Situation:*

Ein Benutzer möchte die Startseite des Webservice aufrufen.

*URL, mit der sich der Benutzer einloggen kann:*

[http://localhost:8080/WebServicesInJava](http://localhost:8080/WebServicesInJava/ernhofer/login/dologin?name=David&username=david@gmail.com&password=david)

*Erwartetes Ergebnis:*

Im Browser erscheint die Startseite des Webservice. Diese beinhaltet, mit fetter Schrift, „Web Service by Andi Ernhofer“.

*Tatsächliches Ergebnis:*

Im Browser erscheint eine leere Seite.

*Ergebnis des Tests:*

Der Test ist fehlgeschlagen, da die gewünschte Seite nicht geladen wurde. Die Lösung des Problems ist unter Probleme zu finden. Nach erfolgreicher Behebung des Fehlers ist ein neuer Test durchzuführen.

**ID: S02**

*Situation:*

Ein Benutzer möchte die Startseite des Webservice aufrufen.

*URL, mit der sich der Benutzer einloggen kann:*

[http://localhost:8080/WebServicesInJava](http://localhost:8080/WebServicesInJava/ernhofer/login/dologin?name=David&username=david@gmail.com&password=david)

*Erwartetes Ergebnis:*

Im Browser erscheint die Startseite des Webservice. Diese beinhaltet, mit fetter Schrift, „Web Service by Andi Ernhofer“.

*Tatsächliches Ergebnis:*

Im Browser erscheint die gewünschte Startseite des Webservices.

*Ergebnis des Tests:*

Der Test ist erfolgreich.

**Registrieren**

**ID: R01**

*Situation:*

Ein Benutzer will sich registrieren.  
Er gibt alle angeben an und er ist ein neuer Benutzer.

*Angaben:*

Name: David

Username: [david@gmail.com](mailto:david@gmail.com)

Password: david

*URL um den Benutzer registrieren:*

<http://localhost:8080/WebServicesInJava/ernhofer/register/doregister?name=David&username=david@gmail.com&password=david>

*Erwartetes Ergebnis:*

Da der Benutzer noch nicht in der Datenbank eingetragen ist, und alle erforderlichen Angaben getätigt hat soll ein neuer Benutzer in der Datenbank angelegt werden und eine Erfolgsmeldung im Browser erscheinen.

*Tatsächliches Ergebnis:*

*Ergebnis des Tests:*

Da der Benutzer nicht angelegt wurde sind die Erwartungen erfüllt worden und der Test kann als gelungen gewertet werden.

**ID: R02**

*Situation:*

Ein Benutzer will sich registrieren.  
Es handelt sich um einen neuen Benutzer, jedoch gibt er keinen Namen an.

*Angaben:*

Name:

Username: florian@live.de

Password: esel

*URL um den Benutzer registrieren:*

<http://localhost:8080/WebServicesInJava/ernhofer/register/doregister?username=florian@live.de&password=esel>

*Erwartetes Ergebnis:*

Der Benutzer ist in der Datenbank noch nicht vorhanden. Von dieser Seite sollte es also keine Probleme geben. Jedoch hat der Benutzer keinen Namen angegeben. Daher sollte er nicht registriert werden und es soll kein Datenbankeintrag für ihn angelegt werde.

*Tatsächliches Ergebnis:*

*Ergebnis des Tests:*

Da der Benutzer nicht angelegt wurde sind die Erwartungen erfüllt worden und der Test kann als gelungen gewertet werden.

**ID: R03**

*Situation:*

Ein Benutzer möchte sich registrieren.  
Der Benutzer ist noch nicht vorhanden, jedoch gibt er keine Benutzernamen an.

*Angaben:*

Name: Ferdinand

Username:

Password: fahrrad

*URL um den Benutzer registrieren:*

<http://localhost:8080/WebServicesInJava/ernhofer/register/doregister?name=Ferdinand@live.de&password=fahrrad>

*Erwartetes Ergebnis:*

Da der Benutzer keine Angaben zu seinem Benutzernamen gemacht hat, soll kein Eintrag in der Datenbank, für ihn, erzeugt werden.

*Tatsächliches Ergebnis:*

*Ergebnis des Tests:*

**ID: R04**

*Situation:*

Ein Benutzer möchte sich registrieren.  
Der Benutzer ist noch nicht vorhanden, jedoch gibt er kein Passwort an.

*Angaben:*

Name: Michael

Uname: michi@gmx.at

Password:

*URL um den Benutzer registrieren:*

<http://localhost:8080/WebServicesInJava/ernhofer/register/doregister?name=Michael&username=florian@live.de>

*Erwartetes Ergebnis:*

Da der Benutzer kein Passwort angegeben hat, soll kein Eintrag in der Datenbank erzeugt werden.

*Tatsächliches Ergebnis:*

*Ergebnis des Tests:*

**ID: R05**

*Situation:*

Ein Benutzer möchte sich registrieren.  
Er gibt alle erforderlichen Angaben an, jedoch hat er sich bereits registriert.

*Angaben:*

Name: David

Uname: david@gmail.com

Password: david

*URL um den Benutzer registrieren:*

<http://localhost:8080/WebServicesInJava/ernhofer/register/doregister?name=David&username=david@gmail.com&password=david>

*Erwartetes Ergebnis:*

Die Angaben des Benutzers sind korrekt, jedoch hat er sich bereits registriert und es soll daher nicht noch ein Eintrag von ihm erzeugt werde. Im Browser soll ein Text erscheinen der Angibt, dass sich der Benutzer bereits registriert hat.

*Tatsächliches Ergebnis:*

*Ergebnis des Tests:*

**Login**

**ID: L01**

*Situation:*

Ein Benutzer möchte sich einloggen.

*Angaben:*

Name:

Uname:

Password:

*URL, mit der sich der Benutzer einloggen kann:*

<http://localhost:8080/WebServicesInJava/ernhofer/login/dologin?name=David&username=david@gmail.com&password=david>

*Erwartetes Ergebnis:*

*Tatsächliches Ergebnis:*

*Ergebnis des Tests:*

**ID: L02**

*Situation:*

Ein Benutzer möchte sich einloggen.

*Angaben:*

Name:

Uname:

Password:

*URL, mit der sich der Benutzer einloggen kann:*

<http://localhost:8080/WebServicesInJava/ernhofer/login/dologin?name=David&username=david@gmail.com&password=david>

*Erwartetes Ergebnis:*

*Tatsächliches Ergebnis:*

*Ergebnis des Tests:*

**ID: L03**

*Situation:*

Ein Benutzer möchte sich einloggen.

*Angaben:*

Name:

Uname:

Password:

*URL, mit der sich der Benutzer einloggen kann:*

<http://localhost:8080/WebServicesInJava/ernhofer/login/dologin?name=David&username=david@gmail.com&password=david>

*Erwartetes Ergebnis:*

*Tatsächliches Ergebnis:*

*Ergebnis des Tests:*

**ID: L04**

*Situation:*

Ein Benutzer möchte sich einloggen.

*Angaben:*

Name:

Uname:

Password:

*URL, mit der sich der Benutzer einloggen kann:*

<http://localhost:8080/WebServicesInJava/ernhofer/login/dologin?name=David&username=david@gmail.com&password=david>

*Erwartetes Ergebnis:*

*Tatsächliches Ergebnis:*

*Ergebnis des Tests:*

**ID: L05**

*Situation:*

Ein Benutzer möchte sich einloggen.

*Angaben:*

Name:

Uname:

Password:

*URL, mit der sich der Benutzer einloggen kann:*

<http://localhost:8080/WebServicesInJava/ernhofer/login/dologin?name=David&username=david@gmail.com&password=david>

*Erwartetes Ergebnis:*

*Tatsächliches Ergebnis:*

*Ergebnis des Tests:*

Ein Benutzer will sich einloggen (alle angeben & neuer Benutzer)

Ein Benutzer will sich einloggen (kein Name & schon vorhanden)

Ein Benutzer will sich einloggen (kein uname & schon vorhanden)

Ein Benutzer will sich einloggen (kein passwort & schon vorhanden)

Ein Benutzer will sich einloggen (alle Angaben & schon vorhanden)

**Testergebnisse:**

Erfolgreich

Gescheitert

S01

S02

R01

R02

R03

R04

R05

L01

# Probleme

**Eclipse**

Zu Beginn des Projektes habe ich mit Eclipse gearbeitet. Genauer gesagt mit der JEE Version von Eclipse. Das ich mich für diese Programmierumgebung entschieden habe, lag daran, dass es in dem Tutorial von Androidguru so empfohlen war. Da das Tutorial sehr vielversprechend aussah habe ich diese Empfehlung auch befolgt. Ich habe das gesamte Tutorial Schritt für Schritt abgearbeitet und war überrascht, wie Problemlos alles funktioniert hat. Ich konnte anschließend Benutzer registrieren sowie einloggen. So weit so gut, jedoch musste ich, um die Aufgabenstellung zu erfüllen, ein Build Tool verwenden. Dieses war bis zu diesem Punkt nicht eingebunden. Genau an diesem Punkt begannen jedoch meine Probleme. Nach Stundenlanger Arbeit bin ich von Fehler zu Fehler gewandert und ein war kein Licht am Ende des Tunnels in Sicht. Ich hatte es geschafft ein Projekt anzulegen, welches alle benötigten Daten mittels Maven ladet. Das Einzige, was nicht möglich war, war es das Programm automatisch auf einem Tomcat Server auszuführen. Es war somit also nicht Zufriedenstellend. Nach weiterer Arbeit hatte ich es geschafft ein neues Projekt anzulegen, welches automatisch mittels maven auf einem Tomcat-Server ausgeführt wurde. Sobald ich jedoch mein ursprüngliches Program, mit Register und Anmeldefunktion, auf diesem Server ausführen wollte gab es wieder Probleme. Insgesamt bin ich über Probleme wie beispielsweise Verbindungs-probleme bedingt durch Berechtigungsfehler auf localhost:8080. Ein weiteres Schwerwiegendes Problem war es, dass Eclipse fortan immer meine Java jre Installation verwendet hat. Nach Recherchen im Internet und vielen Einstellungsänderungen in Eclipse und in meinen Umgebungsvariablen war das Problem jedoch immer noch nicht beseitigt. Das konkrete Problem war es, dass Eclipse mitteilte, dass es mit jre meinen Code nicht kompilieren kann, sondern ein jdk dazu braucht. Es war mir nicht geläufig, warum vorherige Projekte, welche ich ausführen konnte, diesen Fehler nicht erzeugten. Schlussendlich habe ich keine Lösung gefunden und nach stundenlanger reseach auch keine Ideen mehr gehabt, was ich noch ausprobieren konnte.

Eine andere Lösung musste her und so bin ich zu dem Entschluss gekommen eine andere Programmierumgebung zu verwenden. Meine Wahl fiel dabei auf IntelliJ. Mittels diesem Programm habe ich es anschließend geschafft alle Probleme, welche ich durch die Verwendung von Eclipse erhalten habe zu beseitigen und ein lauffähiges Programm zu erzeugen.**Vagrant**

Ich habe das Problem, dass ich mittels vagrant kein privates Netzwerk konfigurieren kann. Wenn ich im *Vagrantfile* eine Zeile wie Beispielsweiße config.vm.network "private\_network", ip: "192.168.56.56" schreibe wird automatisch ein neues Hostonly Netzwerk erstellt, welcher eine Adresse von 169.153.\*.\* besitzt. Anschließend erfolgt eine Meldung, dass vagrant keinen privaten Adapter erzeugen konnte. Folgende Fehlermeldung wird ausgegeben:

==> default: Clearing any previously set network interfaces...

There was an error while executing `VBoxManage`, a CLI used by Vagrant

for controlling VirtualBox. The command and stderr is shown below.

Command: ["hostonlyif", "create"]

Stderr: 0%...

Progress state: E\_FAIL

VBoxManage.exe: error: Failed to create the host-only adapter

VBoxManage.exe: error: Operation canceled by the user

VBoxManage.exe: error: Details: code E\_FAIL (0x80004005), component VirtualBox,

interface IVirtualBox

VBoxManage.exe: error: Context: "int \_\_cdecl handleCreate(struct HandlerArg \*,int,int \*)" at line 66 of file VBoxManageHostonly.cpp

Im Internet gibt es dazu jede Menge Tickets. Es scheint als ob es sich dabei um ein bekanntes Problem, bei der Kombination von Vagrant, VirtualBox und Windows10, handelt. Als Lösung wir eine .exe Datei angeboten, welche angeblich alle Probleme beseitigt. Aus Gründen der Ungewissheit, was diese Datei wirklich ausführt und weil ich meinen Rechner noch länger brauche, habe ich auf diesen Lösungsvorschlag verzichtet. Stattdessen hatte ich einen anderen Weg gefunden dieses Problem zu umgehen. Dazu habe ich *vagrant up* ausgeführt. Wie erwartet kam die Fehlermeldung. Die Instanz wird jedoch erstellt. Händisch habe ich einen zusätzlichen Netzwerkadapter hinzugefügt, welcher die VM mittels Host-only Netzwerks mit meinem Host-System verbinden soll.

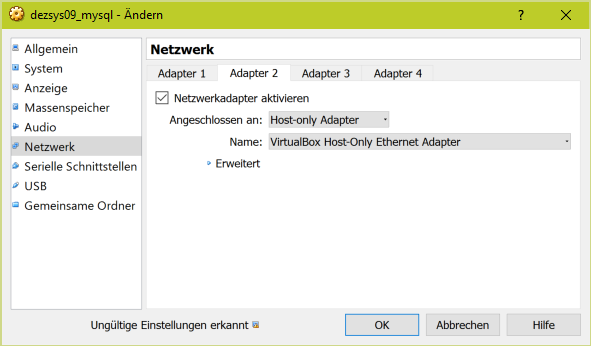


Abbildung 5: Manuelles einfügen eines Adapters

Anschließend habe ich die VM manuell gestartet, wieder beendet und erneut den Befehl *vagrant up* ausgeführt. Obwohl die Netzwerkschnittstellen beim Ausführen dieses Befehls erneut erzeugt werden funktioniert nach dem zweiten Mal alles und die VM wird wie gewünscht erzeugt und gestartet.

Dies ist zwar nicht die optimale Lösung, jedoch funktioniert sie. Zufriedenstellend war sie deshalb aber nicht für mich und so habe ich mich wieder auf die Suche nach anderen Lösungen gemacht. Immerhin war meine Lösung meinerseits und genaugenommen war es keine Lösung, sondern nur ein Weg das Problem zu umgehen. Etwas Besseres habe ich jedoch auch nach weiteren Recherchen nicht gefunden.

Mit dem Stand, welches das Projekt mit dem Datum 16.02.2016 ist es ohne Probleme Möglich die VM mittels *vagrant up* zu starten. Dabei ist es egal ob die Instanz schon vorhanden ist oder ob sie zum ersten Mal erzeugt wird. Meiner Ansicht nach gibt es zwei Möglichkeiten, weshalb es zu Beginn Probleme gab und nun reibungslos Funktioniert. Hypothese 1 ist, dass es durch das einmalige händische Eingeben des zusätzlichen Netzwerkadapters nun immer wieder Funktioniert. Gegen diese Behauptung spricht jedoch, dass ich Anfang jedes Mal, nachdem ich die VM mittels *vagrant destroy* beendet habe und sie mittels *vagrant up* wieder erzeugen wollte, immer beim ersten Mal händische Änderungen vornehmen musste. Die zweite Hypothese, so unglaublich sie auch klingt, besteht darin, dass durch das Hinzufügen von

config.vm.provider "virtualbox" do |v|

v.name = "dezsys09\_mysql"

end

dieses Problem gelöst wurde. Möglicherweise handelt es sich um einen Zufall und etwas Anderes hat das Problem gelöst, jedoch taucht der Fehler nicht mehr auf seit ich diese Zeilen dem Vagrantfile hinzugefügt habe. Alleine diese Zeilen können es jedoch nicht die Lösung sein, da, als ich bei einem Anderen Vagrantfile diese Zeilen eingefügt habe, immer noch das Problem mit dem privaten Netzwerk bestand. Die Lösung hält sich somit immer noch versteckt, jedoch bin ich froh mit den aktuellen Einstellungen eine lauffähige Version zu haben.

**Startseite**

Nachdem das Projekt erstellt war und gestartet werden konnte, wurde meine Startseite nicht angezeigt. Anstatt der gewünschten Anzeige war nur eine leere Seite zu sehen. Die Lösung des Problems bestand darin das url-Pattern in der *web.xml* Datei von /\* auf /ernhofer/\* zu ändern.

**Login**

Nach dem fertigen Konfigurieren des Projektes war es möglich Benutzer anzumelden, jedoch bekam ich jedes Mal einen Fehler, wenn ich einen Benutzer anmelden wollte. Nach langer Fehlersuche in meinem Code hatte ich nicht verstanden, weshalb beim Registrieren alles klappt, beim Login jedoch immer Fehler auftreten. Das Registrieren eines Benutzers war möglich und wenn sich ein Benutzer Registrieren wollte, welcher schon vorhanden war, wurde dies auch mit der richtigen Fehlermeldung verhindert. Nach mehreren Kontrollen in der Datenbank und bei der Werten des anzumeldenden Benutzers hatte ich den Fehler gefunden. Aus einem Vermutlichen Copy-Paste Fehlers hatte sich bei dem Passwort-Eintrag in der Datenbank am Ende des Passwortes ein „.“ angefügt. Nach Löschen und erneut anlegen des Benutzers war das Problem behoben und das Registrieren sowie Einloggen von Benutzern war möglich.

**Maven-Vagrant**

Ich wollte es ermöglichen, dass automatisch eine VM erzeugt/gestartet wird, wenn das Projekt gestartet wird. Dazu habe ich ein maven-plugin gefunden. [MAVV] Dieses habe ich in meine pom.xml aufgenommen und zu den default Goals *vagrnat:up* hinzugefügt. Dabei sind mir anschließend gleich 2 Probleme aufgetreten. In fand keine Möglichkeit diesem Plugin meinen Pfad zu meinem Vagrantfile anzugeben. Dieses Problem war jedoch für Probezwecke nicht wirklich schwerwiegend, und konnte durch einfaches Kopieren der Vagrant Daten in den Root Ordner des Projektes behoben werden. Das zweite Problem war jedoch nicht so leicht zu Lösen. Beim ausführen des Befehls *vagrant:up* erschien folgende Fehlermeldung:

Failed to execute goal net.ju-n.maven.plugins:vagrant-maven-plugin:1.0.1:up.

Der Grund, dass es nicht ausgeführt werden konnte war:

NoMethodError: undefined method `configure' for Vagrant:Module

Nach Recherchen im Internet habe ich folgende Lösung für dieses Problem gefunden[LOEV] Diese besagt, dass mit der Vagrant Version 1.0.1 keine Vagrant.config möglich ist. Als Lösung wird geraten eine neuere Vagrat Version zu verwenden. Normalerweiße kein Problem. Einfach eine neuere Version installieren und die Sache ist erledigt. Da ich jedoch mit dem Maven Plugin arbeite ist es nicht so einfach. Ich bin an die Version des Plugins angewiesen. Eine neuere (genau genommen gar keine andere) Version habe ich jedoch nicht gefunden und somit konnte das Problem nicht gelöst werden. Die einzige Möglichkeit, welche halbwegs akzeptabel ist, ist es zuerst manuell die vagrant VM zu erzeugen/starten und anschließend das Projekt mittels maven zu generieren.

# Zeitaufzeichnung

|  |  |
| --- | --- |
| Tätigkeit | Dauer |
| Mittels Vagrant Datenbank zur Verfügung Stellen | 120min |
| Aufsetzen einer Webservice-Schnittstelle | 120min |
| Registrierung von Benutzern mit entsprechender Persistierung | 30min |
| Login und Rückgabe einer Willkommensnachricht | 60min |
| AcceptanceTests |  |
| Problembearbeitung und Fehlerbehebung | 690min |
| Protokollierung | 240min |
| Gesamt | 20h |

# Quellen

[ANDR1]"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/>   
zuletzt besucht 15.02.2016

[ANDR2]"Android Restful Webservice Tutorial – Introduction to RESTful webservice – Part 2"; Posted By Android Guru on May 11, 2014; online:  
<http://programmerguru.com/android-tutorial/android-restful-webservice-tutorial-how-to-create-restful-webservice-in-java-part-2/>   
zuletzt besucht 17.02.2016

[RESJ] "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>   
zuletzt besucht 15.02.2016

[HERM] "Heroku makes it easy to deploy and scale Java apps in the cloud"; online: <https://www.heroku.com/>  
zuletzt besucht 15.02.2016

[VAGM] “Vagrant Vorlage für eine VM mit MySQL”

<https://github.com/AlexDisler/mysql-vagrant>

zuletzt besucht 16.02.2016

[DOCJ] „Jersey Dokumentation“  
<https://jersey.java.net/documentation/latest/user-guide.html>  
zuletzt besucht 16.02.2016

[REPD] „Git-Repository des Projektes“

<https://github.com/aernhofer-tgm/DEZSYS09/>

zuletzt besucht 17.02.2016

[MAVV] „Vagrant Maven plugin“

[http://nicoulaj.github.io/vagrant-maven-plugin/#](http://nicoulaj.github.io/vagrant-maven-plugin/)

zuletzt besucht 17.02.2016

[LOEV]“undefined method configure for Vagrant:Module”

<http://www.fvue.nl/wiki/Vagrant:_Vagrantfile:4:_undefined_method_configure_for_Vagrant:Module>

zuletzt besucht 17.02.2016