

**Laborprotokoll**

**CORBA Overview**

**Systemtechnik Labor**

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

**Manuel Reiländer**

|  |  |
| --- | --- |
|  | **Version 1.0** |
| **Note:** | **Begonnen am 16. April 2016** |
| **Betreuer: Michael Borko** | **Beendet am 17. April 2016** |

Inhaltsverzeichnis

[1Einführung 3](#__RefHeading___Toc105_1963609346)

[1.1Ziele 3](#__RefHeading___Toc107_1963609346)

[1.2Voraussetzungen 3](#__RefHeading___Toc109_1963609346)

[1.3Aufgabenstellung 3](#__RefHeading___Toc111_1963609346)

[2Ergebnisse 4](#__RefHeading___Toc113_1963609346)

# 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+2) 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 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 (Acceptance-Tests bez. aller funktionaler Anforderungen mittels JUnit) dokumentiert werden. Es muss noch keine grafische Oberfläche implementiert werden! Verwenden Sie auf jeden Fall ein gängiges Build-Management-Tool (z.B. Maven). Dabei ist zu beachten, dass ein einfaches Deployment möglich ist (auch Datenbank mit z.B. file-based DBMS).

# Ergebnisse

Vorweg, der Restful Webservice wurde nicht von mir implementiert, sondern stammt vom lieben Kollegen Hr. Paul Kalauner. Lediglich die Unit Tests, sowie dieses Protokoll wurden von mir gemacht.

## Testing eines Restful Webservice

Um einen Restful Webservice zu testen bedarf es einer Bibliothek um Httpd Requests an den Webservice zu senden. Apache bietet hier eine leicht zu implementierende Bibliothek an (Apache Http Client).

Als erstes muss die Gradle dependency zur *build.gradle* hinzugefügt werden

* org.apache.httpcomponents:httpclient:4.3.4

## Verwendung Code example

Zur einfachen Erklärung wurde hier eine Rest Schnittstelle verwendet, womit man Benutzer in eine Datei basierte Datenbank abspeichern kann. Zum Registrieren eines Benutzers dient die Schnittstelle */register* wobei die Parameter (*email*, *name* und password) per HTTP-Post weitergegeben werden.

try {  
 CloseableHttpClient httpclient = HttpClients.*createDefault*();  
 // Parameter wie Protokoll, host, port setzen  
 URI uri = new URIBuilder()  
 .setScheme(*protocol*)  
 .setHost(*host*)  
 .setPort(*port*)  
 .setPath("/register")  
 .build();  
 HttpPost post = new HttpPost(uri);  
 // Content type auf json setzen  
 post.setHeader(HttpHeaders.*CONTENT\_TYPE*, "application/json");  
 // POST Parameter setzen  
 List<NameValuePair> nvps = new ArrayList<>();  
 nvps.add(new BasicNameValuePair("name", "test"));  
 nvps.add(new BasicNameValuePair("email", "test"));  
 nvps.add(new BasicNameValuePair("password", "test"));  
 post.setEntity(new UrlEncodedFormEntity(nvps));  
 // POST Request senden  
 httpclient.execute(post);  
} catch (URISyntaxException e) {  
 e.printStackTrace();  
} catch (ClientProtocolException e) {  
 e.printStackTrace();  
} catch (IOException e) {  
 e.printStackTrace();  
}

Da dieser Code zum Testen jedoch sehr unübersichtlich ist, ist der Apache http client nicht geeignet.

## Springs testing Framework

Wesentlich geeigneter ist das hauseigene Testing Framework von spring. Durch dieses wird das Senden eines Requests deutlich vereinfacht, da man den Request nicht mehr selbst „bauen“ muss, sondern einfach Entities verwenden kann. Dies wird im Folgenden anhand eines Code Beispiels veranschaulicht:

User user = new User("user1@test.at", "user1", "secret");  
this.restTemplate

.postForEntity("http://localhost:8080/register", user, String.class);

Dies ist deutlich einfacher zu verwenden, übersichtlicher und verringert den Code deutlich.

## Test Datenbank „aufsetzen“

Um für das Testing eine saubere Datenbank zu benutzen, muss diese in der *application.properties* Datei konfiguriert werden.

spring.datasource.driver-class-name=org.h2.Driver  
spring.datasource.url = jdbc:h2:file:./UserDB\_test;FILE\_LOCK=FS  
spring.datasource.username = dezsys10  
spring.datasource.password = dezsys10  
spring.jpa.hibernate.ddl-auto = update

## Saubere Datenbank

Um Fehler bei den Tests zu vermeiden, ist es sinnvoll die Datenbank vor jedem Test zu „säubern“. Dazu ist es notwendig bei einer Dateibasierten Datenbank, das Datenbankfile zu löschen. Dies kann wie folgt passieren:

@BeforeClass  
public static void beforeClass() {  
 String current = System.*getProperty*("user.dir");  
 String embeddedDatabaseFilename = "UserDB\_test.mv.db";  
 if(new File(current + "\\" + embeddedDatabaseFilename).delete())  
 Application.*main*(new String[]{"--spring.profiles.active=test", "--server.port=" + *port*});  
 else  
 System.*out*.println(String.*format*("Could not delete or find embedded database file %s", embeddedDatabaseFilename));  
}

## Testing

Bei einer Restschnittstelle werden zusätzlich zu Error Messages, http Status codes zurückgegeben, falls irgendetwas bei einem Request schiefgeht. Nun vergleicht man einfach den zurückgegebenen Status Code und falls erwünscht die Error Message mit dem erwarteten Status Code/Error Message und kann so Unit Tests schreiben. Der Folgende Code zeigt einen Acceptance Test, wen ein User doppelt angelegt wird:

@Test  
public void testUserAlreadyExists() {  
 User user = new User("user1@test.at", "user1", "secret");  
 this.restTemplate.postForEntity(*url* + "/register", user, String.class);  
 try {  
 this.restTemplate.postForEntity(*url* + "/register", user, String.class);  
 } catch (HttpClientErrorException e) {  
 *assertEquals*("User " + user.getEmail() + " already exists!", e.getResponseBodyAsString());  
 *assertEquals*(HttpStatus.*BAD\_REQUEST*, e.getStatusCode());  
 }  
}

# Probleme

## POST Request ende in Exception beim Webservice

CloseableHttpClient httpclient = HttpClients.*createDefault*();  
// Parameter wie Protokoll, host, port setzen  
URI uri = new URIBuilder()  
 .setScheme(*protocol*)  
 .setHost(*host*)  
 .setPort(*port*)  
 .setPath("/register")  
 .build();  
HttpPost post = new HttpPost(uri);   
// POST Parameter setzen  
List<NameValuePair> nvps = new ArrayList<>();  
nvps.add(new BasicNameValuePair("name", "test"));  
nvps.add(new BasicNameValuePair("email", "test"));  
nvps.add(new BasicNameValuePair("password", "test"));  
post.setEntity(new UrlEncodedFormEntity(nvps));  
// POST Request senden  
CloseableHttpResponse response = httpclient.execute(post);  
response.close();

Nachdem mit diesem Code ein Request an den Webservice gesendet wurde, endete dies in einer Exception auf dem Restful Webservice mit dem Inhalt:

A servlet request to the URI <http://localhost:8080/register> contains form parameters in the request body but the request body has been consumed by the servlet or a servlet filter accessing the request parameters. Only resource methods using @FormParam will work as expected. Resource methods consuming the request body by other means will not work as expected. UT005023: Exception handling request to /error

Durch das setzen des Content types auf „application/json“, kann dieser Fehler behoben werden.

Fehlerbeschreibung und Lösung kann hier nachgelesen werden

* <http://stackoverflow.com/questions/36666834/apache-http-request-ends-in-exception-on-restful-webservice?noredirect=1#comment60925612_36666834>

(mein eigener Thread)