
GK9_3
Cloud-Datenmanagement

Systemtechniklabor
5BHIT 2017/18

Johannes Bishara

Note:
Betreuer: Michael Borko

Version 1.1
Begonnen am 5. Oktober 2017
Beendet am 19. Oktober 2017

Inhaltsverzeichnis

1	Einführung	1
1.1	Ziele	1
1.2	Vorraussetzungen	1
1.3	Aufgabenstellung	1
2	Ergebnisse	2
2.1	Projekt einrichten	2
2.1.1	Programm erstellen	2
2.1.2	pom.xml	2
2.2	Register	3
2.2.1	register() Methode	3
2.2.2	submit() Methode	4
2.3	Login	4
2.3.1	login() Methode	4
2.3.2	submit() Methode	4
2.4	DB	5
2.4.1	Attribute	5
2.4.2	getInstance()	5
2.4.3	createCon()	5
2.4.4	closeCon()	6
2.4.5	createTable()	6
2.4.6	registerUser()	6
2.4.7	loginUser()	7
2.5	Testing	8
2.6	Demo	8
2.6.1	Register	8
2.6.2	Login	10

1 Einführung

Diese Übung zeigt die Anwendung von mobilen Diensten.

1.1 Ziele

Das Ziel dieser Übung ist eine Webanbindung zur Benutzeranmeldung umzusetzen. Dabei soll sich ein Benutzer registrieren und am System anmelden können.

Die Kommunikation zwischen Client und Service soll mit Hilfe einer REST Schnittstelle umgesetzt werden.

1.2 Voraussetzungen

- Grundlagen einer höheren Programmiersprache
- Verständnis über relationale Datenbanken und dessen Anbindung mittels ODBC oder ORM-Frameworks
- Verständnis von Restful Webservices

1.3 Aufgabenstellung

Es ist ein Webservice 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 (Acceptance-Tests bez. aller funktionaler Anforderungen mittels Unit-Tests) dokumentiert werden. Verwenden Sie auf jeden Fall ein gängiges Build-Management-Tool (z.B. Maven oder Gradle). Dabei ist zu beachten, dass ein einfaches Deployment möglich ist (auch Datenbank mit z.B. file-based DBMS).

2 Ergebnisse

2.1 Projekt einrichten

Zu Beginn habe ich das Tutorial mit Jersey und JAX-RS umgesetzt, um eine Basiskonfiguration zu erhalten, auf die ich aufbauen kann. Folgende Schritte waren notwendig:

2.1.1 Programm erstellen

Zuerst erstellt man ein 'Dynamic Web Project'. Dabei wählt man als Target Runtime Apache Tomcat v9.0. Da bei der Erstellung des Projekts kein web.xml File vorhanden war, habe ich einen Deployment Descriptor Stub generieren lassen.

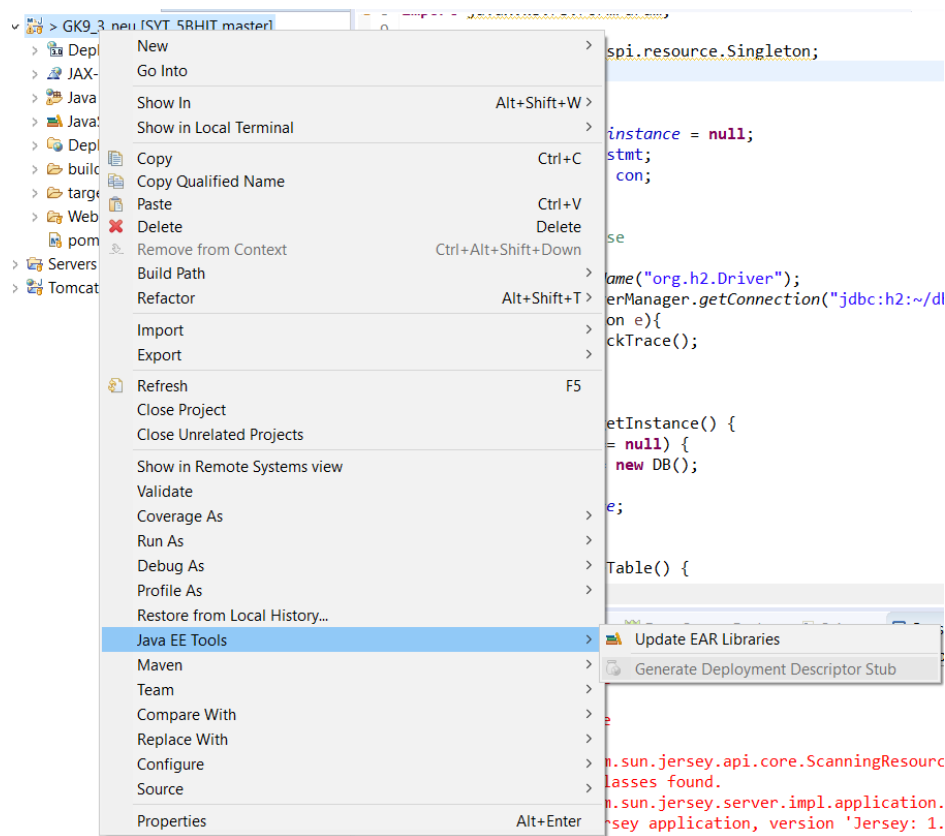


Abbildung 1: Generate Deployment Descriptor Stub

2.1.2 pom.xml

Nun konvertiert man das Projekt in ein Maven-Projekt und added folgende Dependencies in das pom.xml File:

- asm.jar
- jersey-bundle.jar

- json.jar
- jersey-server.jar
- junit
- maven

Hier ein Ausschnitt wie man eine Dependencie in ein pom.xml hinzufügt.

```
<dependency>
  <groupId>com.sun.jersey</groupId>
  <artifactId>jersey-bundle</artifactId>
  <version>1.19.4</version>
</dependency>
```

Abbildung 2: Dependency-Struktur in pom.xml

2.2 Register

Beim Register habe ich als Path /register gemäß der Aufgabenstellung angegeben.

2.2.1 register() Methode

In der Klasse habe ich die Methode register() geschrieben, welche das Registrierungsformular zur Verfügung stellt. Wichtig ist dabei die @Produces Annotation. Diese sagt aus, dass der Rückgabewert dieser Methode ein HTML ist. Die @GET Annotation besagt, dass die Parameter über die URL die Parameter übergeben werden. Die Klasse liefert ein String zurück, indem das HTML für die Seite dabeisteht.

```
1 @GET
2 @Produces(MediaType.TEXT_HTML)
3 public String register() throws Exception {
4
5     return "<html>\r\n" +
6           "<head><meta charset=\"UTF-8\"></head>\r\n" +
7           "<body>\r\n" +
8           "<form method=\"post\">\r\n" +
9           "  First name:<br>\r\n" +
10          "<input type=\"text\" name=\"firstname\"><br>\r\n" +
11          "  Last name:<br>\r\n" +
12          "<input type=\"text\" name=\"lastname\"><br>\r\n" +
13          "  E-Mail:<br>\r\n" +
14          "<input type=\"text\" name=\"mail\"><br>\r\n" +
15          "  Password:<br>\r\n" +
16          "<input type=\"password\" name=\"pw\">\r\n" +
17          "<input type=\"submit\" value=\"Submit\">\r\n" +
18          "</form>\r\n" +
19          "</body>\r\n" +
20          "</html>\r\n" +
21          ";"
22 }
```

2.2.2 submit() Methode

Die submit() Methode hat eine @POST Annotation, das ist wichtig, da sonst die Parameter im Klartext in der URL angezeigt werden.

Diese Methode wird aufgerufen, wenn der User auf den Submit-Button drückt. Es werden alle Inhalte der Felder aus dem Formular genommen und als Parameter an die Methode übergeben. Es wird eine Verbindung zur Datenbank aufgebaut, ein User-Table created falls es noch nicht existiert und zu guter Letzt der User in der Datenbank angelegt. Am Ende wird die Verbindung wieder geschlossen. Es wird wieder ein String returned, der die HTML-Page beinhaltet. Bei erfolgreicher Registrierung ist die Nachricht "Registrierung erfolgreich" zu sehen. Andernfalls "Registrierung nicht erfolgreich".

```

1  @POST
2  @Produces(MediaType.TEXT_HTML)
  public String submit(@FormParam("firstname") String firstname, @FormParam("lastname") String lastname
    , @FormParam("mail") String mail, @FormParam("pw") String pw)
    throws Exception {
4      String response = "";

6      db.getInstance().createCon();
      db.getInstance().createTable();
8      response = db.getInstance().registerUser(firstname, lastname, mail, pw);
      db.getInstance().closeCon();
10
12      return response;
    }

```

2.3 Login

Beim Register ist der Path /login. Generell kann man sagen, dass der Login gleich aufgebaut ist wie die Klasse Register.

2.3.1 login() Methode

Die login()-Methode sieht exakt gleich aus, wie die login-Methode von Register. Einzig und allein der String ist anders, da natürlich die HTML-Page anders aufgebaut ist.

```

2  @GET
  @Produces(MediaType.TEXT_HTML)
  public String login() throws Exception {
4
6      return "<html>\r\n" + "<head><meta charset=\"UTF-8\"></head>\r\n" + "<body>\r\n" + "<form method"
    = "<post>\r\n"
    + "    E-Mail:<br>\r\n" + "    <input type=\"text\" name=\"mail\"><br>\r\n" + "    Password:<br>\r\n"
    + "    <input type=\"password\" name=\"pw\">\r\n" + "    <input type=\"submit\" value=\""
    + "    Submit\">\r\n"
    + "</form>\r\n" + "</body>\r\n" + "</html>\r\n" + "";
8  }

```

2.3.2 submit() Methode

Bei der submit-Methode werden wie bei Register, die Inhalte der Textfelder der HTML-Website als Parameter angegeben. Es wird anschließend eine Verbindung zur Datenbank hergestellt, der

User-Table erstellt falls er nicht existiert und danach abgeglichen, ob die eingetragene E-Mail inklusive Passwort mit einem Datensatz übereinstimmt. Falls ja, wird man auf eine HTML-Website weitergeleitet, auf der eine Willkommensnachricht angezeigt wird.

```

1  @POST
   @Produces(MediaType.TEXT_HTML)
3  public String submit(@FormParam("mail") String mail, @FormParam("pw") String pw) throws Exception {
   String response = "";
5   db.getInstance().createCon();
   db.getInstance().createTable();
7   response = db.getInstance().loginUser(mail, pw);
   db.getInstance().closeCon();
9
   return response;
11 }

```

2.4 DB

Als Datenbank wird H2 verwendet. Um sicherzustellen, dass die Datenbank nur einmal geöffnet wird, habe ich Singleton verwendet um sicherzugehen, dass nur eine Instanz der Datenbank existiert.

2.4.1 Attribute

Die Klasse DB hat 3 Attribute. Ein statisches DB-Objekt, um sicherzustellen, dass nur eine Instanz verwendet wird. Ein privates Statement um später SQL-Befehle an die Datenbank senden zu können und eine private Connection, um eine Verbindung zur Datenbank zu ermöglichen.

```

1  private static DB instance = null;
   private Statement stmt;
3  private Connection con;

```

2.4.2 getInstance()

Diese Methode überprüft ob eine Instanz von DB existiert. Falls nicht wird eine neue erstellt und diese zurückgegeben. Damit man Befehle an verschiedenen Stellen des Programms immer auf derselben Instanz aufrufen kann, sollte man immer über diese Instanz arbeiten.

```

1  public static DB getInstance() {
   if (instance == null) {
3     instance = new DB();
   }
5  return instance;
   }

```

2.4.3 createCon()

In dieser Methode wird die Verbindung zum Server hergestellt. Bevor man also einen SQL-Befehl absetzen kann, muss man immer zuerst diese Methode aufrufen um eine Verbindung herzustellen.

```

   public void createCon() throws Exception {
2     Class.forName("org.h2.Driver");
   con = DriverManager.getConnection("jdbc:h2:~/dbuser", "sa", "");
4  }

```

2.4.4 closeCon()

Hier wird lediglich die Verbindung zur Datenbank geschlossen, damit sich gegebenenfalls andere User auf die Datenbank zugreifen können.

```

2      public void closeCon() throws Exception {
        con.close();
      }

```

2.4.5 createTable()

Um einen User in die Datenbank einspeichern zu können, muss die Tabelle User erstellt werden. Dies wird in dieser Methode abgearbeitet.

```

1      public void createTable() throws Exception {
        stmt = con.createStatement();
3      stmt.executeUpdate(
            "CREATE TABLE IF NOT EXISTS users ( vname varchar(255), lname varchar(255), mail varchar
              (255) primary key, password                                varchar(255))");
5      }

```

2.4.6 registerUser()

Diese Methode hat den Vornamen, Nachnamen, Email und Passwort als Parameter. Der Rückgabewert dieser Methode ist ein String, welches die HTML-Response beinhaltet. Somit kann auf eine fehlerhafte Eingabe reagiert werden. Fehlerhafte Eingaben sind wenn nicht alle Felder ausgefüllt wurden, bzw. der Primary Key des Tables (E-Mail) bereits einen Datenbankeintrag hat.

```

1      public String registerUser(String firstname, String lastname, String mail, String pw) {
        String statement = "HTML";
3      boolean error = false;
        try {
5      if((firstname.length() == 0) || (lastname.length() == 0) || (mail.length() == 0) || pw.
            length() == 0) {
                statement = "<html>\r\n" +
7      "    <head>\r\n" +
                "      <meta charset='UTF-8'>\r\n" +
                "    <body>\r\n" +
9      "      <h1> Alle Felder sind Pflichtfelder</h1>\r\n" +
                "    </body>\r\n" +
11     "  </html>\r\n" +
                "  ";
13     error = true;
        }

15     if(error == false) {
17     stmt = con.createStatement();
            stmt.executeUpdate( "INSERT INTO users VALUES ( '"+firstname+" ', '"+lastname+" ', '"+mail+"
              ' , '"+pw+" ' ) ");
19     }

21     }catch(Exception e) {
        //e.printStackTrace();
23     statement = "<html>\r\n" +
                "    <head>\r\n" +
                "      <meta charset='UTF-8'>\r\n" +
25     "    <body>\r\n" +
                "      <h1> Registrierung nicht erfolgreich</h1>\r\n" +
27     "    </body>\r\n" +
                "  </html>\r\n" +
29     "  ";
        }
31     return statement;
      }

```


2.4.7 loginUser()

Als Parameter sind die Email und das Passwort gesetzt. Die Methode überprüft nun, ob es einen User mit der Kombination aus Mail-Adresse und Passwort in der Datenbank vorhanden ist. Ist ein Datensatz vorhanden, wird ein String mit einer HTML-Seite returned, in dieser ist eine Willkommensnachricht enthalten. Der User wird mit seinem Namen begrüßt. Falls kein Datensatz vorhanden ist, wird die Nachricht returned, dass der Login fehlgeschlagen ist.

```

2  public String loginUser(String mail,String pw) {
    String statement= "HTML";
    try {
4      stmt = con.createStatement();
      ResultSet rs= stmt.executeQuery( "SELECT password FROM users WHERE mail = '"+mail+"'");
6      if(rs.next()) {
          String password = rs.getString("password");
8          if(password.equals(pw)) {
              String vname = "";
              String lname = "";

10             ResultSet vnameRs = stmt.executeQuery("SELECT vname FROM users WHERE mail = '"+mail+"
                "'");
12             if(vnameRs.next()) {
                 vname = vnameRs.getString("vname");
14             }
              ResultSet lnameRs = stmt.executeQuery("SELECT lname FROM users WHERE mail = '"+mail+"
                "'");
16             if(lnameRs.next()) {
                 lname = lnameRs.getString("lname");
18             }

20             System.out.println("Password = Pw");
22             statement = "<html>\r\n" +
                "<head>meta charset=\\\"UTF-8\\\"></head>\r\n" +
24                 "<body>\r\n" +
                "<h1>Login erfolgreich</h1><br>\r\n" +
26                 "Hallo <b>" + vname + " " + lname + " </b>!\r\n" +
                "</body> \r\n" +
28                 "</html> ";
            }
30         }
    } catch (Exception e) {
32         e.printStackTrace();
    }
34     return statement;
}

```

2.5 Testing

Zum Testen habe ich JUnit verwendet. Da ein funktionaler Test erfolgen soll, habe ich zugleich darauf geachtet eine hohe Coverage zu erzielen. Es wurden die wichtigsten Funktionalitäten auf verschiedene Szenarien getestet. Um überhaupt im Project testen zu können, muss man zuerst JUnit im pom.xml als dependency angeben. Danach kann man im Testfolder eine Klasse erstellen und diese dann als JUnitTest starten. Insgesamt wurden 19 Testcases geschrieben, welche die Funktionalität abdecken.

```

1      <dependency>
2          <groupId>junit</groupId>
3          <artifactId>junit</artifactId>
4          <version>4.11</version>
5          <scope>test</scope>
6      </dependency>

```

Element	Coverage	Covered Instru...	Missed Instruct...	Total Instructio...
✓ GK9_3_neu	92,3 %	526	44	570
✓ src	92,3 %	526	44	570
✓ bishara	88,3 %	218	29	247
> DB.java	85,4 %	170	29	199
> Login.java	100,0 %	23	0	23
> Register.java	100,0 %	25	0	25

Abbildung 3: Coverage

2.6 Demo

2.6.1 Register

Hier der Ablauf einer erfolgreichen Registrierung:

localhost:8080/GK9_3_neu x

localhost:8080/GK9_3_neu/bishara/register

Apps ★ Bookmarks Outlook Web App eLearning

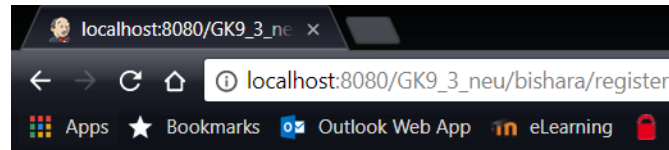
First name:

Last name:

E-Mail:

Password:

Abbildung 4: Registrierung Hansi Hinterseer



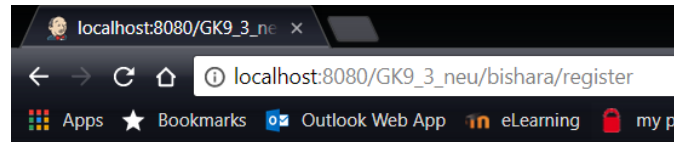
Registrierung erfolgreich

Abbildung 5: Registrierung Hansi Hinterseer erfolgreich

Hier der Ablauf einer nicht erfolgreichen Registrierung:

A screenshot of a web browser window showing a registration form. The address bar shows 'localhost:8080/GK9_3_neu/bishara/register'. The browser's toolbar includes back, forward, refresh, and home buttons, along with a search icon. Below the toolbar, there are links for 'Apps', 'Bookmarks', 'Outlook Web App', and 'eLearning'. The main content area displays a registration form with the following fields and labels: 'First name:' with a text box containing 'Hansi2'; 'Last name:' with a text box containing 'Hinterseer2'; 'E-Mail:' with a text box containing 'hansi@mail.at'; and 'Password:' with a text box containing three dots. To the right of the password field is a 'Submit' button.

Abbildung 6: Registrierung Hansi2 Hinterseer2



Registrierung nicht erfolgreich

Abbildung 7: Registrierung Hansi2 Hinterseer2 nicht erfolgreich

2.6.2 Login

Hier der Ablauf eines erfolgreichen Logins:

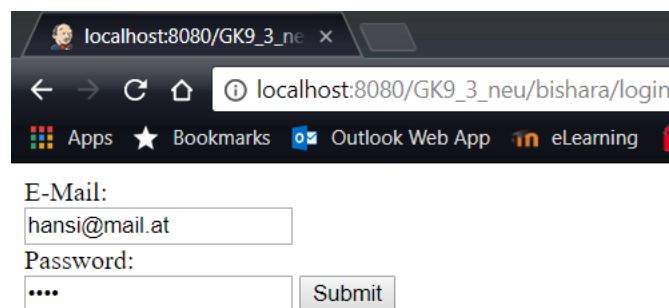
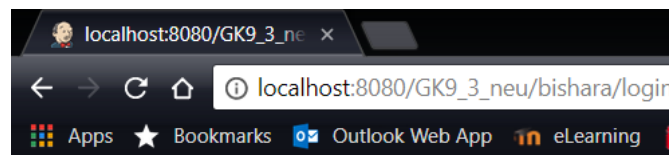


Abbildung 8: Login Hansi Hinterseer

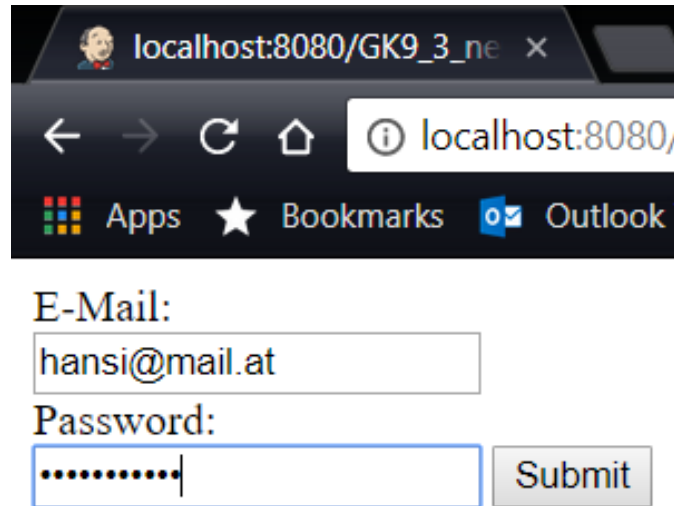


Login erfolgreich

Hallo **Hans Hinterseer** !

Abbildung 9: Hansi Hinterseer Willkommensnachricht

Hier der Ablauf eines nicht erfolgreichen Logins:



localhost:8080/GK9_3_ne x

← → ↻ 🏠 ⓘ localhost:8080/

Apps ★ Bookmarks Outlook

E-Mail:

Password:

Abbildung 10: Login Hansi Hinterseer

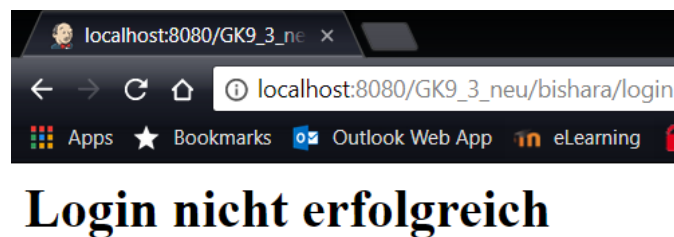


Abbildung 11: Login Hansi Hintersee nicht erfolgreich

Tabellenverzeichnis

Listings

Abbildungsverzeichnis

1	Generate Deployment Descriptor Stup	2
2	Dependency-Struktur in pom.xml	3
3	Coverage	8
4	Registrierung Hansi Hinterseer	8
5	Registrierung Hansi Hinterseer erfolgreich	9
6	Registrierung Hansi2 Hinterseer2	9
7	Registrierung Hansi2 Hinterseer2 nicht erfolgreich	9
8	Login Hansi Hinterseer	10
9	Hansi Hinterseer Willkommensnachricht	10
10	Login Hansi Hinterseer	11
11	Login Hansi Hintersee nicht erfolgreich	11