JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Laborprotokoll
Web-Services in Java

Systemtechnik Labor
5BHITT 2015/16, Gruppe Y

Martin Kritzl

Version 1.0

Note:

Betreuer: Michael Borko

Begonnen am 11. März 2016

Beendet am 31. März 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+2) 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 (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).

Bewertung: 16 Punkte
- 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)

2 Ergebnisse

Unter folgendem Link ist das Repository dieser Übung einzusehen:
https://github.com/mkritzl-tgm/DezSys09-Web-Services

2.1 Designüberlegung

2.1.1 Package „data“

Hier sind alle Klassen vorhanden die Daten darstellen und damit persistiert oder übertragen werden.

2.1.2 Package „endpoints“

In diesem Package sind alle Endpoints der Rest-Schnittstelle enthalten. Dies beinhaltet das Registrieren und den Login.

2.1.3 Package „Persistance“

Hier ist das Repository für die Kommunikation mit der Datenbank vorhanden.

2.1.4 Package „config“

In diesem Package sind alle Konfigurationen für die Datenbank, sowie die Definition der verschiedenen Endpoints die verwendet werden sollen, angegeben.

2.1.5 Package „utils“

Hier werden Klassen abgelegt, die für einen einfacheren Umgang mit den Benutzerdaten sorgen. Zum Beispiel die Überprüfung der Benutzerdaten.

2.2 Arbeitsdurchführung

2.2.1 Erstellen des Useraccounts

Der User soll in weiterer Folge nicht nur temporär als Objekt vorhanden sein, sondern ebenso in der Datenbank persistiert werden können. Aus diesem Grund ist die Klasse mit der Annotation „`@Entity`" versehen worden.

Als Primären Schlüssel hat der Benutzer eine E-Mail-Adresse. Diese wird durch das Hinzufügen von „`@Id`" zum Primary-Key. Als zweites Attribut ist das Passwort vorhanden. Beide haben ebenso eine Eingeschränkte Größe („`@Size`") und dürfen nicht null („`@NotNull`") sein.

```
@Id
@Size(max = 50)
@NotNull
@Email
private String email;

@NotNull
@Size(min=5, max = 50)
private String password;
```

Der Konstruktor ist speziell für die Deserialisierung der Client-Anfragen geschrieben. Dadurch enthalten die Paramater die Annotation „`@JsonProperty(<Key>)`", um diese zuordnen zu können.

```
public UserAccount(@JsonProperty("email") String email,
@JsonProperty("password") String password) {
…
}
```

2.3 Konfiguration von Jersey

Um Jersey mit Spring verwenden zu können, müssen Deklarationen getätigt werden. Dazu zählt unter anderem die Angabe der Endpoints, als auch die Aktivierung der Beanvalidation.

```
@Configuration
public class JerseyConfiguration extends ResourceConfig {
    public JerseyConfiguration() {
        this.property(ServerProperties.BV_SEND_ERROR_IN_RESPONSE, true);
        this.register(UserEndpoint.class);
    }
}
```

2.4 RESTful Endpoints

Die Endpoints stellen die Punkte dar, von denen von außen zugegriffen werden kann. Angegeben wird der Typ, der generiert werden soll. Dies wird mit der Annotation „`@Produces`“ umgesetzt.

```
@Path("/")
@Produces({MediaType.APPLICATION_JSON})
@Named
public class UserEndpoint {
```

Um auf die User-Objekte der Datenbank zugreifen zu können, wird ein Attribut für das UserRepository angelegt. Dieses wird mittels Dependency-Injection ermöglicht.

```
@Autowired
private UserRepository repository;
```

Im Folgenden wurde sowohl „`/register`“ als auch „`/login`“ als Endpoint definiert.

2.4.1 Registrierung

Die Registrierung geschieht über die URL „`/register`“. Dazu muss der geeignete Body mitgeliefert werden. Dieser sieht wie folgt aus:

```
{
      „email“: „<E-Mail>“,
      „password“: „<Passwort>“
}
```

Umgesetzt wurde dies durch die Implementierung einer Methode, welche durch zwei Annotationen ergänzt wurde. „@Post" gibt die Methode der Anfrage an und „@Path" die URL unter welcher diese Methode aufgerufen werden soll. Der Body des Requests wird dann in den Paramter geparst. Die Annotation @Valid legt den Zeitpunkt fest, an dem die im UserAccount festgelegten Annotationen der Attribute überprüft werden sollen.

```
@POST
@Path("/register")
public Response register(@Valid UserAccount requestAccount) {
…
```

2.4.2 Login

Der Login geschieht über die URL „`/login`". Dazu muss der geeignete Body mitgeliefert werden. Dieser sieht wie folgt aus:

```
{
      „email": „<E-Mail>",
      „password": „<Passwort>"
}
```

Genauso wie bei der Registrierung ist auch hier die Methode aufgebaut.

```
@POST
@Path("/login")
public Response login(@Valid UserAccount requestAccount) {
…
```

2.5 Persistierung

Zur demonstrativen und einfachen Persistierung der Daten wurde eine h2-Datenbank verwendet. Die Konfiguration für die Verwendung dieser ist in einem Properties-File vorhanden

```
spring.datasource.url=jdbc:h2:./db/test;Mode=Oracle;DB_CLOSE_ON_EXIT=FALSE
spring.datasource.platform=h2
spring.datasource.continue-on-error=true
spring.jpa.hibernate.ddl-auto=update
```

Als erstes sind die Verbindungsdaten spezifiziert. An zweiter Stelle ist die Art der Datenbank definiert worden. Die dritte Zeile ermöglicht einen fortlaufenden Dienst, auch wenn Fehler aufgetreten sind. Zu guter Letzt ist „ddl-auto=update" definiert worden. Dies ermöglicht einen Erhalt der Daten und eine Anpassung des Schemas aufgrund der Entitäten.

2.6 Testfälle

Für die Überprüfung der Software wurden Acceptance Tests auf Ebene der RESTful Endpoints implementiert. Diese generieren ebenfalls einen Testbericht der nach einem Install durch Maven einsehbar ist. Der Ordner befindet sich unter „`target/classes/testreport`"

UserEndpointAcceptanceTest

	testRegisterToLongPassword	1,163
	testRegisterToShortPassword	0,019
	testRegisterEmptyPassword	0,008
	testRegisterEmptyEmail	0,01
	testRegisterInvalidPassword	0,008
	testRegisterInvalidEmail1	0,013
	testRegisterInvalidEmail2	0,147
	testLoginSuccess	0,054
	testLoginEmptyPassword	0,019
	testLoginInvalidEmail1	0,009
	testLoginInvalidEmail2	0,007
	testLoginEmptyEmail	0,01
	testLoginEmptyEmailAndPassword	0,012
	testRegisterEmptyEmailAndPassword	0,01
	testRegisterToLongEmail	0,008
	testLoginInvalidPassword	0,03
	testRegsiterAlreadyPresentUser	0,046
	testRegisterSuccess	0,013

2.7 Deployment und Ausführung

Generiert wird die Applikation durch das Ausführen von „`maven install`“. Spring Boot integriert ebenso einen Applikationsserver, welches zu einer einfachen Ausführung des Projekts mittels „`java –jar dezsys09.jar`“. Dieses jar-File befindet sich im Ordner „`final`“.

URL Registrierung: `127.0.0.1:8080/register`

URL Login: `127.0.0.1:8080/login`

Der Zugriff auf das Registrieren und den Login erfolgt mittels POST-Request mit folgendem Body:

```
{
     „email“: „<E-Mail>“,
     „password“: „<Passwort>“
}
```

Dabei ist zu beachten, dass sowohl „`Accept`“ und „`Content-Type`“ auf „`application/json`“ gestellt ist.

2.8 Probleme

- Das ausführen der Applikation führte zu einer Warning die meinte, die Applikation nicht im default package auszuführen. Danach folgte eine Exception `FileNotFoundException: class path resource [org/springframework/security/config/annotation/authentication/configurers/GlobalAuthenticationConfigurerAdapter.class]`
  - Lösung war das definieren eines übergeordneten Packages
- Bei einer falschen Eingabe von E-Mail oder Passwort, sollte eine Validierung mittels Annotation die Speicherung des Accounts verhindern. Dies geschah jedoch nicht und führte zu einem Transaktionsfehler. Grund dafür war die fehlende Dependency für `jersey-bean-validation`

3 Verweise

[1] S. Geyer, „DezSys09," 23 02 2016. [Online]. Available: https://github.com/sgeyer-tgm/DezSys09. [Zugriff am 11 03 2016].

[2] P. Kalauner, „DezSys09-java-webservices," 13 02 2016. [Online]. Available: https://github.com/pkalauner-tgm/dezsys09-java-webservices. [Zugriff am 11 03 2016].