

# **Seminar Hibernate**

**Seminar Hibernate**

**ZHAW - Zürcher Hochschule für Angewandte Wissenschaften**

**Ramon Schilling**

**`schilram@students.zhaw.ch`**

**12. Juni 2013**

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>2</b>
1.1	Aufgabenstellung . . . . .	2
1.2	Deliveries . . . . .	2
1.3	Zielsetzung . . . . .	2
1.4	Motivation . . . . .	2
<b>2</b>	<b>Projektmanagement</b>	<b>3</b>
2.1	Projektplanung . . . . .	3
2.2	Aufwand . . . . .	3
2.3	Hilfsmittel . . . . .	3
2.3.1	Versionskontrolle . . . . .	3
2.3.2	Dokumentation . . . . .	3
2.3.3	Programmierung . . . . .	4
2.3.4	Datenbank . . . . .	4
<b>3</b>	<b>Einarbeitung</b>	<b>5</b>
3.1	Vorgehen . . . . .	5
<b>4</b>	<b>Anwendung</b>	<b>6</b>
4.1	Frameworks . . . . .	6
4.2	Architektur . . . . .	6
4.2.1	Dependencies . . . . .	6
4.2.2	Struktur . . . . .	6
4.2.3	Klassendiagramm . . . . .	8
4.2.4	Klassendiagramm Paket model . . . . .	8
4.2.5	Klassendiagramm Pakete service und repository . . . . .	8
4.3	Konfigurationsfiles . . . . .	8
4.4	Installation . . . . .	9
4.4.1	Voraussetzungen . . . . .	9
4.4.2	Datenbank einrichten . . . . .	9
4.4.3	Tomcat einrichten . . . . .	9
4.4.4	Applikation starten . . . . .	9
4.5	Applikation verwenden . . . . .	10
4.5.1	Zutaten erfassen . . . . .	10
4.5.2	Masseinheiten erfassen . . . . .	10
4.5.3	Rezepte erfassen . . . . .	10
4.5.4	Rezepte suchen . . . . .	11

---

<b>5</b>	<b>Fazit</b>	<b>12</b>
<b>6</b>	<b>Abbildungsverzeichnis</b>	<b>13</b>
<b>7</b>	<b>Literaturverzeichnis</b>	<b>14</b>

# 1 Einleitung

Dieses Dokument wurde für das Seminar „SW Entwicklung mit Hibernate“ geschrieben.

## 1.1 Aufgabenstellung

- Einarbeitung in das Thema ORM mit Hibernate [1].
- Erstellung einer kleinen Anwendung mit Gui welche das Hibernate-Framework nutzt.
- Dokumentation welche die Architektur sowie die einzelnen Komponenten erklärt.

## 1.2 Deliveries

- Source Code und lauffähiger Maschinencode.
- Dokumentation

## 1.3 Zielsetzung

Das Ziel der Seminararbeit ist es, das O/R-Mapping Paradigma am Beispiel des Hibernate Frameworks zu verstehen und eine Anwendung unter Verwendung dieses Frameworks zu implementieren.

## 1.4 Motivation

Da ich seit Oktober letzten Jahres als Java Entwickler arbeite, war für mich klar, dass ich ein Seminar wählen wollte mit welchem ich meine Entwicklungskenntnisse vertiefen und neue Erfahrungen sammeln kann. Da das Thema ORM bei der Entwicklung von Software sehr wichtig ist, da die meisten Applikationen irgendwie mit einer Datenbank arbeiten war die Wahl schnell gefallen. Wir arbeiten in der Firma auch mit Hibernate und setzen auch das Framework Spring [8] ein. Da ich aber noch kein Projekt von Grund auf mitaufgebaut habe, wollte ich diese Arbeit als Chance nutzen nicht nur Hibernate genauer kennenzulernen sondern auch Spring, bzw. Teile davon. Im Speziellen wollte ich Spring MVC und Spring Data genauer anschauen.

Zur Umsetzung der Anforderungen möchte ich eine Anwendung entwickeln in welcher Rezepte verwaltet werden können. Als Besonderheit soll es möglich sein nach Rezepten zu Suchen in dem man die vorhandenen Zutaten angibt.

## 2 Projektmanagement

### 2.1 Projektplanung

20. März 2013 - Kick Off Meeting  
27. März 2013 - Einreichen der Aufgabe  
12. Juni 2013 - Abgabe Schriftliche Arbeit und Anwendung  
19. Juni 2013 - Präsentation

### 2.2 Aufwand

Der Aufwand für die Seminararbeit sollte 50 h betragen.

Beschreibung	Soll	Ist
Einarbeitung in Hibernate	4 h	2 h
Einarbeitung in Spring	4 h	2 h
Einrichten der Entwicklungsumgebung inkl. Hibernate Test	4 h	8 h
Programmieren der Anwendung	28 h	32 h
Dokumentation	8 h	8 h
<b>Total</b>	<b>48 h</b>	<b>52 h</b>

Die Aufwandschätzung fiel mir relativ schwer, was sich auch darin zeigt, dass die Soll und Ist Zeiten nicht übereinstimmen. Ich habe für die theoretische Einarbeitung in Hibernate und Spring weniger Zeit aufwendete als geplant um schneller mit der Umsetzung zu beginnen. Bis ich dann aber alles soweit konfiguriert hatte, dass das Object-Relation-Mapping funktionierte hatte ich viel mehr Zeit gebraucht als geplant. Beim Programmieren habe ich ebenfalls z.T. sehr viel Zeit gebraucht um Fehler zu finden welche ich mit etwas mehr Erfahrung wohl gar nicht erst gemacht hätte.

### 2.3 Hilfsmittel

#### 2.3.1 Versionskontrolle

Um eine Versionskontrolle zu haben, habe ich in Github [3] ein Repository erstellt und sämtliche für das Projekt nötigen Dateien dort eingchecked.

#### 2.3.2 Dokumentation

Die Dokumentation habe ich mit  $\text{\LaTeX}$  erstellt. Als Editor habe ich TeXnicCenter [9] genutzt.

---

### **2.3.3 Programmierung**

Als Entwicklungsumgebung für die Java Programmierung habe ich IntelliJ IDEA [5] genutzt.

### **2.3.4 Datenbank**

Als Datenbank habe ich auf PostgreSQL [4] zurückgegriffen.

## 3 Einarbeitung

### 3.1 Vorgehen

Um mich mit dem Thema ORM und Hibernate auseinanderzusetzen habe ich hauptsächlich im Internet nach Informationen gesucht. Da mich die Integration zusammen mit Spring interessiert hat und ich die Seminararbeit ebenfalls als Gelegenheit nutzen wollte um mich mit Teilen von diesem Framework auseinanderzusetzen habe ich mein spezielles Augenmerk darauf gerichtet.

Grundlegende Informationen zu finden war eigentlich relativ einfach, als ich aber konkrete Beispiele suchen wollte stiess ich zum Teil auf Probleme. Es gibt zwar jede Menge von Tutorials und Beispiel Applikationen. Viele von diesen beleuchten aber nur ein ganz spezifisches Thema oder werfen im ersten Moment jede Menge neuer Fragen auf. Ich habe mich dadurch zum Teil beim Aufklären dieser neuen Fragen etwas verzettelt und Zeit verloren. Ich habe dadurch zwar jede Menge Interessantes gelesen, bin aber dem Ziel der Arbeit nicht immer näher gekommen.

Sehr geholfen hat mir das Buch Spring Data [6] in welchem in den ersten Kapiteln Spring Data JPA gut erklärt wird. Eine weitere wichtige Quelle waren natürlich die Seiten von Hibernate und Spring selber.

## 4 Anwendung

### 4.1 Frameworks

Die Anforderung an diese Seminararbeit war, dass Hibernate genutzt wird. Zusätzlich habe ich mich mit Spring auseinandergesetzt. Speziell nutzte ich für diese Arbeit Spring MVC und Spring Data. Für das Logging habe ich Logback [7] genutzt. Um die Darstellung der Web Oberfläche einfach und ansprechend zu gestalten habe ich Bootstrap [10] benutzt.

### 4.2 Architektur

Die Anwendung Besteht aus einer Webapplikation. Ich habe ein Model-View-Controller Ansatz gewählt welcher mit dem Framework Spring MVC einfach umzusetzen war. Web Anfragen werden von Spring entgegengenommen und an die Controller Klassen weitergeleitet, welche die Anfrage bearbeiten. Für das ORM habe ich Spring Data und Hibernate verwendet.

#### 4.2.1 Dependencies

Da es sich um ein Maven Projekt handelt werden sämtliche Dependencies über die Datei *pom.xml* verwaltet.

#### 4.2.2 Struktur

Der Source Code ist grundsätzlich in 3 Ordnern untergebracht:

- *java*: hier sind sämtliche JAVA Klassen in verschiedenen Paketen untergebracht.
- *resources*: hier werden Konfigurationsdateien gespeichert
- *webapp*: hier sind die für die Web Applikation benötigten Dateien (jsp, css, js) und das web.xml gespeichert



#### 4.2.2.1 Pakete

Ich habe für die Anwendung den Paket Prefix `ch.zhaw.schilram.sem_hib` genutzt. Die Anwendung hat den Namen `sem_hib`. Die JAVA Klassen sind in verschiedene Pakete unterteilt.

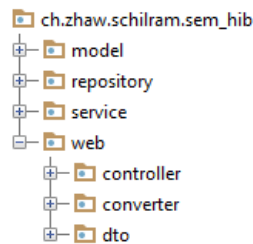


Abbildung 4.1: Paketstruktur

##### **model**

Im Paket `model` sind die Model Klassen untergebracht welche von Hibernate für das OR Mapping genutzt werden. Sämtliche Model Klassen implementieren das Interface `Uniqueness`

##### **repository**

Im Paket `repository` sind sämtliche Interfaces welche das `JpaRepository` Interface implementieren und für den Zugriff auf die Datenbank dienen.

##### **service**

Das Paket `service` beinhaltet für jede Model Klasse ein Service Interface und eine Service Klasse welche den Zugriff auf die Datenbank ermöglicht. Die Klasse `AbstractCrudService` implementiert die CRUD Methoden welche durch die Interfaces `CrudService` und `ReadService` vorgegeben werden. Die Service Interfaces extenden jeweils das Interface `CrudService` und werden von den Service Klassen welche auch den `AbstractCrudService` extenden implementiert. Siehe dazu auch Abbildung 4.3.

##### **web.controller**

Im Paket `web.controller` sind die Controller Klassen abgelegt welche die Web Requests entgegennehmen und verarbeiten

##### **web.converter**

Hier sind einerseits die Converter Klassen gespeichert, welche statische Methoden zur Umwandlung einer Model Klasse in die entsprechende DTO Klasse anbieten, sowie auch die Converter, welche gebraucht werden um über in Web Formularen als String übermittelte ID das zugehörige persistierte Objekts zu finden.

##### **web.dto**

Im Pakte `web.dto` sind die DTO bzw. Formular Klassen welche für die Formulareingabe genutzt werden abgelegt.

### 4.2.3 Klassendiagramm

Unten sind die Klassendiagramme der Pakete *model* und *service* aufgeführt.

### 4.2.4 Klassendiagramm Paket model

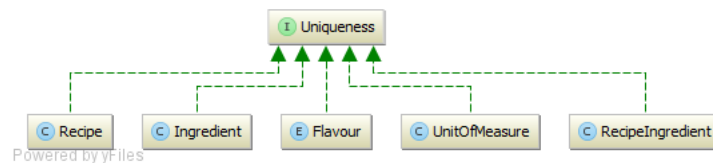


Abbildung 4.2: Klassendiagramm Paket *model*

### 4.2.5 Klassendiagramm Pakete service und repository

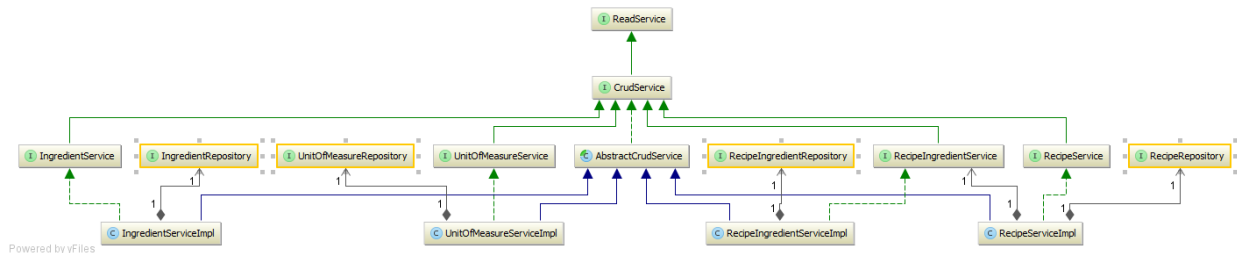


Abbildung 4.3: Klassendiagramm Paket *service*

## 4.3 Konfigurationsfiles

Im Ordner *ressources* sind die Konfigurationsfiles hinterlegt.

**application.properties** In dieser Datei ist die Hibernate Konfiguration gespeichert. Um eine andere Datenbank oder einen anderen Datenbankuser zu verwenden müsste diese Datei angepasst werden.

#### root-context.xml

Dies ist das Spring Basis Konfigurationsfile. Hier wird festgelegt in welchen Paketen Spring nach den Repository Klassen und Service Klassen sucht.

#### datasource-context.xml

Hier wird Spring die Datenquelle bekanntgegeben. Die relevanten Daten werden aus dem Konfigurationsfile *application.properties* ausgelesen.

#### dispatcher-context.xml

Dies ist die Konfigurationsdatei für Spring MVC.

### logback.xml

Die Konfigurationsdatei für das Logging.

## 4.4 Installation

### 4.4.1 Voraussetzungen

Damit die Applikation läuft muss sie auf eine Datenbank zugreifen können. Zusätzlich wird Tomcat [2] vorausgesetzt um die Web Applikation laufen zu lassen.

### 4.4.2 Datenbank einrichten

Die Applikation läuft mit PostgreSQL. PostgreSQL kann unter <http://www.postgresql.org/> heruntergeladen werden.

Nach der Installation von PostgreSQL muss der Benutzer angelegt werden. Standardmässig wird der Benutzer *sem\_hib* mit dem Passwort *sem\_hib* genutzt. Der Benutzer kann über das Gui oder mit folgendem SQL Statement erstellt werden.

```
CREATE ROLE sem_hib LOGIN
    ENCRYPTED PASSWORD 'md5198ad17e37731cbad30b3130a0a88919 '
    VALID UNTIL 'infinity';
```

Nach der Erstellung des Benutzers muss noch die Datenbank erstellt werden. Der Datenbankbesitzer muss dabei auf den eben erstellten Benutzer festgelegt werden. Die Datenbank kann wahlweise über das Gui oder über das untenstehende SQL Statement erzeugt werden.

```
CREATE DATABASE sem_hib
    WITH ENCODING='UTF8'
    OWNER=sem_hib
    CONNECTION LIMIT=-1;
```

### 4.4.3 Tomcat einrichten

Der Tomcat Server kann von <http://tomcat.apache.org/> heruntergeladen werden. Nach der Installation von Tomcat kann die Applikation über das Management Interface installiert werden. Die Benötigte Datei heisst *sem\_hib.war*. Alternativ kann das File *sem\_hib.war* kann direkt in den Ordner *webapps* von Tomcat kopiert werden.

### 4.4.4 Applikation starten

Nachdem die Applikation installiert ist kann über [http://localhost:8080/sem\\_hib/](http://localhost:8080/sem_hib/) darauf zugegriffen werden.

## 4.5 Applikation verwenden

Die Applikation besteht aus den Teilen *Zutaten*, *Masseinheiten*, *Rezepte* und *Suche*.

Zuerst müssen Zutaten und Masseinheiten erfasst werden. Danach können diese beim Erfassen von Rezepten ausgewählt werden. Wenn einmal ein paar Rezepte erfasst sind, kann über die Suche nach Rezepten gesucht werden welche bestimmte Zutaten enthalten.

### 4.5.1 Zutaten erfassen

Einer Zutat kann ein Name gegeben werden und eine Beschreibung. Zusätzlich kann noch der Geschmack (Salzig, Süß, Sauer, Bitter oder Umami) angegeben werden.

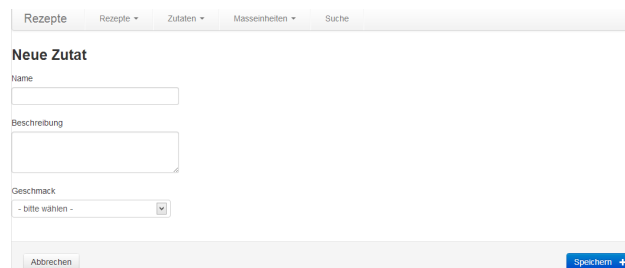


Abbildung 4.4: Zutat erfassen

### 4.5.2 Masseinheiten erfassen

Der Masseinheit kann ein Key und ein Name gegeben werden. Ebenfalls kann noch eine Beschreibung hinzugefügt werden. Über den Key (z.B. EL für Esslöffel) wählt man beim Rezept dann die Masseinheit aus.

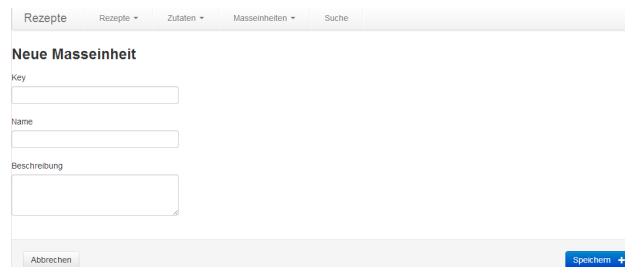
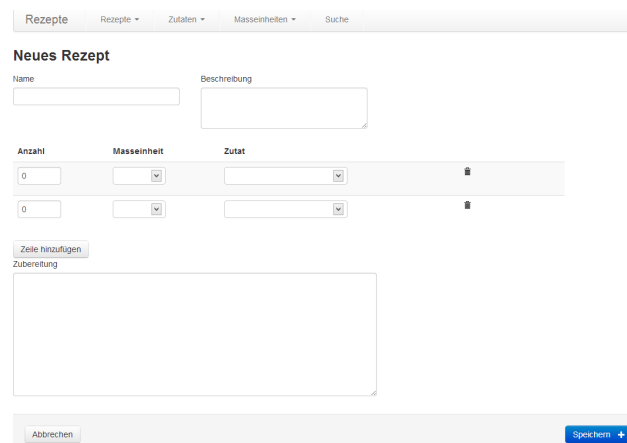


Abbildung 4.5: Masseinheit erfassen

### 4.5.3 Rezepte erfassen

Beim Erfassen eines Rezeptes ist neben dem Namen natürlich wichtig, dass die Zutaten angegeben werden können. Über den Button *Zeile hinzufügen* kann eine weitere Zutatenzeile hinzugefügt werden. Über das Löschen Icon neben der Zeile kann eine Zeile gelöscht werden. Im Feld *Zubereitung* wird die Kochanleitung hinterlegt.



Rezepte   Rezepte -   Zutaten -   Masseinheiten -   Suche

### Neues Rezept

Name  Beschreibung

Anzahl	Masseinheit	Zutat	
<input type="text" value="0"/>	<input type="text"/>	<input type="text"/>	
<input type="text" value="0"/>	<input type="text"/>	<input type="text"/>	

Zubereitung

Abbildung 4.6: Rezept erfassen

#### 4.5.4 Rezepte suchen

Bei der Suche können bis zu drei Zutaten angegeben werden. Als Resultat werden alle Rezepte aufgelistet welche mindestens eine dieser Zutaten benötigen.

## 5 Fazit

Ich fand die Auseinandersetzung mit den genutzten Technologien spannend und sehr lehrreich. Ich bin überzeugt, dass ich die hier gewonnenen Erfahrungen weiter nutzen kann da ich in Zukunft noch öfters mit Hibernate arbeiten werde. Trotz, oder vielleicht auch wegen einiger nervenzehrender Momente hat mir die Arbeit auch viel Spass bereitet.

Bis ich die ersten Tests mit Hibernate erfolgreich hinter mich gebracht hatte dauerte es recht lange. Das war zwar einerseits frustrierend, zeigte mir aber auch wie wichtig das grundlegende Verständnis für die genutzten Technologien ist. Einige im Internet verfügbare Tutorials gehen total verschiedene Wege und zeigen zum Teil nur einen kleinen Ausschnitt. Daraus jeweils die für das eigene Problem wichtigen Teile herauszufiltern ist manchmal ziemlich knifflig. Dies ist vor allem der Fall wenn man sich mit mehreren Frameworks gleichzeitig auseinandersetzt welche man noch nicht kennt.

Ich habe auch den reinen Programmieraufwand unterschätzt, da ich mich manchmal mit Fehlern aufgehalten habe, welche ich bestimmt nicht nochmals machen werde. Ein Beispiel hierfür ist, dass beim Editieren eines Rezeptes nicht die richtigen Masseinheiten und Zutaten angezeigt wurden obwohl ich mit dem Debugger sehen konnte, dass die richtigen Eigenschaftsnamen ausgelesen wurden und auch im DTO die Angaben stimmten. Da ich in der Model Klasse aber die *equals* Methode nicht überschrieben hatte, konnten die Objekte nicht verglichen werden.

Ich hätte die Applikation gerne noch ausgebaut und verfeinert. Leider bin ich nicht mehr dazu gekommen eine Validierung zu implementieren. Dies wäre sicher einer der nächsten Schritte. Ein weiterer Punkt welchem ich gerne Beachtung geschenkt hätte wäre die internationalisierung gewesen.

## 6 Abbildungsverzeichnis

4.1	Paketstruktur . . . . .	7
4.2	Klassendiagramm Paket <i>model</i> . . . . .	8
4.3	Klassendiagramm Paket <i>service</i> . . . . .	8
4.4	Zutat erfassen . . . . .	10
4.5	Masseinheit erfassen . . . . .	10
4.6	Rezept erfassen . . . . .	11

## 7 Literaturverzeichnis

- [1] JBoss Community. Hibernate. <http://hibernate.org/>, 2013. aufgerufen 6. Juni 2013.
- [2] The Apache Software Foundation. Apache tomcat. <http://tomcat.apache.org/>, 2013. aufgerufen 6. Juni 2013.
- [3] Github. Github. <https://github.com/>, 06 2013. aufgerufen am 11. Juni 2013.
- [4] The PostgreSQL Global Development Group. Postgresql. <http://www.postgresql.org/>, 2013. aufgerufen 6. Juni 2013.
- [5] JetBrains. IntelliJ idea. <http://www.jetbrains.com/idea/>, 06 2013. aufgerufen am 11. Juni 2013.
- [6] Petri Kainulainen. *Spring Data*. Packt Publishing, Packt Publishing Ltd Livery Place 35 Livery Street Birmingham B3 2PB, UK, 2012. ISBN 978-1-84951-904-5.
- [7] QOS.ch. Logback. <http://logback.qos.ch/>, 2013. aufgerufen 6. Juni 2013.
- [8] Spring Source. Spring. <http://www.springsource.org/>, 2013. aufgerufen 6. Juni 2013.
- [9] texniccenter.org. Texniccenter. <http://www.texniccenter.org/>, 2013. aufgerufen 6. Juni 2013.
- [10] Twitter. Bootstrap. <http://twitter.github.io/bootstrap/index.html/>, 2013. aufgerufen 6. Juni 2013.