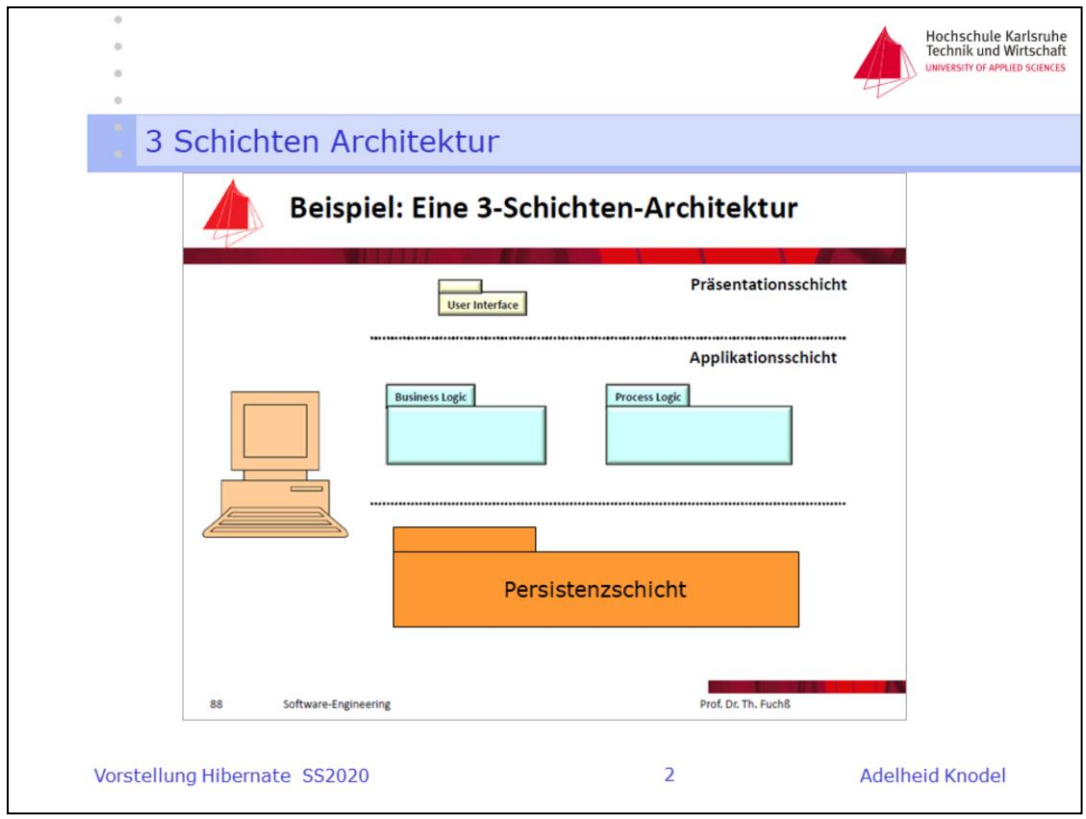
Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Vorstellung des Frameworks Hibernate

Labor für Verteilte Systeme
Master
Fachgebiet Informatik
Hochschule Karlsruhe

Dipl.-Inform.(FH) Adelheid Knodel



Wie schon in der Vorstellung des Frameworks Struts2 wird auch hier noch mal auf die 3 Schichten Architektur verwiesen.

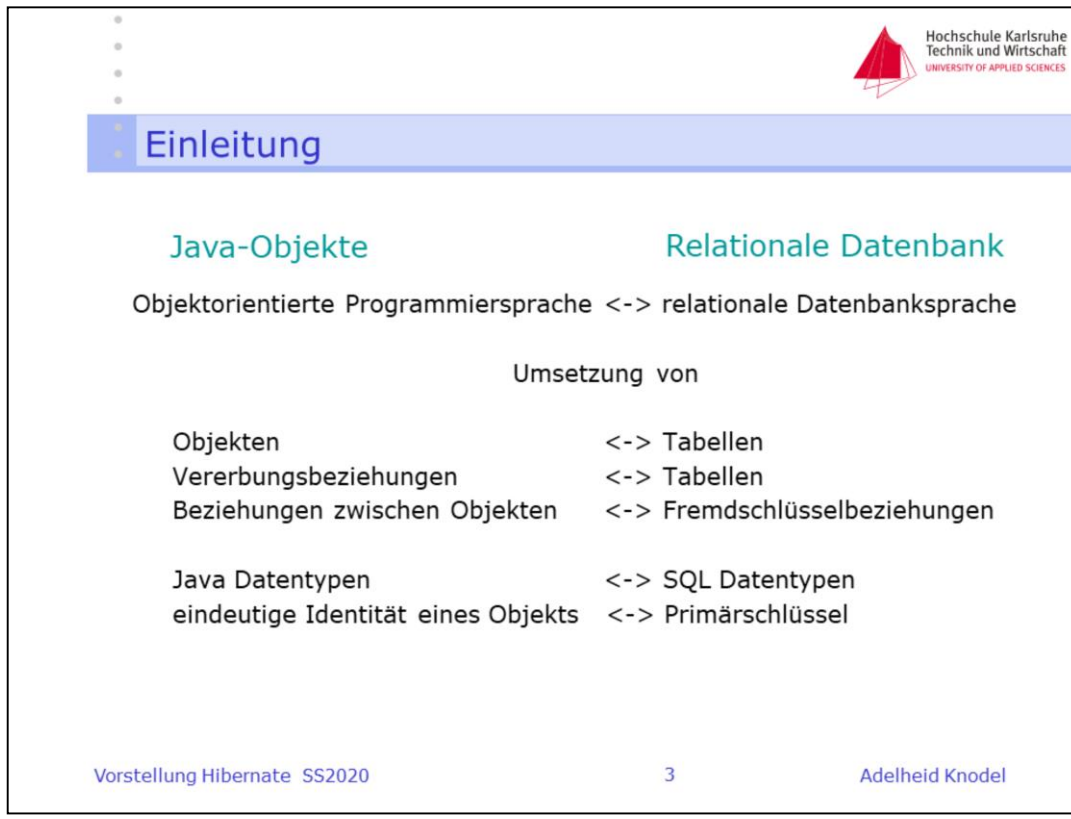
Die Verbindung zwischen einer Java-Anwendung und einer Datenbank auf der Basis von JDBC und SQL ist realtiv aufwändig und damit teuer.

Für jede neue Anwendung mit einem anderen Domänenmodell ist erneut dieser Aufwand zu treiben, der wieder mit neuen Fehlern im Code einher geht.

Wenn zwischen der Datenbank und der Anwendungsschicht eine Persistenzschicht die Abbildung von Objekten

auf Relationen übernimmt, fördert dies die Modularität des Gesamtsystems und ermöglicht erst den Austausch der Datenhaltungskomponente.

Mit Hilfe des Framework Hibernate kann dieser Prozess der Umsetzung bis zu einem gewissen Maße strukturiert und automatisiert werden.



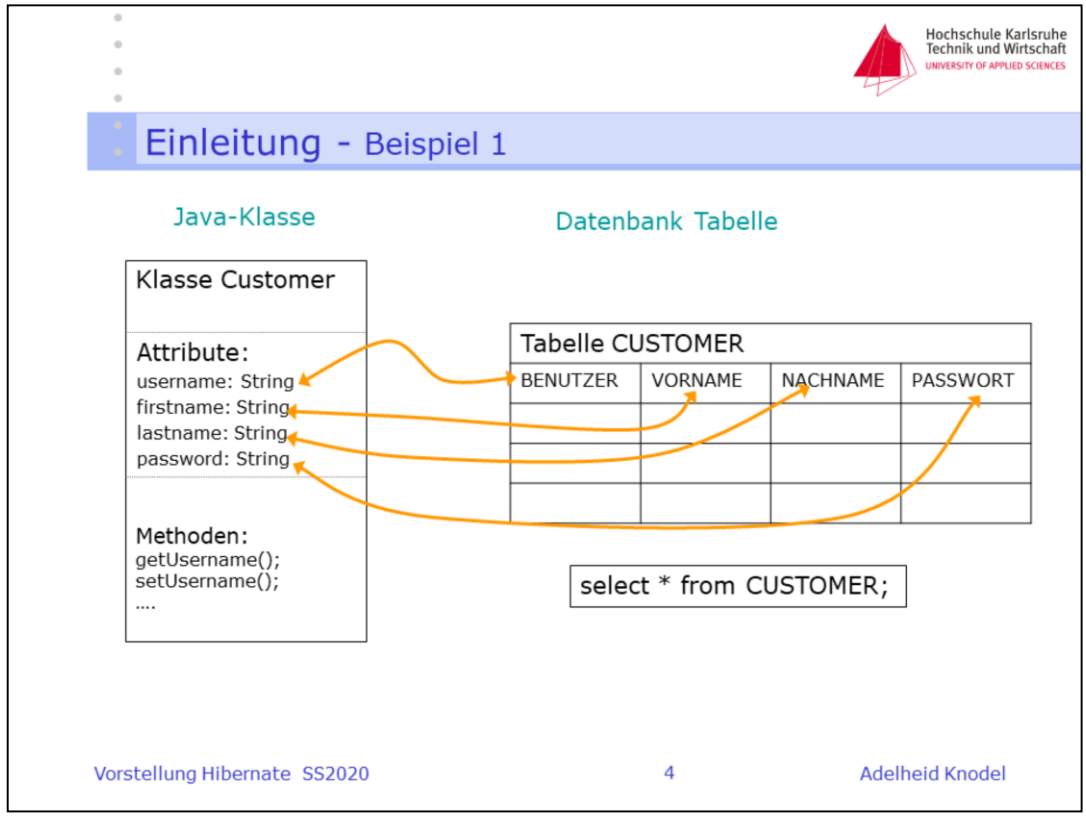
Objektorientierte Programmiersprachen sind heute Standard bei der Softwareerstellung, im Bereich der Datenhaltung sind relationale Datenbanksysteme etabliert.

Die meisten Webanwendungen speichern ihre Daten in relationalen Datenbanken und sind selbst objektorientiert programmiert. Während des Programmablaufs sind die Daten in Objekten gekapselt, diese serialisiert in der Datenbank zu hinterlegen wäre keine gute Idee, würde man doch die Vorteile einer relationalen Datenbank einbüßen.

Ein Objekt hat einen Zustand, Verhalten und eine Identität.

Eine relationale Datenbank kennt nur Relationen und dazugehörige Tupel.

Die Schwierigkeit ist nun die Umsetzung von Objekten in Datenbank-Tabellen, wie funktioniert das mit Datenbankrelationen, wie werden Assoziationen in Fremdschlüsselbeziehungen umgesetzt, wie kann eine Vererbungshierarchie umgesetzt werden, da es in der Datenbank keine Entsprechung gibt.



Grundlegende Techniken

Im einfachsten Fall werden Klassen auf Tabellen abgebildet, jedes Objekt entspricht einer Tabellenzeile und für jedes Attribut wird eine Tabellenspalte reserviert. Die Identität eines Objekts entspricht dem Primärschlüssel der Tabelle. Hat ein Objekt eine Referenz auf ein anderes Objekt, so kann diese mit einer Fremdschlüssel-Primärschlüssel-Beziehung in der Datenbank dargestellt werden.

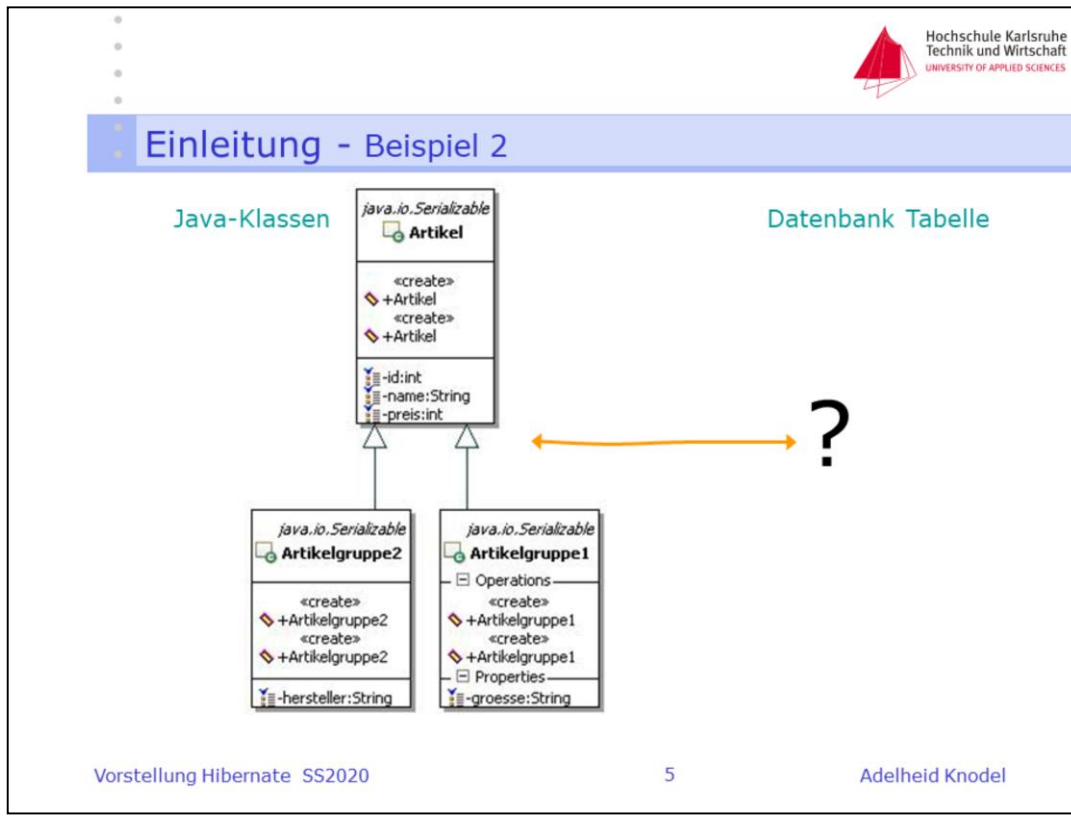


Abbildung von Vererbungshierarchien

Es gibt im Wesentlichen drei verschiedene Verfahren, um Vererbungshierarchien auf Datenbanktabellen abzubilden.

Tabelle pro Vererbungshierarchie

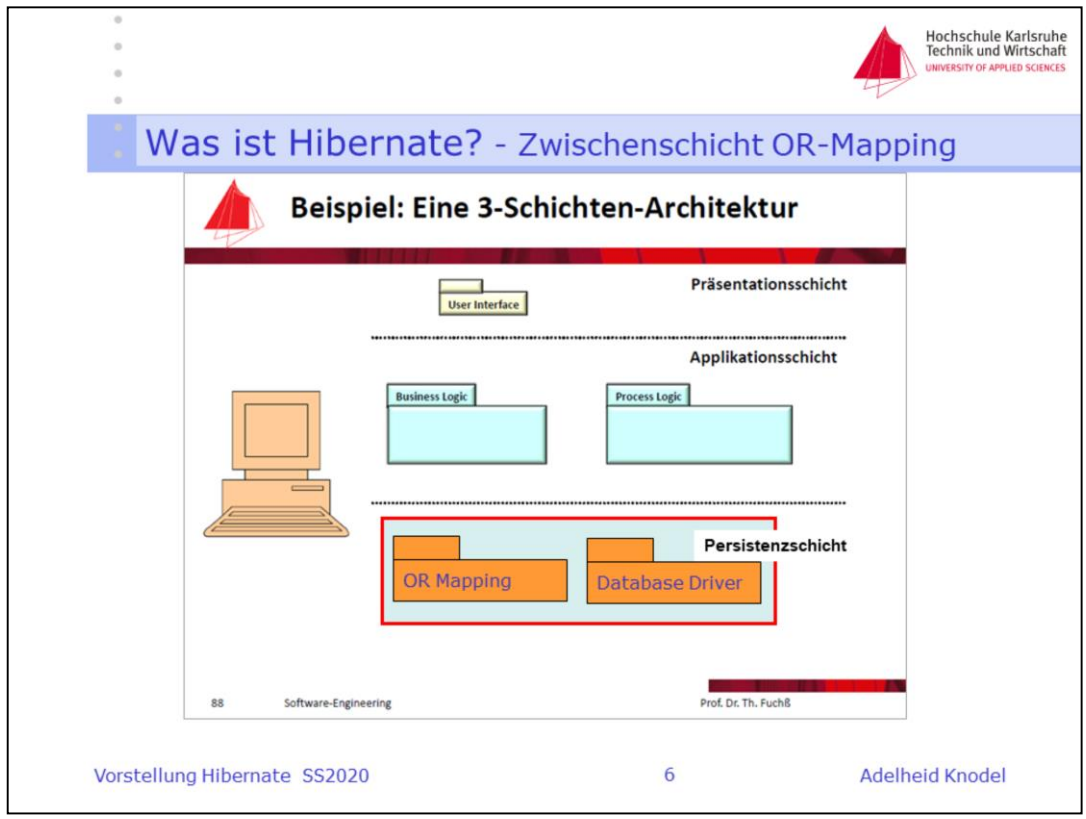
(auch *Single Table*, „einzelne Tabelle“) Bei diesem Verfahren werden alle Attribute der Basisklasse und aller davon abgeleiteten Klassen in einer gemeinsamen Tabelle gespeichert. Zusätzlich wird ein sogenannter „Diskriminator“ in einer weiteren Spalte abgelegt, der festlegt, welcher Klasse das in dieser Zeile gespeicherte Objekt angehört. Attribute von abgeleiteten Klassen dürfen bei diesem Ansatz aber in den meisten Fällen nicht mit einem NOT NULL Constraint versehen werden. Ausserdem können Beschränkungen der Anzahl erlaubter Spalten pro Tabelle diesen Ansatz bei großen Klassen bzw. Klassenhierarchien vereiteln.

Tabelle pro Unterklasse

(auch *Joined*) Bei diesem Verfahren wird eine Tabelle für die Basisklasse angelegt und für jede davon abgeleitete Unterklasse eine weitere Tabelle. Ein Diskriminator wird nicht benötigt, weil die Klasse eines Objekts durch eine 1-zu-1-Beziehung zwischen dem Eintrag in der Tabelle der Basisklasse und einem Eintrag in einer der Tabellen der abgeleiteten Klassen festgelegt ist.

Tabelle pro konkrete Klasse

(auch *Table per Class*) Hier werden die Attribute der abstrakten Basisklasse in die Tabellen für die konkreten Unterklassen mit aufgenommen. Die Tabelle für die Basisklasse entfällt. Der Nachteil dieses Ansatzes besteht darin, dass es nicht möglich ist, mit einer Abfrage Instanzen verschiedener Klassen zu ermitteln.



Mit Einführung der objektorientierten Programmierung (OOP) wurde ein neues Datenzugriffskonzept benötigt.

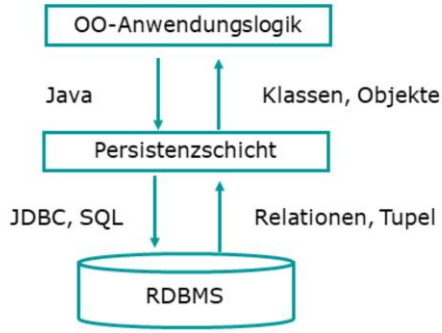
Bestehende relationale Datenbanken können mit OOP-Anwendungen über spezielle Mapping-Tools sogenannte O/R-Mapper verbunden werden.

Die Frameworks für das O/R-Mapping haben die Aufgabe den Datenaustausch zwischen Objekten und Tabellen zu automatisieren.

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Was ist Hibernate?

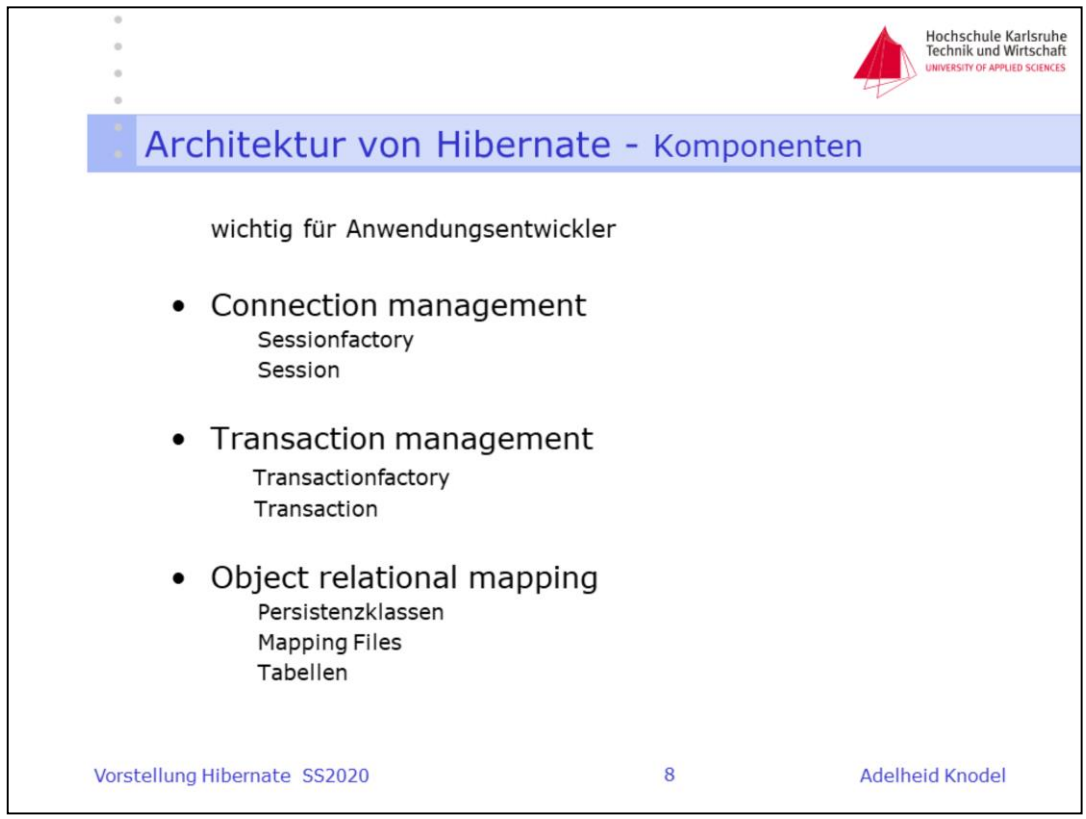
- Hibernate ist ein O/R Persistenz-Framework
- Verbindung zwischen Objektmodell und relationalem Modell



Eines dieser Frameworks ist das Framework Hibernate

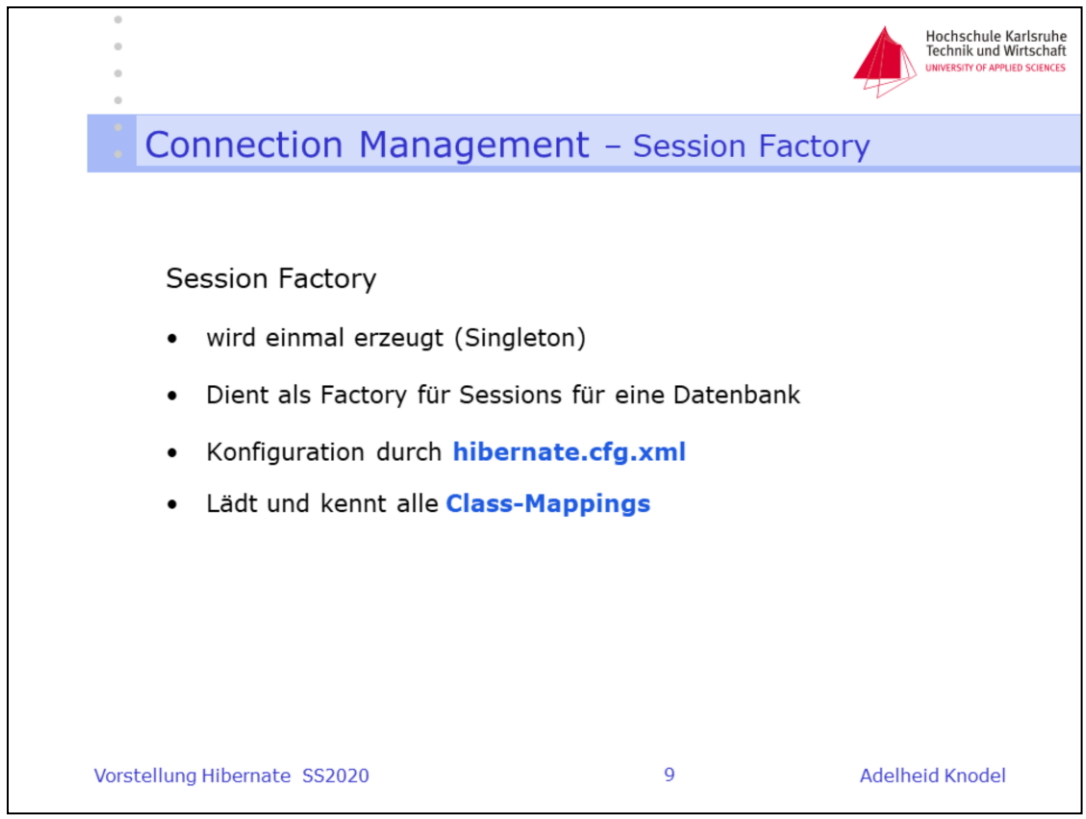
Für Java gibt es darüber hinaus auch eine standardisierte Schnittstelle, die Java Persistence API.

Die Abbildung von Objekten auf Relationen hat den Vorteil, dass einerseits die Programmiersprache selbst nicht erweitert werden muss und andererseits relationale Datenbanken als etablierte Technik in allen Umgebungen als ausgereifte Software verfügbar sind.



Die Architektur von Hibernate benötigt 3 hauptsächliche Komponenten

- **Connection management**
  Da das Herstellen und Schließen von Verbindungen zur Datenbank der aufwändigste Teil einer Interaktion mit der Datenbank ist, werden die Verbindungen in einem Pool gehalten und immer wieder verwendet. Das Connection Management besteht darin Session Instanzen anzulegen, dafür zuständig ist die SessionFactory, die es für jede Anwendung einmal gibt.
  Die Sessions sind Threads, die die Requests an die Datenbank abarbeiten.
- **Transaction management**
  Eine Transaction wird benutzt, wenn in einer Anfrage an die Datenbageht schief.nk mehr als eine Query ausgeführt werden soll. Bei einer Transaktion ist entweder die gesamte Transaktion erfolgreich oder die komplette Transaktion.
- **Object relational mapping**
  Dies ist der Teil, wo Hibernate aus einem Java Objekt ein Datenbankobjekt erzeugt, um das zu ermöglichen, muss es verschiedene Konfigurationen geben,
  z.B.: welche Klassen sollen persistiert werden, welche Objekte sollen auf welche Tabellen gemappt werden, wie sieht die Struktur der Objekte, bzw. Tabellen aus.



Wie sieht die Konfiguration konkret aus?

Beginnen wir bei der Session Factory fürs Connection Management:

Die SessionFactory wird normalerweise einmal beim Systemstart erzeugt, mit einem load-on-startup servlet.

Ganz wichtig ist auch, dass die SessionFactory threadsafe ist, d.h. die SessionFactory hat eine ThreadLocal Variable, die die Session für den aktuellen Thread hält, der die Anfrage stellt.

Konfiguriert wird die SessionFactory über die hibernate.cfg.xml Datei

Die hibernate.cfg.xml Datei enthält die Daten für die Verbindung zur Datenbank und die Klassen, die persistiert werden sollen.

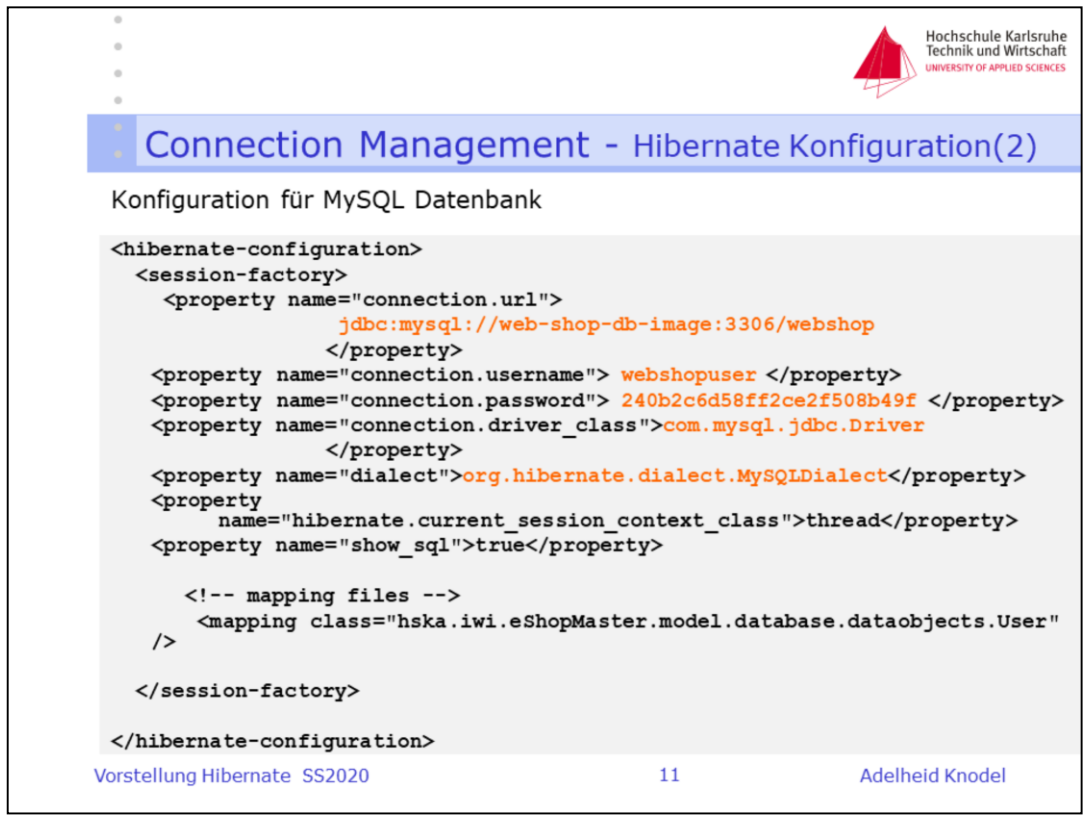
Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Connection Management - Hibernate Konfiguration

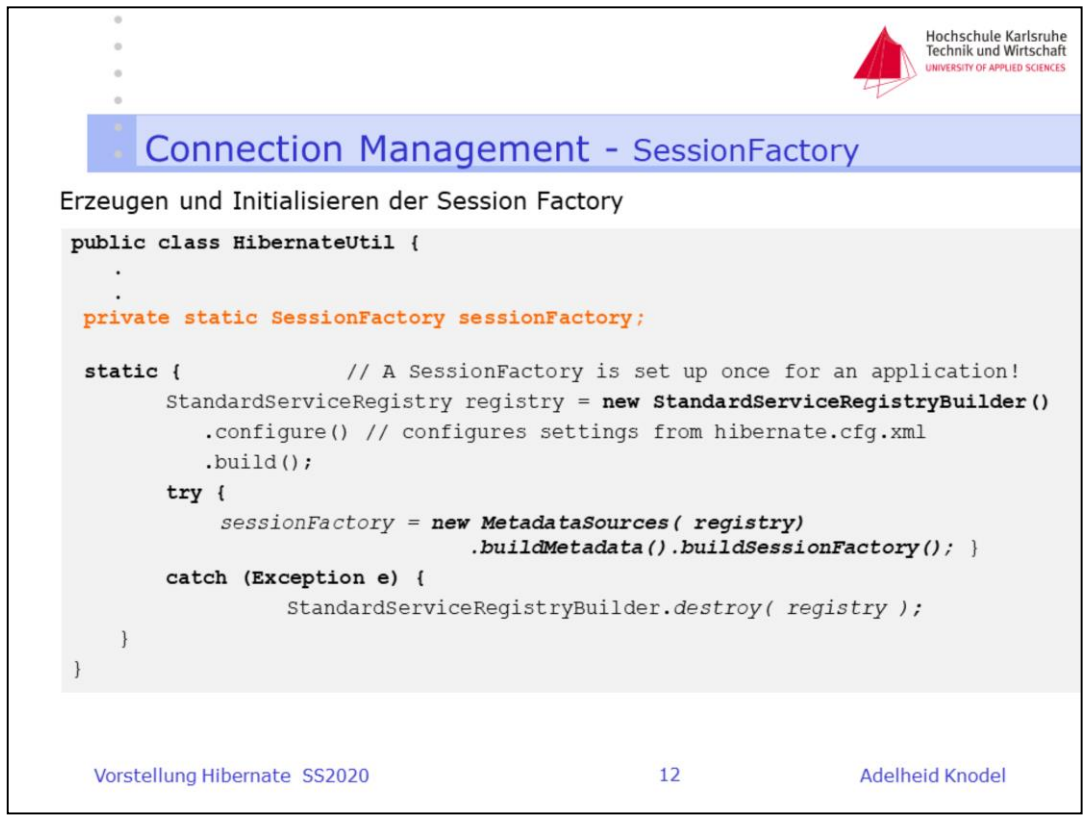
hibernate.cfg.xml

```
<hibernate-configuration>
    <session-factory>

   <property name="connection.url"> url zur Datenbank </property>
   <property name="connection.username">username</property>
   <property name="connection.password">passwort</property>
   <property name="connection.driver_class"> DriverKlasse </property>
   <property name="dialect">Datenbankdialekt</property>
   <property name="show_sql">true</property>

      <!-- mapping files -->
      <mapping class="classA "/>
      .
      .
      <mapping class="classZ "/>
    </session-factory>

</hibernate-configuration>
```

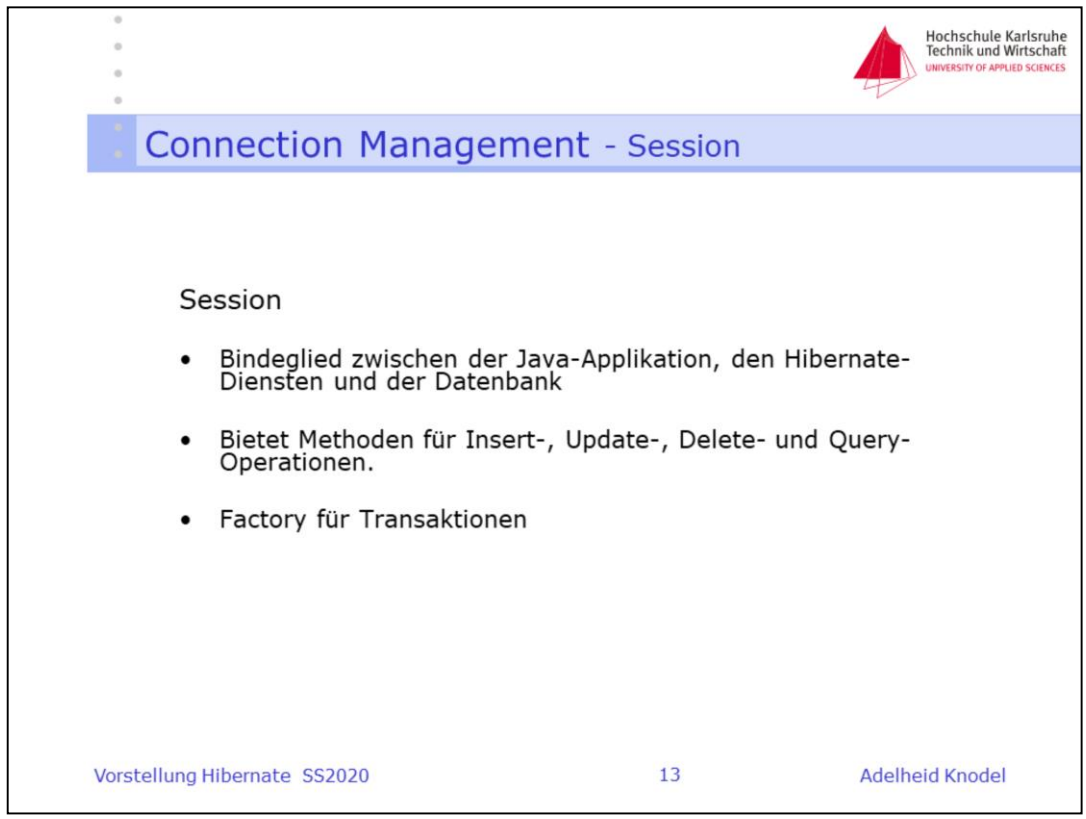

Hier das prinzipielle Aussehen der hibernate.cfg.xml Datei



Hier das konkrete Beispiel für den zu analysierenden Webshop



Anhand dieser Klasse HibernateUtil und der hibernate.cfg.xml wird beim Start des Webshop die SessionFactory erzeugt und die Verbindung zur Datenbank aufgebaut.

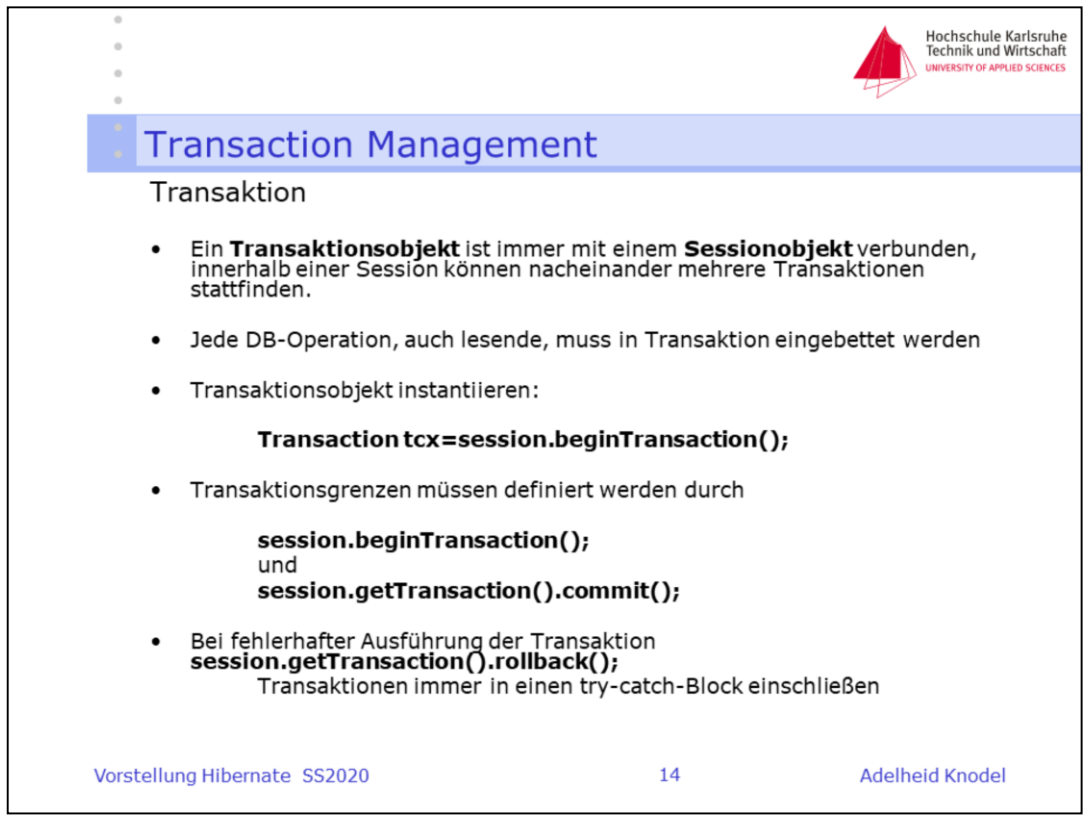


Im Gegensatz zur SessionFactory werden die Sessions jedesmal erzeugt, wenn eine Anfrage an die Datenbank gestellt werden soll.

Ist diese Anfrage abgearbeitet, wird die Session wieder geschlossen.

Das Session Objekt stellt Operationen zum Arbeiten mit der Datenbank zur Verfügung, wie z.B. insert, update, delete und query

Jede Operation muss in eine Transaktion eingebettet werden.



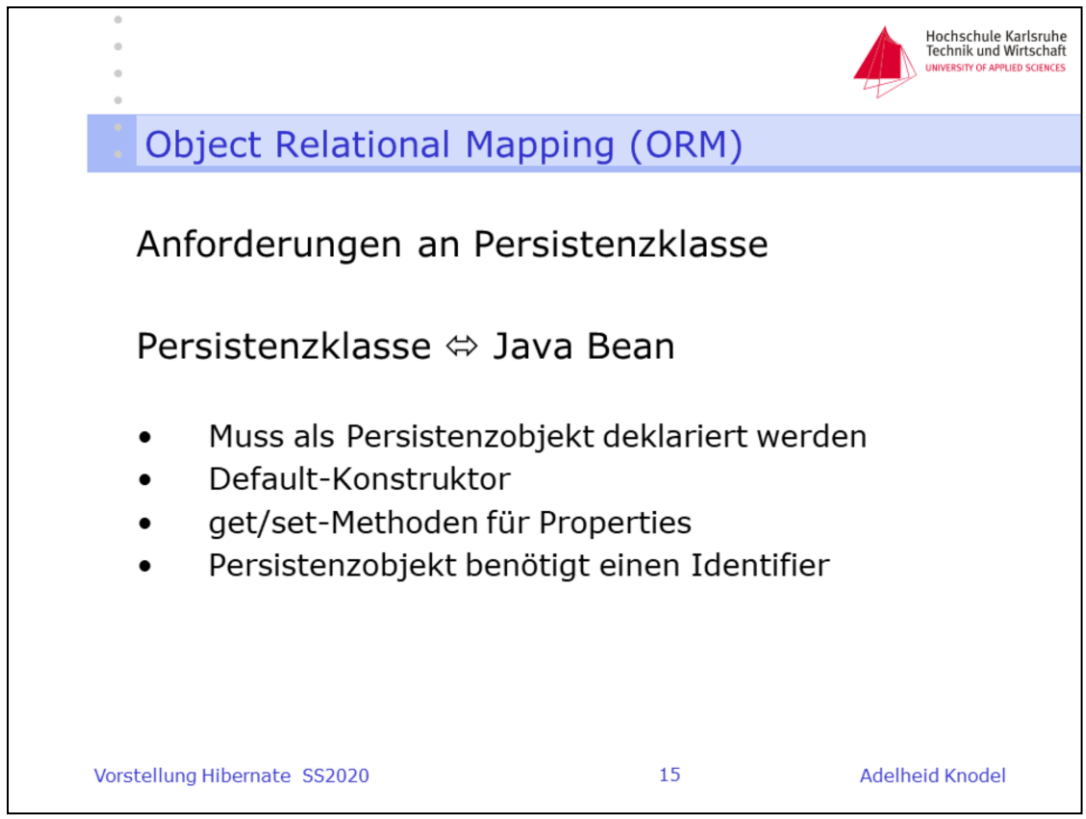
Ein Transaktionsobjekt ist immer mit einer Session verbunden.

Innerhalb einer Session können mehrere Transaktionen hintereinander stattfinden.

Jede Datenbank Operation muss in eine Transaktion eingebettet werden, d.h. es muss zuerst eine Transaktion erzeugt werden durch session.beginTransaction()

Eine Transaktion muss immerfentweder mit session.getTransaction().commit() bei erfolgreicher Transaktion beendet werden oder

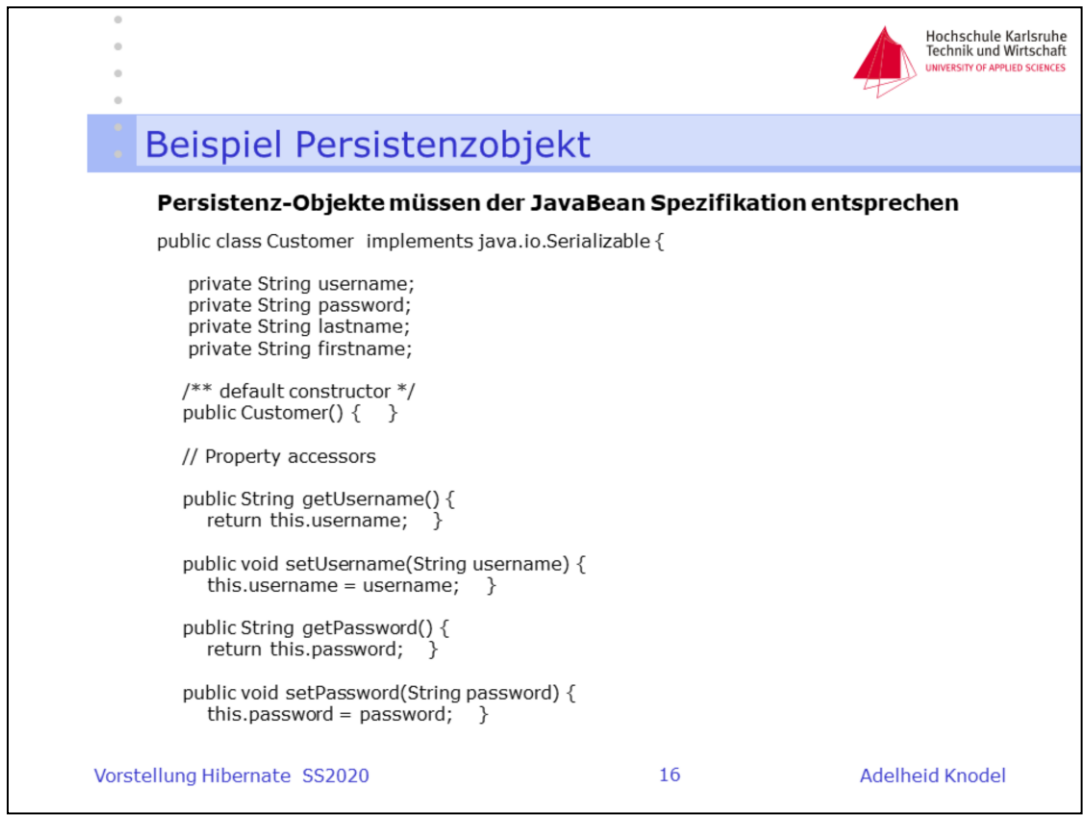
Falls etwas schiefgegangen ist mit session.getTransaction().rollback() wieder zurückgesetzt werden.



Als weiteres müssen die zu persistierenden Klassen gewissen Anforderungen genügen,

Die Klasse muss

- als zu persistierende Klasse annotiert werden,
- Mit einem default Konstruktor versehen werden
- Für alle zu persistierenden Eigenschaften jeweils getter und setter Methoden besitzen
- Einen eindeutigen Identifier besitzen

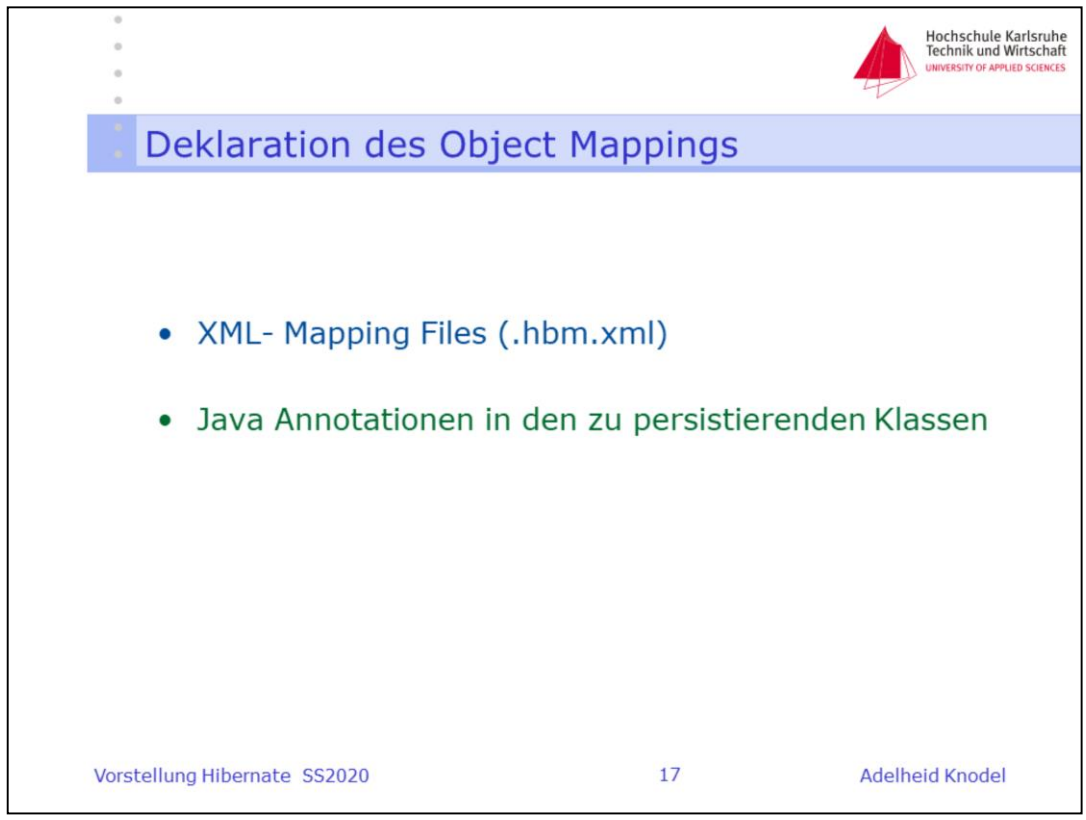


Beispiel für eine zu persistierende Klasse,

Die Klasse Customer besitzt 4 Properties: username, password, lastname, firstname

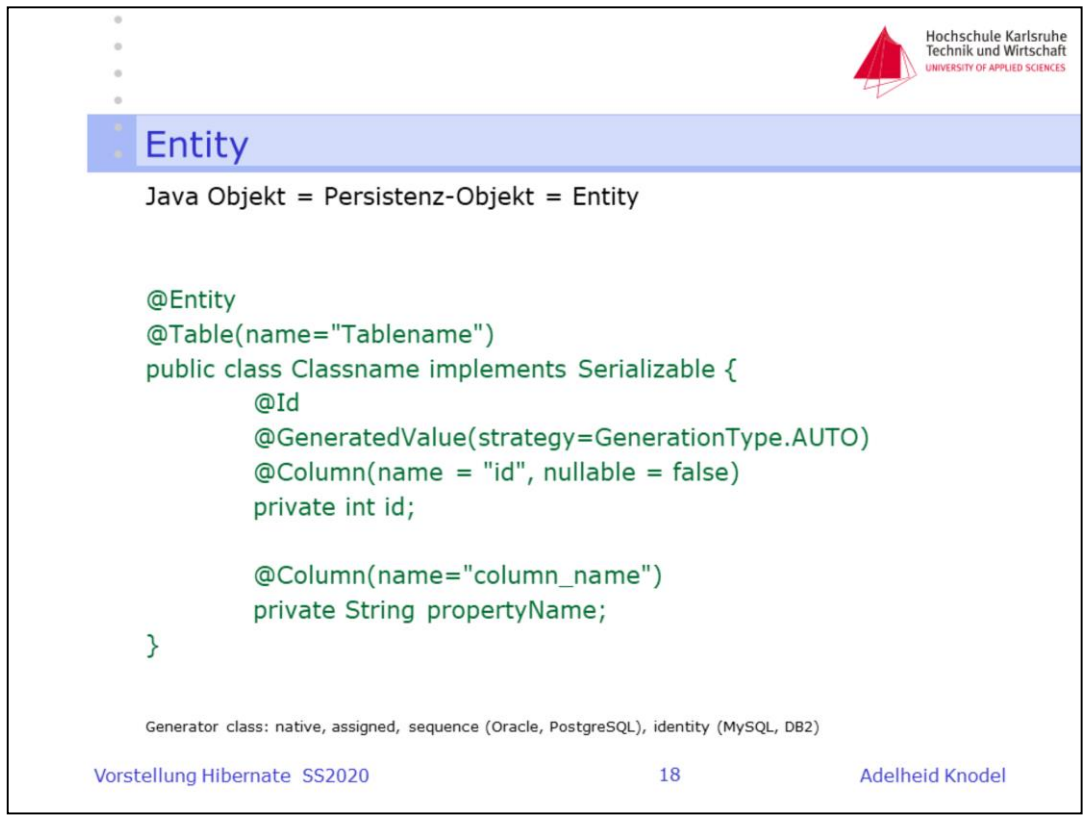
Zu jeder Property gibt es sowohl eine getter als auch eine setter Methode,

Über diese Zuordnung von Property Name und getter/ setter kann Hibernate die Daten aus den Java Objekten in die Tabellen Objekte schreiben, bzw. lesen



Um das O/R Mapping zu konfigurieren gibt es 2 Möglichkeiten, entweder über xml Files oder durch die Annotation der zu persistierenden Klassen.

Da im zu analysierenden Webshop die Variante der Annotation verwendet wird, beschränke ich die weiteren Erklärungen darauf.



Eine zu persistierende Klasse wird mit der Annotation **@Entity** versehen,

@Table damit kann der Tabellenname angegeben werden.

@Id kennzeichnet welche Property als Identifier verwendet werden soll,

@GeneratedValue damit kann festgelegt werden, wie der Identiier erzeugt werden soll

auto – die Strategie zur Erzeugung der Id wird aufgrund des Hibernate Dialekts gewählt

assigned – Id wird durch die Anwendung festgelegt, Anwendung muss sicherstellen, dass die Id eindeutig ist

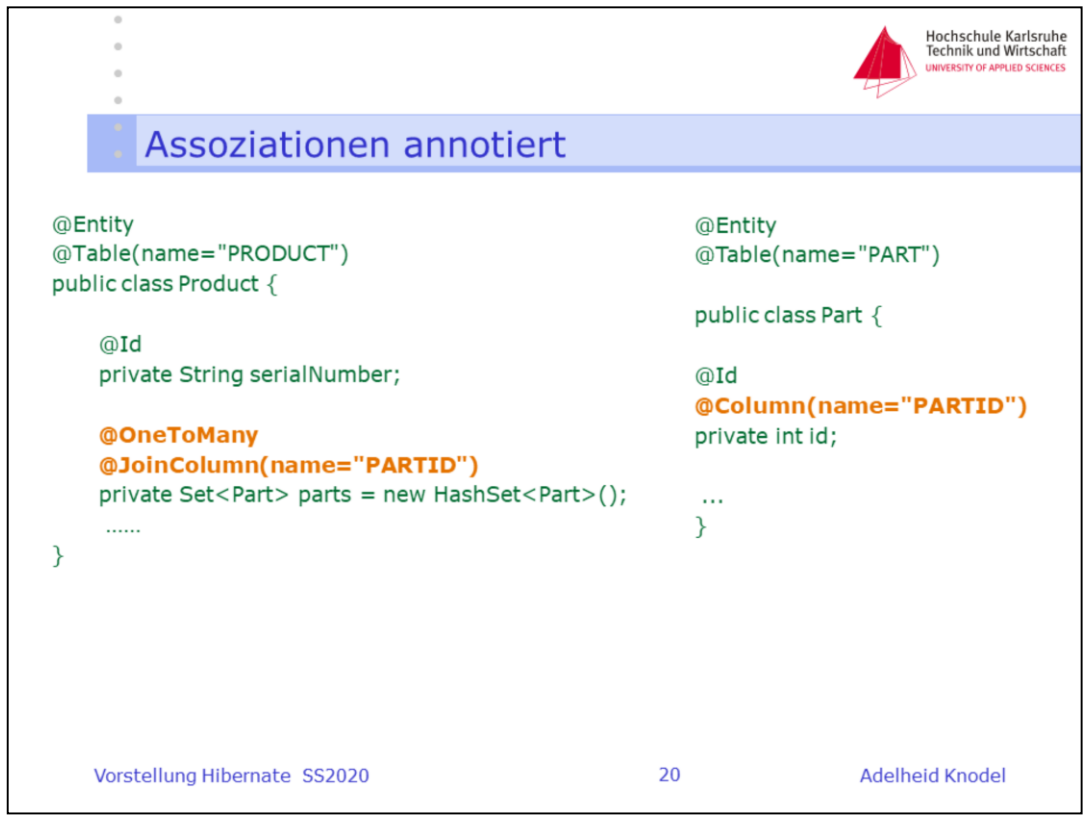
native – Generierung der Id wird anhand der verwendeten Datenbank gewählt

@Column kennzeichnet die Porty als zu persistierende Spalte, mit name kann ein Name für die Spalte in der Tabelle angegeben werden

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Beispiel annotierte Klasse

```
@Entity
@Table(name="CUSTOMER")
public class User  implements java.io.Serializable {

    @Id
    @Column(name="username")
    private String username;

    @Column(name="firstname")
    private String firstname;

    @Column(name="lastname")
    private String lastname;

    @Column(name="password")
    private String password;
}
```



Beziehungen von Klassen kann über Annotationen realisiert werden:

@OneToMany

@ManyToOne

@ManyToMany

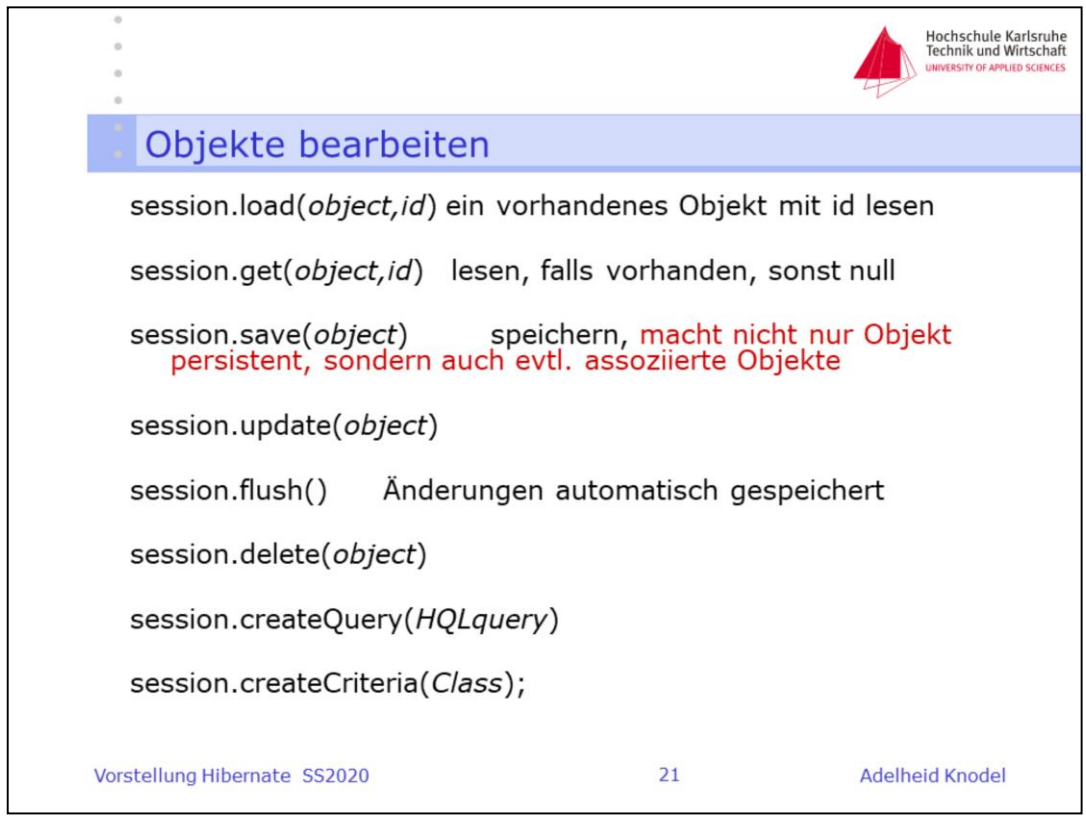
@OneToOne

Im obigen Beispiel bedeutet dies:

Ein Produkt hat eine Beziehung zu mehreren Teilen (OneToMany),

Die Property, die die Beziehung herstellt, wird mit @JoinColumn annotiert und hat den Namen PARTID.

Dieser Name muss in der Klasse Part als Spaltennamen existieren, damit die Assoziation korrekt aufgebaut werden kann.



Wie können jetzt Objekte bearbeitet werden:

Dafür stellt die Session Methoden zur verfügung.

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Objekt lesen (Beispiel)

Lesen von Objekten

```
public Customer getCustomerByPrimaryKey(String primaryKey) {   /* a
  Hibernate session */
  Session session =
        HibernateUtil.getSessionFactory().getCurrentSession();
  session.beginTransaction();
  Customer customer =
        (Customer) session.get(Customer.class, primaryKey);
  session.getTransaction().commit();
  return customer;  }
```


Wenn ein Objekt gelesen werden soll, ist es wichtig zu wissen, ist die Id bekannt, um auf das Objekt zugreifen zu können.

Falls die Id bekannt ist, kann mit der Methode

Session.get, der Angabe welcher Klasse das Objekt zugehört und der Id, das Objekt gelesen werden.

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Objekte speichern (Beispiel)

Speichern eines Objekts

```
public void saveCustomer(Customer customer) {
  /* a Hibernate session */

   Session session = HibernateUtil.getSessionFactory().getCurrentSession();

   session.beginTransaction();
        session.save(customer);
   session.getTransaction().commit();}
```

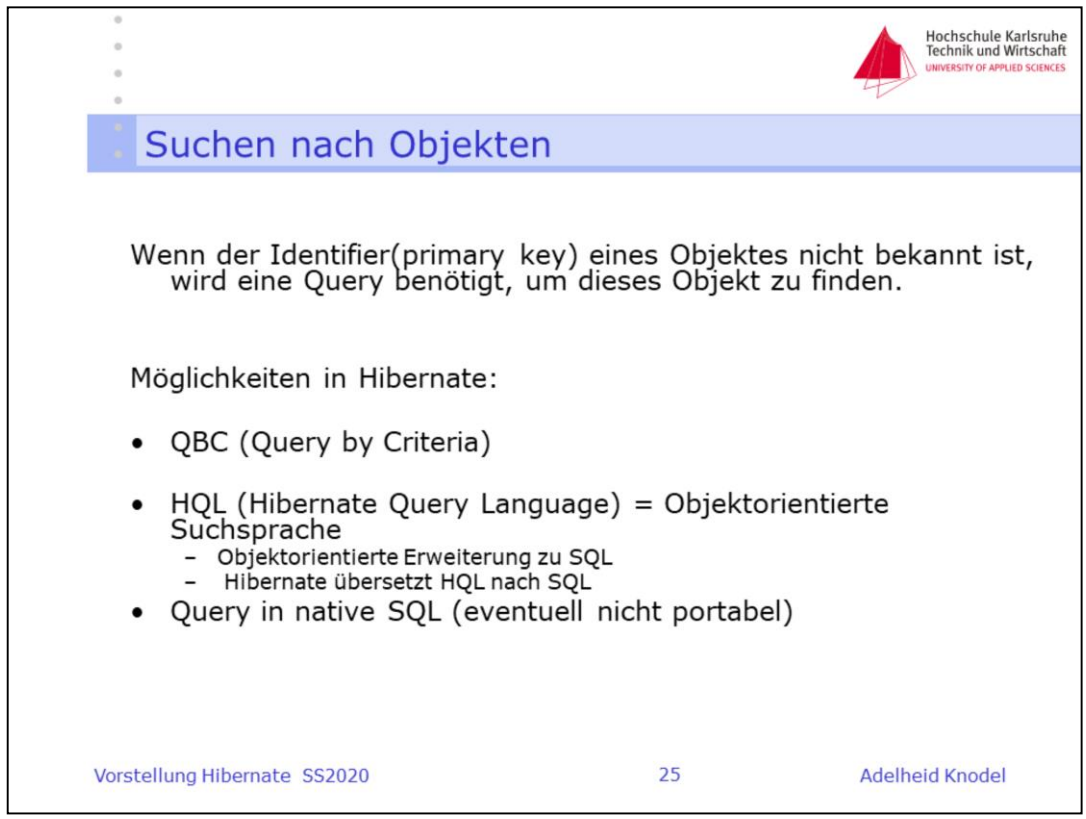

Mit Session.save wird ein neues Objekt abgespeichert,

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Objekte löschen (Beispiel)

```
Session session = getSessionFactory().openSession();

session.beginTransaction();
Customer customer =
        (Customer) session.get(Customer.class, primaryKey);

// deleting the customer
session.delete(customer);

session.getTransaction().commit();
```


Mit Session.delete wird das angegebene Objekt gelöscht



Wenn der Identifier eines Objekts nicht bekannt ist, muss mit einer Query das Objekt gesucht werden.

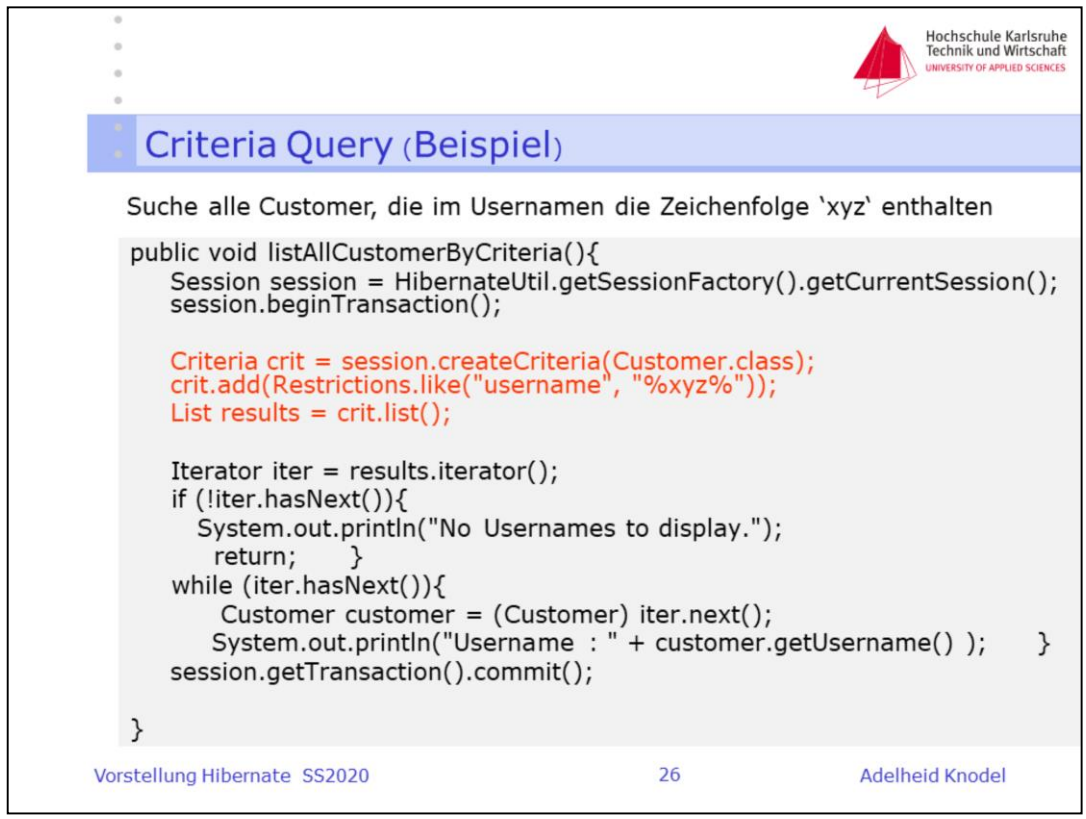
Für die Abfrage stehen drei Varianten zur Verfügung:

Die Objektorientierte Anfrage „Query by Criteria“ Api

Die Hibernate Query Language ist eine objektorientierte Erweiterung von SQL

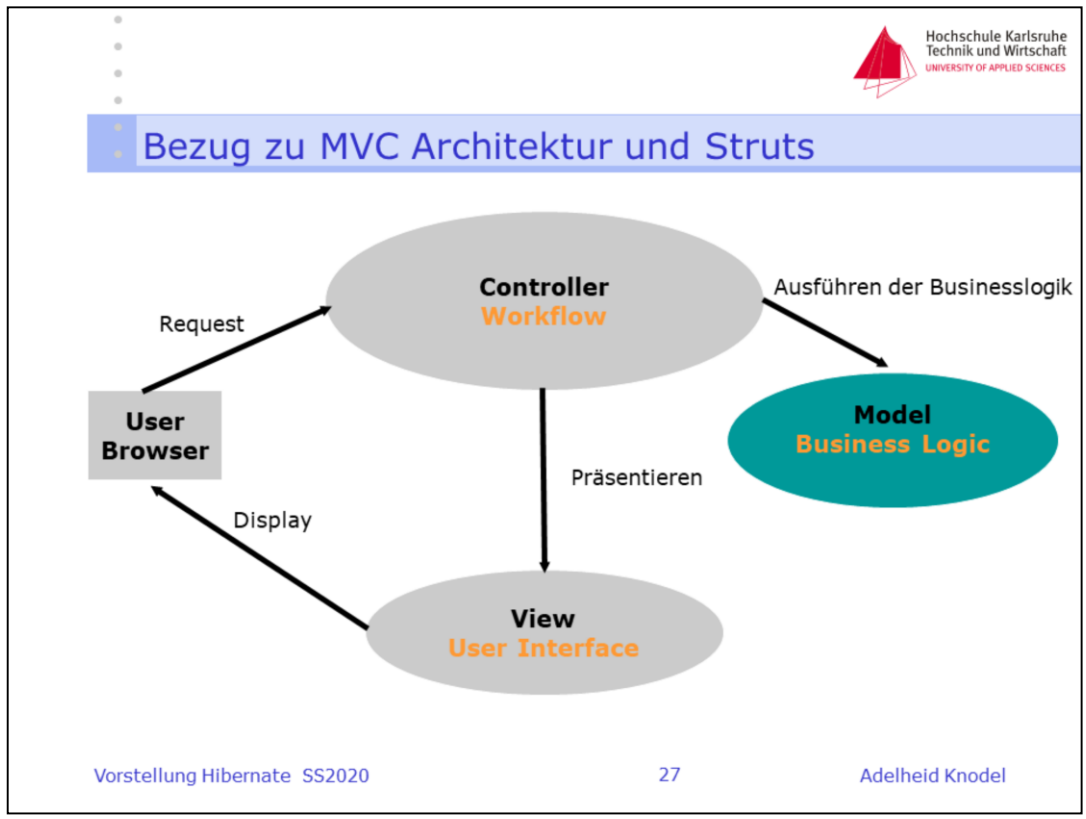
Oder es kann auch eine native SQL Anfrage verwendet werden, diese ist dann aber evtl. nicht portierbar auf andere Datenbanken

Für die Variante Query by Criteria wird auf der folgenden Folie ein Beispiel gezeigt.



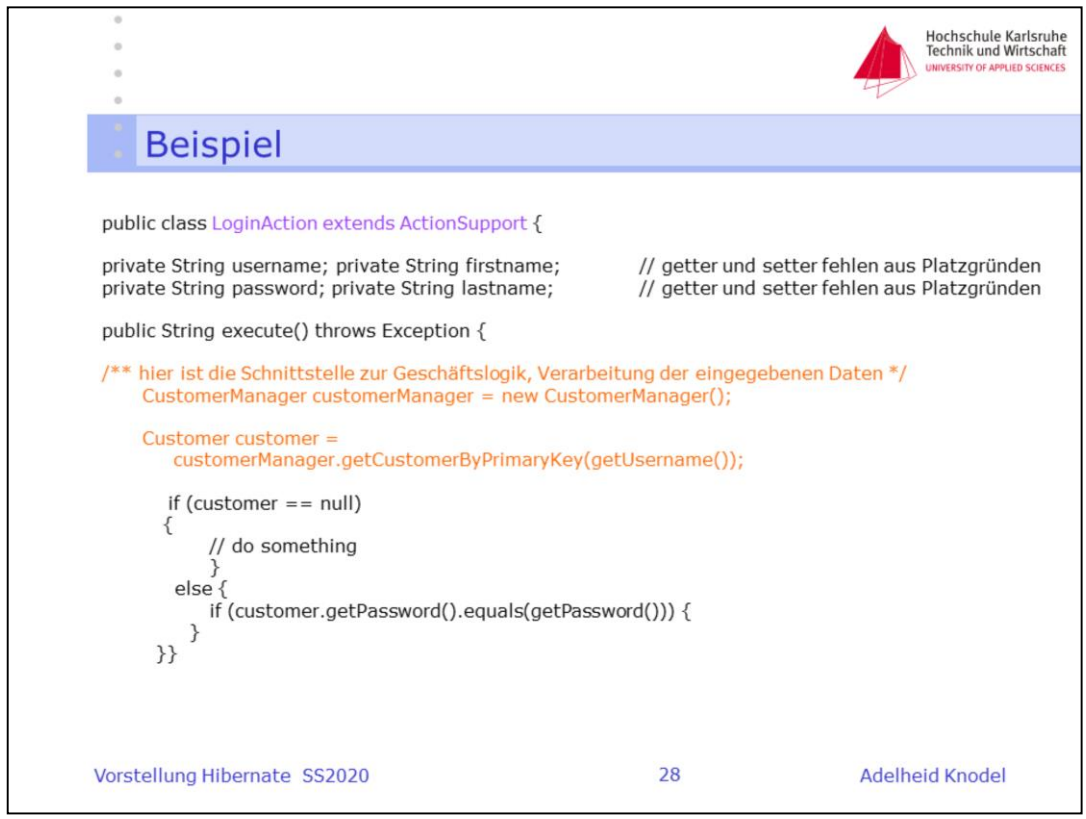
Über die Criteria Klasse werden die Kriterien für die Suche zusammengebaut,

Mit der Zeile crit.list() wird die Anfrage gestartet und das Ergebnis in results gespeichert.



Um den Bezug zur Aufgabe und zu Struts2 herzustellen, hier die Folie aus der Vorstellung des Frameworks Struts2.

Das Framework Hibernate kommt im Bereich Model, bzw. der Business Logic im Webshop zum Einsatz.



Sie erinnern sich an die ActionKlassen im Stuts2 Teil.

In diesen Klassen ist die Schnittstelle zur Businesslogik zu finden.

Die Businesslogik wurde in sogenannten ManagerKlassen implementiert.

In den Managerklassen werden Daten in die Datenbank geschrieben, bzw. müssen von dort gelesen werden und verarbeitet werden.

Manager-Klasse Businesslogik

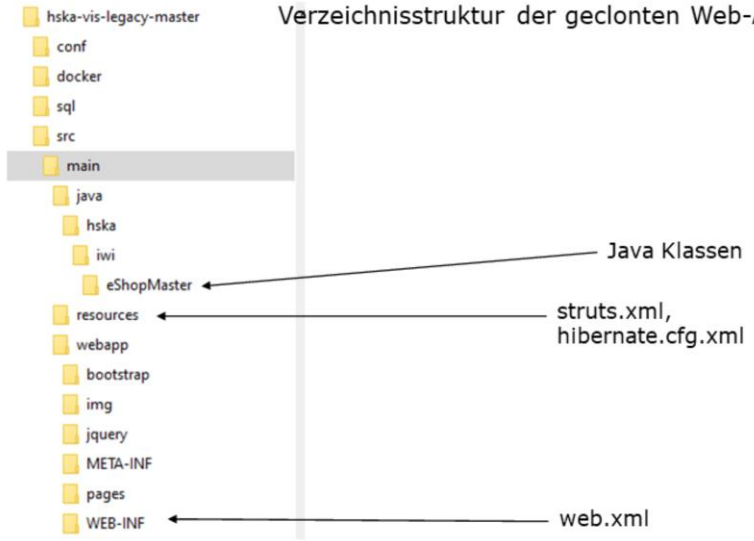
```
public Customer getCustomerByPrimaryKey(String primaryKey)
{

 /* a Hibernate session */
  Session session =
        HibernateUtil.getSessionFactory().getCurrentSession();

  session.beginTransaction();
  Customer customer =
        (Customer) session.get(Customer.class, primaryKey);
  session.getTransaction().commit();

  return customer;
 }
```


In der Klasse CustomerManger in der Methode getCustomerByPrimaryKey wird ein Objekt anhand des Primärschlüssels aus der Datenbank gelesen.

Alle weiteren Anfragen an die Datenbank sind in ähnlicher Form aufgebaut. Dies ist auch die Stelle, an der Anfragen an einen entfernten Service gestellt werden müssten.

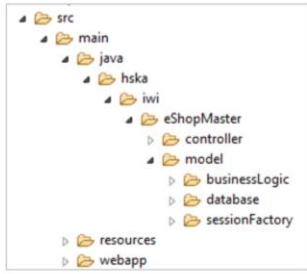
Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

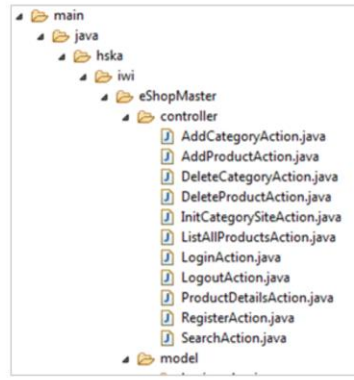
Verzeichnisstruktur des Projekts

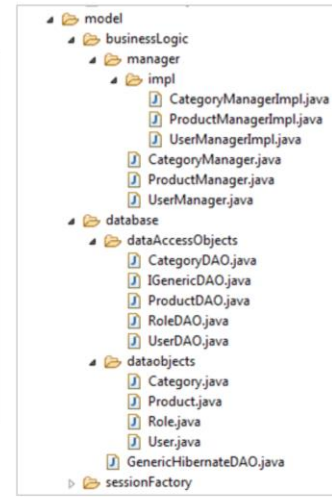


Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Struktur Java Klassen







Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Projekt URL

Clonen des Projekts:
https://github.com/mavogel/hska-vis-legacy

Bei Fragen schreiben Sie eine Email an mich adelheid.knodel@hs-karlsruhe.de und ich werde versuchen Ihre Fragen zu beantworten.