

Informationsintegration und Web-Portale

Tutorial:
Enterprise Java Beans

Erik Buchmann
buchmann@ira.uka.de



Benötigte Software

- installiertes JDK
 - übersetzen von EJB-Komponenten
- Datenbankserver mit JDBC-Treiber
 - Backend f. persistente Speicherung von Entity Beans
 - *MySQL 5.0.26 mit Connector 5.0.4*
- Applikationsserver
 - Container für EJB-Komponenten
 - *Sun App Server, Bestandteil des Java EE 5 SDK*
- Entwicklungsumgebung
 - Assistenten zum Deployment, DD erzeugen etc.
 - *Netbeans 5.0*

[Einführung](#)

[Vorarbeiten](#)

[Entity Beans](#)

[Session
Beans](#)

[Web-
Schnittstelle](#)

[Deployment](#)

[Schluss](#)





Allgemeine Vorgehensweise: Bottum-Up

1. Datenbank anlegen

- Datenbankschema, Schlüssel, Fremdschlüsselbez.

2. Entity Beans generieren

- Übername der Attribute aus dem Datenbankschema

3. Session Beans erzeugen

- Geschäftsmethoden definieren

4. Web-Applikation schreiben

- Oberfläche für den Anwender, Logik für Benutzerschnittstelle

5. Deployment

- .WAR-Archive erzeugen und auf dem App-Server installieren und registrieren

Einführung

Vorarbeiten

Entity Beans

Session Beans

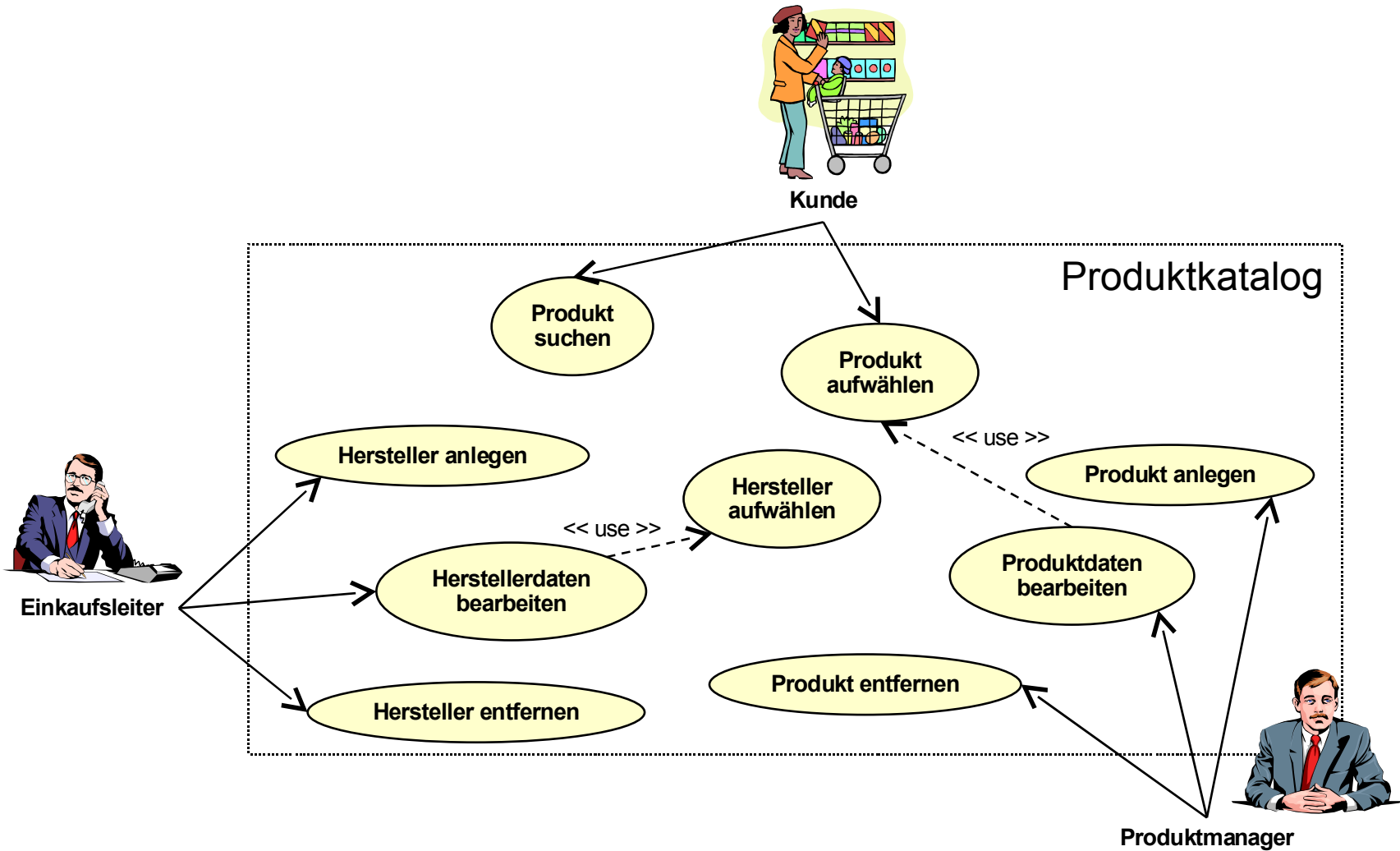
Web-Schnittstelle

Deployment

Schluss



Die Beispielanwendung





Aufsetzen der Datenbank: DB, User

```
buchmann@localhost:~> mysql -uroot -prootpw
```

```
# Datenbank anlegen
```

```
CREATE DATABASE EJBTUT;
```

```
# Nutzer anlegen
```

```
GRANT ALL PRIVILEGES ON EJBTUT.* TO ejbuser@localhost  
IDENTIFIED BY 'ejbpw';
```

```
# Nutzer aktivieren
```

```
FLUSH PRIVILEGES;
```

Einführung

Vorarbeiten

Entity Beans

Session
Beans

Web-
Schnittstelle

Deployment

Schluss





Aufsetzen der Datenbank: Schema

```
buchmann@localhost:~> mysql -uejbuser -pejbpw EJBTUT
```

```
CREATE TABLE MANUFACTURER (  
  ID      INTEGER      PRIMARY KEY,  
  NAME    VARCHAR(30)  NOT NULL,  
  URL     VARCHAR(200) NOT NULL  
);  
  
CREATE TABLE PRODUCT (  
  ID              INTEGER      PRIMARY KEY,  
  NAME            VARCHAR(30)  NOT NULL,  
  DESCRIPTION     VARCHAR(200) NOT NULL,  
  PRICE          DECIMAL(8,2) NOT NULL,  
  M_ID           INTEGER      NOT NULL,  
  FOREIGN KEY (M_ID) REFERENCES MANUFACTURER(ID)  
);
```

Einführung

Vorarbeiten

Entity Beans

Session
Beans

Web-
Schnittstelle

Deployment

Schluss



Vorarbeiten

- JDBC-Treiber in Netbeans registrieren
 - Runtime → Databases → Drivers: New Driver
 - Databases → mysql/JDBC: Connect
`jdbc:mysql://localhost/EJBTUT?user=ejbuser&password=ejbpw`
- Sun App-Server in Netbeans registrieren
 - Runtime → Servers: Add Server
- Neues Projekt anlegen
 - New Project → Enterprise → Enterprise Application
 - Name: „KlickAndBau“
 - Projektordner festlegen

Einführung

Vorarbeiten

Entity Beans

Session
Beans

Web-
Schnittstelle

Deployment

Schluss





Entity Beans generieren

- Übername der Attribute aus dem Datenbankschema
- im ejb-Teilprojekt:
 - New → CMP Entity Beans from Database
 - JDBC Connection: jdbc:mysql://localhost/EJBTUT
 - Package: „kub“
 - Tables: Add All

Einführung

Vorarbeiten

Entity Beans

Session
Beans

Web-
Schnittstelle

Deployment

Schluss





Was wurde generiert? (1)

- Home-Interface

- „Home“ ist hier der Container, d.h. der App-Server
→ Interface definiert Methoden zum Verwalten einer Bean
- allgemeine Methoden zum Anlegen und Zerstören einer Bean werden von *EJBLocalHome* geerbt
- Bean-spezifische Methoden zum Suchen anhand von Datenbank-Attributen werden im Interface selbst definiert

Einführung

Vorarbeiten

Entity Beans

Session
Beans

Web-
Schnittstelle

Deployment

Schluss





Was wurde generiert? (2)

- Local/LocalBusiness-Interfaces
 - Zugriff auf die Attribute der Entity
 - getter/setter-Methoden
 - für Relation *Manufacturer*:
 - *Integer getId()*
 - *String getName()* *void setName(String)*
 - *String getUrl()* *void setUrl(String)*

Einführung

Vorarbeiten

Entity Beans

Session
Beans

Web-
Schnittstelle

Deployment

Schluss





Was wurde generiert? (3)

- Die Bean-Klassen selbst
 - implementiert die Methoden des LocalBusiness-Interfaces (wenigstens *abstract*)
 - enthält eine Methode zum Erzeugen des Objekts sowie weitere Methoden, die zur EJB-Infrastruktur gehören

Einführung

Vorarbeiten

Entity Beans

Session
Beans

Web-
Schnittstelle

Deployment

Schluss





Was wurde generiert? (4)

- Deployment-Descriptor (*ejb-jar.xml*)

- allgemeine Angaben

- `<description>...<display-name>...<ejb-name>...`

- Persistenz

- `<ejb-ql>SELECT OBJECT(p) FROM Product AS p WHERE...`

- Transaktionen

- `<container-transaction>...</container-transaction>`

Einführung

Vorarbeiten

Entity Beans

Session
Beans

Web-
Schnittstelle

Deployment

Schluss





Session Beans erzeugen

- Abbildung von Geschäftsmethoden auf Geschäftsobjekte (die Entity Beans)
- im ejb-Teilprojekt:
 - New → Session Bean
 - Name „ViewSession“, Package: „kub“
 - Typ „Stateless“, nur Remote Interface

Einführung

Vorarbeiten

Entity Beans

Session
Beans

Web-
Schnittstelle

Deployment

Schluss





was wurde erzeugt?

- Remote-Interface
 - Interface ist noch leer, da noch keine Geschäftsmethoden definiert wurden
- die Bean selbst
 - ebenfalls noch leerer Methodenrumpf

Einführung

Vorarbeiten

Entity Beans

Session
Beans

Web-
Schnittstelle

Deployment

Schluss





Business Logic hinzufügen (1)

- Rechtsklick in die Klasse *ViewSessionBean*, Enterprise Resources → Call Enterprise Bean
 - Auswahl von *ProductEB*
→ generiert wurde Methode *lookupProductBean()*
- Attribut zur Klasse hinzufügen
 - `private kub.ProductLocalHome prodHome;`
- In die Init-Methode der Bean *ejbCreate()* den Lookup-Aufruf einbauen
 - `prodHome = lookupProductBean();`

Einführung

Vorarbeiten

Entity Beans

Session Beans

Web-Schnittstelle

Deployment

Schluss

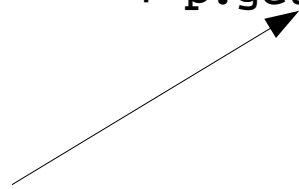




Business Logic hinzufügen (2)

- Rechtsklick, EJB Methods → Add Business Method

```
String showProduct(int id) throws FinderException {
    kub.ProductLocal p = prodHome.findByPrimaryKey(
        new Integer(id));
    return " Produkt: "      + p.getName()
        + " Beschreibung: " + p.getDescription()
        + " Preis: "        + p.getPrice()
        + " Hersteller: "   + p.getMId().getName();
}
```



*wir brauchen uns nicht mehr darum zu kümmern,
verschachtelte Objekte aufzulösen → EJB sucht
den Manufacturer aus der Fremdschlüsselbeziehung
in der Datenbank für uns!*

- Einführung
- Vorarbeiten
- Entity Beans
- Session Beans
- Web-Schnittstelle
- Deployment
- Schluss





Web-Schnittstelle hinzufügen (1)

- in unserem Beispiel: Servlet als Benutzerschnittstelle
→ Lokator zur Kommunikation zwischen Servlet und Session Bean erzeugen
- im web-Teilprojekt:
 - New → Enterprise → Service Lokator
 - Name „KubLocator“, Package „kub“

Einführung

Vorarbeiten

Entity Beans

Session Beans

Web-Schnittstelle

Deployment

Schluss





Web-Schnittstelle hinzufügen (2)

- Servlet anlegen
 - erzeugt den HTML-Code, der dem Webbrowser des Nutzers präsentiert wird
- im web-Teilprojekt:
 - New → Servlet
 - Name „ViewServlet“, Package „kub“
- generiert wurden
 - leeres Beispiel-Servlet
 - Deployment-Deskriptor (*web.xml*, *sun-web.xml*)
 - <context-root>, <servlet-name>, <url-pattern> etc.

Einführung

Vorarbeiten

Entity Beans

Session
Beans

Web-
Schnittstelle

Deployment

Schluss





Nutzerschnittstelle (1)

- Lookup-Methode für die Session Bean einfügen
 - Rechtsklick, Enterprise Resources → Call Enterprise Bean: *ViewSessionBean* auswählen, als Lokator *kub.KubLocator* eintragen

Einführung

Vorarbeiten

Entity Beans

Session
Beans

Web-
Schnittstelle

Deployment

Schluss





Nutzerschnittstelle (2)

- *processRequest()*: Webformular für Eingabedaten

```
out.println("<form>");
```

```
out.println("Product number:  
  <input type='text' name='prod_nr' />");
```

```
out.println("<input type=submit  
  value=Select />");
```

```
out.println("</form>");
```

Einführung

Vorarbeiten

Entity Beans

Session
Beans

Web-
Schnittstelle

Deployment

Schluss





Nutzerschnittstelle (3)

- *processRequest()*: Daten bzw. Fehler ausgeben

```
out.println("<h1>Servlet ViewServlet at " +  
    request.getContextPath() + "</h1>");
```

```
String nr = request.getParameter("prod_nr");
```

```
if (nr != null && !(nr.equals(""))) {  
    try {  
        kub.ViewSessionRemote vsr =  
            lookupViewSessionBean();  
        out.println("Produkt "+nr+" [ " +  
            vsr.showProduct(Integer.parseInt(nr)) + "]" );  
    } catch (Exception ex) {  
        out.println("Error with Id. " + nr + "<p>");  
        out.println(ex.toString());  
        StackTraceElement ste[] = ex.getStackTrace();  
        out.println("<p>");  
        for (int i = 0; i<ste.length; i++)  
            out.println(ste[i]+"<br>");  
    }  
}
```

Einführung

Vorarbeiten

Entity Beans

Session
Beans

Web-
Schnittstelle

Deployment

Schluss





Deployment

- Projekt kompilieren
- .war-Packages auf den Applikationsserver spielen
- JDBC-Treiber im App-Server registrieren
- Verbindung zwischen App-Server und DB herstellen
- referenzierte Namen dem App-Server bekannt machen (→ *Deployment Descriptor*)
- In Netbeans:
 - Rechtsklick auf Projekt → Clean and Build Project
 - Rechtsklick auf Projekt → Deploy Project
 - Rechtsklick auf Projekt → Run Project

Einführung

Vorarbeiten

Entity Beans

Session
Beans

Web-
Schnittstelle

Deployment

Schluss





Wenn alles geklappt hat:

- Einführung
- Vorarbeiten
- Entity Beans
- Session Beans
- Web-Schnittstelle
- Deployment
- Schluss

Servlet ViewServlet at /KlickAndBau-WebModule

Produkt 1 [Produkt: Saege Beschreibung: Fuer zu lange Bretter Preis: 8.99 Hersteller: Eisenkarl]

Product number:

