Analyse und Modellierung von Informationssystemen

Dr. Klaus Höppner

Hochschule Darmstadt – Wintersemester 2014/2015

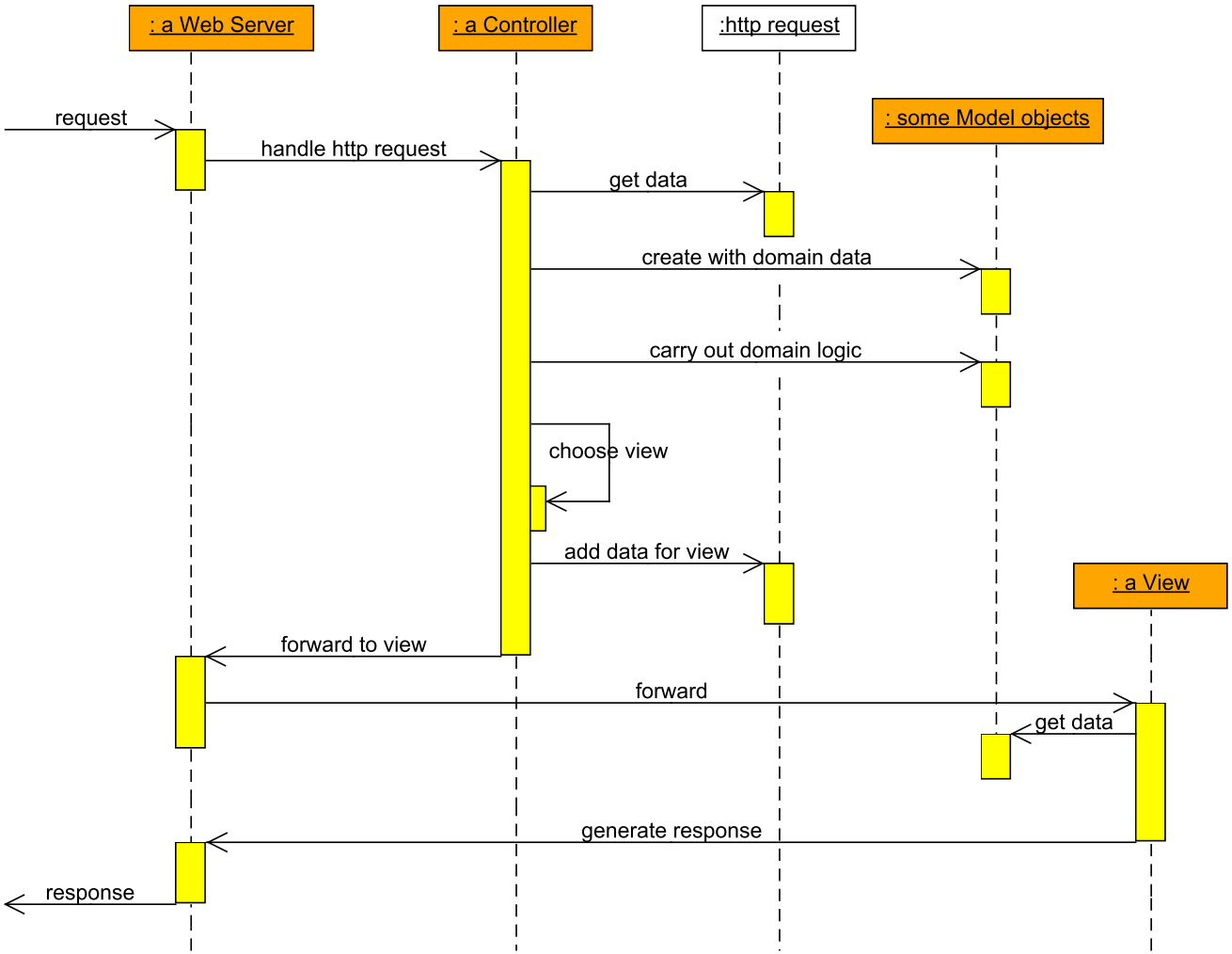
Web-Anwendung mit MVC

MVC bei Webanwendungen

Bei Webanwendungen findet sich häufig das MVC-Pattern. Bei Java-Webapplikationen sind die drei Schichten oft sehr deutlich voneinander separiert:

- Der Controller läuft als Java-Servlet in einem Applikations-Server (z. B. Apache Tomcat),
- dieser greift auf das Datenmodell zu, dass in konventionellen Java-Klassen implementiert ist,
- das Ergebnis wird an eine *Java Server Page* (JSP) als Präsentationsschicht weitergeleitet.

Sequenzdiagramm



Schritt 1: Das Datenmodell

Zunächst werden die Klassen `Abteilung` und `Person` über JPA persistent gemacht:

```
@Entity
public class Abteilung {
    private Integer id;
    private String name;
    private List<Person> personen = new ArrayList<Person>();

    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }
```

Das Datenmodell (Forts.)

```
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @OneToMany(mappedBy="abteilung")
    public List<Person> getPersonen() {
        return personen;
    }

    public void setPersonen(List<Person> personen) {
        this.personen = personen;
    }
}
```

Das Datenmodell (Forts.)

```
@Entity
public class Person {
    private Integer id;
    private String vorname;
    private String nachname;
    private Abteilung abteilung;

    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getVorname() {
        return vorname;
    }
    public void setVorname(String vorname) {
        this.vorname = vorname;
    }
```

Das Datenmodell (Forts.)

```
    public String getNachname() {
        return nachname;
    }
    public void setNachname(String nachname) {
        this.nachname = nachname;
    }
    @ManyToOne
    @JoinColumn(name="abteilung_id")
    public Abteilung getAbteilung() {
        return abteilung;
    }
    public void setAbteilung(Abteilung abteilung) {
        this.abteilung = abteilung;
    }
    @Override
    public String toString() {
        return String.format("%s %s (%s)",
            getVorname(), getNachname(),
            getAbteilung()==null ? "--" : getAbteilung().getName());
    }
}
```

Persistenz mit Hybernate und MySQL

```
<?xml version="1.0" encoding="utf-8" ?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java
    version="2.0">
    <persistence-unit name="manager1" transaction-type="RESOURCE_LOCAL">
        <properties>
            <property name="javax.persistence.jdbc.driver"
                value="com.mysql.jdbc.Driver" />
            <property name="javax.persistence.jdbc.user" value="root" />
            <property name="javax.persistence.jdbc.password" value="" />
            <property name="javax.persistence.jdbc.url"
                value="jdbc:mysql://localhost:3306/employee" />
            <property name="hibernate.dialect"
                value="org.hibernate.dialect.MySQL5InnoDBDialect" />
            <property name="hibernate.max_fetch_depth" value="3" />
            <property name="hibernate.show_sql" value="true" />
        </properties>
    </persistence-unit>
</persistence>
```

Schritt 2: Servlet als Controller

```
public class PersonenQuery extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        EntityManager em = Connection.getEm();

        RequestDispatcher result;
        List<Person> personen =
                em.createQuery("select p from Person p",
                Person.class).getResultList();
        req.setAttribute("personen", personen);
        result = req.getRequestDispatcher("personen-ergebnis.jsp");
        result.forward(req, resp);
    }
}
```

EntityManager als Singleton

```
public class Connection {
   private static EntityManager em;

   public static EntityManager getEm() {
      if (em==null) {
         synchronized (Connection.class) {
            if (em==null) {
               EntityManagerFactory emf =
                  Persistence.createEntityManagerFactory("manager1");
               em = emf.createEntityManager();
            }
         }
      }
      return em;
   }
}
```

Schritt 3: JSP als View

```
<%@page import="java.util.List, data.Person" %>
<html>
<head>
<title>Personenliste: Ergebnis</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
</head>

<body>
<h1>Personen</h1>
<ul>

<% for (Person p : (List<Person>) request.getAttribute("personen") ) { %>
<li>
   <%= p.toString() %>
   (<a href="person-edit?id=<%= p.getId() %>")">bearbeiten</a>)
</li>
<% } %>

</ul>
</body>
</html>
```

Weiteres Beispiel: Formular Personendaten

```
public class PersonenEdit extends HttpServlet {
   @Override
   protected void doGet(HttpServletRequest req, HttpServletResponse resp)
         throws ServletException, IOException {
      EntityManager em = Connection.getEm();
      String path = req.getRequestURI();

      RequestDispatcher result;
      String action = req.getParameter("action");
      if ("submit".equals(action)) {
         String idStr = req.getParameter("id");
         Integer id = idStr==null ? null : Integer.valueOf(idStr);
         EntityTransaction tx = em.getTransaction();
         tx.begin();
         Person p = id==null ? new Person() : em.find(Person.class, id);
         p.setVorname(req.getParameter("vorname"));
         p.setNachname(req.getParameter("nachname"));
         Integer abteilung_id = Integer.valueOf(req.getParameter("abteilun
         p.setAbteilung(em.find(Abteilung.class, abteilung_id));
         if (id==null) em.persist(p);
         tx.commit();
         result = req.getRequestDispatcher("personen");
```

Formular Personendaten (Forts.)

```
        } else {
            String idStr = req.getParameter("id");
            Integer id = idStr==null ? null : Integer.valueOf(idStr);
            List<Abteilung> abteilungen =
                em.createQuery("select a from Abteilung a order by a.name",
                Abteilung.class).getResultList();
            result = req.getRequestDispatcher("person-edit.jsp");
            req.setAttribute("action", path);
            req.setAttribute("abteilungen", abteilungen);
            if (id!=null)
                req.setAttribute("person", em.find(Person.class, id));
        }
        result.forward(req, resp);
    }
}
```

JSP zum Personen-Formular

```
<%@page import="java.util.List, data.*" %>
<% Person p = (Person) request.getAttribute("person"); %>

<html>
<head>
<title>Personen bearbeiten</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
</head>

<body>
<h1>Person bearbeiten</h1>

<form action="<%= request.getAttribute("action") %>" method="get">
<table>
<tr>
   <td>Vorname:</td>
   <td><input type="text" name="vorname"
      value="<%= p==null ? "" : p.getVorname() %>"></td>
</tr>
```

JSP zum Personen-Formular (Forts.)

```
<tr>
   <td>Nachname:</td>
   <td><input type="text" name="nachname"
      value="<%= p==null ? "" : p.getNachname() %>"></td>
</tr>
<tr>
   <td>Abteilung:</td>
   <td>

<select name="abteilung">
<% for (Abteilung a : (List<Abteilung>) request.getAttribute("abteilungen"
   <option value="<%= a.getId() %>"
      <%= a.getId().equals(p==null ? null : p.getAbteilung().getId()) ?
         "selected=\"selected\"" : "" %>><%= a.getName() %></option>
<% } %>
</select>
   </td>
</tr>
</table>
```

JSP zum Personen-Formular (Forts.)

```
<input type="submit" />
<% if (p!=null) { %>
   <input type="hidden" name="id" value="<%= p.getId() %>">
<% } %>
<input type="hidden" name="action" value="submit">
</form>

</body>
</html>
```