



## Java EE – Übung 2

Eine einfache Applikation zum Verwalten und Überwachen von Smart Meters

---



## Group 46

- Carolin Schwarz, 371802
- Fedor Vitkovskiy, 386458
- Robert Koch, 386471
- Jia Fug Liu, 382333



# Content

## Struktur des Projekts

- EJB Models
- Daos
- Servlets

## Detailansichten der einzelnen Klassen mit Code

- EJB Models
- Daos
- Servlets

## JSPs



## Struktur des Projekts – EJB Models

- User:                   Zusätzlich zu den vorgegebenen Methoden enthält die Klasse User Methoden die Formulare zum Login und Logout von Usern bereitstellen, sowie zum Hinzufügen von Smartmetern und Ablesungen.
- SmartMeter:           Die SmartMeter-Klasse enthält die Eigenschaften der Smartmeter und eine Liste von Ablesungen(Records). Die Smartmeter werden als Entities gespeichert und stehen zu den Records in einer OneToMany-Beziehung. Zudem werden hier die Zufallswerte für die Spannung und Stromstärke berechnet und eine optionale Warnung erzeugt.
- Record:                Die Records werden ebenfalls als Entities gespeichert und stehen zu den Smartmetern in einer ManyToOne-Beziehung. Die Zeitpunkte der Ablesungen werden im Konstruktor mitgespeichert.



## Struktur des Projekts - Daos

- UserDao: Der UserDao kann neue User in der Usertabelle speichern, User finden, eine Liste aller User zurückgeben sowie das letzte Logindatum aktualisieren.
- SmartMeterDao: Der SmartMeterDao kann neue Smartmeter in der Smartmetertabelle speichern, Smartmeter finden und eine Liste aller Smartmeter zurückgeben.
- RecordDao: Der RecordDao kann neue Records in der Recordtabelle speichern und eine Liste aller Records zurückgeben.



## Struktur des Projekts - Servlets

**SmartMeterServlet:** Das SmartMeterServlet übernimmt die Aufgabe, die Liste von angelegten Smartmeters anzuzeigen. Zudem wird hier die Erstellung von neuen Smartmetern veranlasst.

**LoginServlet:** Setzt den Login um, zudem können von hier aus neue User angelegt werden.

**LogoutServlet:** Das LogoutServlet kümmert sich um den Logout von Usern.

**DetailServlet:** Das DetailServlet übernimmt die Darstellung eines spezifischen Smartmeters mit dessen Attributen. Es prüft zudem, ob ein User angemeldet ist. Ist dies der Fall kann der User neue Ablesungen vornehmen, welche mit Hilfe der EJBs erstellt und gespeichert werden.



## Detailansicht - User

In der User-Klasse wurde ein persistentes Attribut lastLogin hinzugefügt und ein nicht persistentes Feld, in dem die aktuelle Smartmeter id als Hilfsvariable eingetragen wird.

```
1 package de.tub.as.smm.models;
2
3+ import java.io.Serializable;
13
14 @Entity
15 public class User implements Serializable {
16
17     private static final long serialVersionUID = 5488382585787964091L;
18     // persistent fields
19     private Long id;
20     private String name;
21     private Date lastLogin;
22     // non persistent field:
23     private Long smartmeterId;
24
25     // constructors
26- public User() {
27     }
28
29- public User(String name) {
30     this.name = name;
31     this.lastLogin = new Date(System.currentTimeMillis());
32 }
33
```



Zur Anmeldung der User wird eine Loginform verwendet.

Es gibt eine addDevicesform, um neue Smartmeters einzutragen.

```
77 // forms
78 public static String loginForm() {
79     return "<br />" +
80         "<h2>Login</h2>" +
81         "<hr />" +
82         "<table>" +
83         "<tr>" +
84         "<td><img src=\"./media/unlocked.png\" height=\"200\" width=\"200\"></img></td>" +
85         "<td>Um neue Geräte hinzufügen und Ablesungen vornehmen zu können, müssen Sie sich anmelden." +
86         "<br />" +
87         "<br />" +
88         "<form method=\"POST\" action=\"login\">" +
89         "<b>Nutzerkennung:</b>" +
90         "<input pattern=\".{3,}\" required title=\"Mindestens 3 Zeichen\" type=\"text\" name=\"user\" />" +
91         "<input class=\"inButton\" type=\"submit\" value=\"Anmelden\" />" +
92         "</form>" +
93         "</td>" +
94         "</tr>" +
95         "</table>";
96
97 }
98
99 @Transient
100 public String addDevices() {
101     return "<br />" +
102         "<h2>Geräte hinzufügen</h2>" +
103         "<hr />" +
104         "<form method=\"POST\" action=\"verwalten\">" +
105         "<b>Geräteerkennung:</b>" +
106         "<input pattern=\"[A-Z]{2}[0-9]{8}\" required title=\"Zwei Großbuchstaben gefolgt von 8 Zahlen\" type=\"text\" name=\"geraetekennung\" />" +
107         +
108         "<b>Maximale Belastung</b> (in Ampere): " +
109         "<input required type=\"number\" min=\"50\" max=\"100\" name=\"maxBelastung\" />" +
110         "<input class=\"inButton\" type=\"submit\" value=\"Hinzufügen\" />" +
111         "</form>";
112 }
```





Zudem gibt es natürlich auch eine Logoutform und eine Ableseform um Ablesungen einzutragen.

```

114 @Transient
115 public String logout() {
116     return "<br />" +
117         "<h2>Logout</h2>" +
118         "<hr />" +
119         "<table>" +
120         "<tr>" +
121         "<td><img src=\"./media/locked.png\" height=\"200\" width=\"200\"></img></td>" +
122         "<td>Wenn Sie sich abmelden, können Sie keine neuen Geräte hinzufügen oder Ablesungen vornehmen." +
123         "<br />" +
124         "<br />" +
125         "Sie sind eingeloggt als: <b>" + this.getName() + "</b>" +
126         "<br />" +
127         "Zuletzt eingeloggt am: <b>" + this.getFormattedLastLogin() + "</b>" +
128         "<form method=\"POST\" action=\"logout\">" +
129         "<input type=\"hidden\" name=\"logout\" value=\"\" + this.getId() + "\" />" +
130         "<br />" +
131         "<input class=\"inButton\" type=\"submit\" value=\"Abmelden\" />" +
132         "</form>" +
133         "</td>" +
134         "</tr>" +
135         "</table>";
136 }
137
138 @Transient
139 public String ablesenForm() {
140     return "<br />" +
141         "<h2>Ablesen</h2>" +
142         "<hr />" +
143         "<b>Nutzerkennung:</b>" +
144         "<br />" +
145         "Sie sind eingeloggt als: <b>" + this.getName() + "</b>" +
146         "<br />" +
147         "<br />" +
148         "<form method=\"POST\" action=\"detail\">" +
149         "<input type=\"hidden\" name=\"user\" value=\"\" + this.getName() + "\">" +
150         "<input type=\"hidden\" name=\"id\" value=\"\" + this.getSmartmeterId() + "\">" +
151         "<b>Verbrauchswert:</b>" +
152         "<input required type=\"number\" name=\"verbrauchswert\" min=\"0\" />" +
153         "<input class=\"inButton\" type=\"submit\" value=\"Ablesen\" />" +
154         "</form>";
155 }

```



## Detailansicht - Smartmeter

Die Smartmeter verfügen über die persistent gespeicherten Attribute Id, geraetekenennung, maxBelastung und eine Liste von smartmeterRecords. Dazu kommen noch die nicht persistenten Attribute spannung und strom, die bei jedem Aufruf neu berechnet werden.

```
1 package de.tub.as.smm.models;
2
3 import java.io.Serializable;
4
15
16 @Entity
17 public class SmartMeter implements Serializable {
18
19     private static final long serialVersionUID = -2250177305743873329L;
20     // persistent fields
21     private Long id;
22     private String geraetekenennung;
23     private double maxBelastung;
24     private List<Record> smartmeterRecords;
25     // non persistent fields
26     private double spannung;
27     private double strom;
28
29     // constructors
30     public SmartMeter() {
31     }
32
33     public SmartMeter(String geraetekenennung, double maxBelastung) {
34         this.geraetekenennung = geraetekenennung;
35         this.maxBelastung = maxBelastung;
36         this.smartmeterRecords = new ArrayList<Record>();
37     }
38
39     @Id
40     @GeneratedValue(strategy = GenerationType.IDENTITY)
41     public Long getId() {
42         return id;
43     }
44 }
```

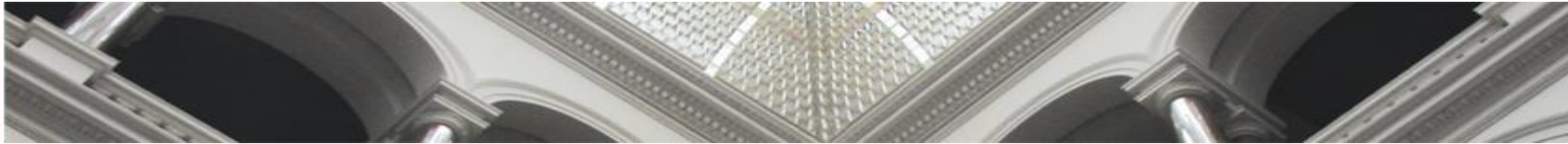


Die Smartmeter stehen in einer OneToMany Beziehung zu den Records.

Bei Aufruf des Smartmeters im Servlet werden neue Zufallswerte für Spannung und Strom berechnet.

In currentStatus wird geprüft, ob die maxBelastung überschritten wurde und eine entsprechende Rückgabe erzeugt.

```
@OneToMany(cascade = { CascadeType.PERSIST,  
    CascadeType.MERGE }, targetEntity = Record.class, mappedBy = "smartmeter", fetch = FetchType.EAGER)  
public List<Record> getSmartmeterRecords() {  
    return smartmeterRecords;  
}  
  
public void setSmartmeterRecords(List<Record> smartmeterRecords) {  
    this.smartmeterRecords = smartmeterRecords;  
}  
  
@Transient  
public double getSpannung() {  
    return spannung;  
}  
  
public void randomSpannung() {  
    this.spannung = Math.round((Math.random() * 20 + 220) * 10.0) / 10.0;  
}  
  
@Transient  
public double getStrom() {  
    return strom;  
}  
  
public void randomStrom() {  
    this.strom = Math.round((Math.random() * (maxBelastung + 5)) * 10.0) / 10.0;  
}  
  
@Transient  
public String currentStatus() {  
    if (strom > maxBelastung)  
        return "<div class=\"warning\"><h3 class=\"warningHead\">WARNUNG:</h3>Die Stromstärke liegt oberhalb der zulässigen Maximalbelastung!</div>";  
    else  
        return "<div class=\"info\"><h3 class=\"infoHead\">Aktueller Status:</h3>Alles läuft nach Plan: Keine weiteren Informationen vorhanden.</div>";  
}
```



## Detailansicht - Records

Die Records verfügen über die persistenten Attribute id, (das zugehörige) smartmeter, den Namen user des Ablesenden, den Ablesewert record und den Zeitpunkt date. Der Ablesezeitpunkt wird im Konstruktor gesetzt.

Die Records stehen in einer ManyToOne Beziehung zu ihrem Smartmeter.

```
17 @Entity
18 public class Record implements Serializable {
19
20     private static final long serialVersionUID = -4497369383015167908L;
21     // persistent fields
22     private Long id;
23     private SmartMeter smartmeter;
24     private String user;
25     private double record;
26     private Date date;
27
28     // constructors
29     public Record() {
30     }
31
32     public Record(SmartMeter smartmeter, String user, double record) {
33         this.smartmeter = smartmeter;
34         this.user = user;
35         this.record = record;
36         this.date = new Date(System.currentTimeMillis());
37     }
38
39     @Id
40     @GeneratedValue(strategy = GenerationType.IDENTITY)
41     public Long getId() {
42         return id;
43     }
44
45     public void setId(Long id) {
46         this.id = id;
47     }
48
49     @ManyToOne(cascade = { CascadeType.PERSIST,
50                     CascadeType.MERGE }, targetEntity = SmartMeter.class, fetch = FetchType.EAGER)
51     @JoinColumn(name = "SMARTMETER_ID", nullable = false)
52     public SmartMeter getSmartmeter() {
53         return smartmeter;
54     }
55 }
```

## Detailansicht - UserDao

Der UserDao kann User speichern, finden, ihren letzten Login aktualisieren und alle User zurückgeben.

```
1 package de.tub.as.smm.dao;
2
3 import java.util.Date;
4
5 /**
6  * Session Bean implementation class UserDaoEJB
7  */
8 @Stateless
9 public class UserDao {
10
11     // injected database connection
12     @PersistenceContext
13     private EntityManager em;
14
15     // store a new user
16     public void persist(User user) {
17         em.persist(user);
18     }
19
20     // retrieve all the users
21     public List<User> getAllUsers() {
22         TypedQuery<User> query = em.createQuery("SELECT u FROM User u ORDER BY u.id", User.class);
23         return query.getResultList();
24     }
25
26     // find a user
27     public List<User> findUserByName(String name) {
28         TypedQuery<User> query = em.createQuery("SELECT u FROM User u WHERE u.name LIKE :userName", User.class)
29             .setMaxResults(1).setParameter("userName", name);
30         return query.getResultList();
31     }
32
33     // update last login date
34     public void lastLogin(Long userId) {
35         em.createQuery("UPDATE User SET lastLogin = :date WHERE id LIKE :userId")
36             .setParameter("userId", userId).setParameter("date", new Date(System.currentTimeMillis()))
37             .executeUpdate();
38     }
39 }
```



## Detailansicht - SmartMeterDao

Der SmartMeterDao kann Smartmeter speichern, finden und eine Liste aller Smartmeter zurückgeben.

```

1 package de.tub.as.smm.dao;
2
3 import java.util.List;
4
11 /**
12  * Session Bean implementation class SmartMeterDaoEJB
13  */
14 @Stateless
15 public class SmartMeterDao {
16
17     // injected database connection
18     @PersistenceContext
19     private EntityManager em;
20
21     // store a new smart meter
22     public void persist(SmartMeter smartMeter) {
23         em.persist(smartMeter);
24     }
25
26     // retrieve all smart meters
27     public List<SmartMeter> getAllSmartMeters() {
28         TypedQuery<SmartMeter> query = em.createQuery("SELECT u FROM SmartMeter u ORDER BY u.id", SmartMeter.class);
29         return query.getResultList();
30     }
31
32     public SmartMeter findSmartmeterById(Long smartmeterId) {
33         TypedQuery<SmartMeter> query = em
34             .createQuery("SELECT u FROM SmartMeter u WHERE u.id LIKE :sId", SmartMeter.class)
35             .setParameter("sId", smartmeterId).setMaxResults(1);
36         return query.getSingleResult();
37     }
38 }
39
40 }

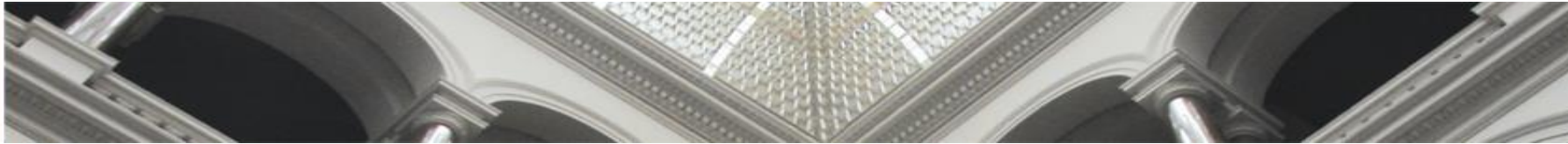
```



## Detailansicht - RecordDao

Der RecordDao kann Records speichern und eine Liste aller Ablesungen zurückgeben.

```
1 package de.tub.as.smm.dao;
2
3 import java.util.List;
4
5 /**
6  * Session Bean implementation class RecordDao
7  */
8 @Stateless
9 public class RecordDao {
10
11     // injected database connection
12     @PersistenceContext
13     private EntityManager em;
14
15     // store a new record
16     public void persist(Record record) {
17         em.merge(record);
18     }
19
20     // retrieve all records
21     public List<Record> getAblesungen() {
22         TypedQuery<Record> query = em.createQuery("SELECT u FROM Record u ORDER BY u.id DESC", Record.class);
23         return query.getResultList();
24     }
25 }
```



## Detailansicht - SmartMeterServlet

In der doGet() Methode wird ein User als Sessionattribut gesetzt, sofern jemand eingeloggt ist. Zudem wird die Liste der vorhandenen Smartmeters vom SmartmeterDao angefordert und ausgegeben.

```
1 package de.tub.as.smm;
2
3 import java.io.IOException;
4
5 /**
6  * Servlet implementation class SmartMeterServlet
7  */
8 @WebServlet("/verwalten")
9 public class SmartMeterServlet extends HttpServlet {
10     private static final long serialVersionUID = 1L;
11
12     // Injected DAO EJB:
13     @EJB
14     SmartMeterDao smartmeterDao;
15     @EJB
16     UserDao userDao;
17
18     /**
19      * display the list of smart meters
20      */
21     @Override
22     protected void doGet(
23         HttpServletRequest request, HttpServletResponse response)
24         throws ServletException, IOException {
25
26         // set a single user as the attribute if someone is logged in
27         HttpSession session = request.getSession(false);
28         if (session != null) {
29             User user = (User) session.getAttribute("user");
30             request.setAttribute("user", user);
31         }
32
33         // set a list of all smart meters as the attribute
34         request.setAttribute("smartmeter", smartmeterDao.getAllSmartMeters());
35
36         // display the list of smart meters
37         request.getRequestDispatcher("/verwalten.jsp").forward(request, response);
38     }
39 }
```



In der doPost() Methode wird die Erstellung von neuen Smartmetern veranlasst. Mit den vom User eingegebenen Werten Gerätekennung und maxBelastung wird ein neues Smartmeter Objekt erstellt und dem SmartmeterDao zum Speichern übergeben.

```
--
53  /**
54   * method is called if a new smart meter or user is created or an user
55   * logged in
56   */
57  @Override
58  protected void doPost(
59      HttpServletRequest request, HttpServletResponse response)
60      throws ServletException, IOException {
61
62      // get the current session
63      HttpSession session = request.getSession();
64
65      // create a new smart meter
66      if (request.getParameter("geraetekennung") != null && request.getParameter("maxBelastung") != null) {
67          String geraetekennung = request.getParameter("geraetekennung");
68          double maxBelastung = Double.parseDouble(request.getParameter("maxBelastung"));
69          SmartMeter smartmeter = new SmartMeter(geraetekennung, maxBelastung);
70          smartmeterDao.persist(smartmeter);
71      }
```



AB12345678

[Zur Detailansicht](#)

KA93726312

[Zur Detailansicht](#)

## Geräte hinzufügen

Gerätekennung:

Maximale Belastung (in Ampere):

[Hinzufügen](#)

## Logout

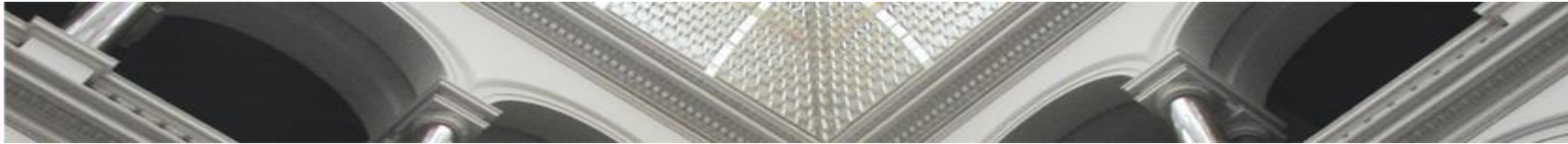


Wenn Sie sich abmelden, können Sie keine neuen Geräte hinzufügen oder Ablesungen vornehmen.

Sie sind eingeloggt als: **Admin**

Zuletzt eingeloggt am: **24.06.2017, 12:23:43**

[Abmelden](#)



## Detailansicht - LoginServlet

Das LoginServlet kümmert sich um den Login von Usern. Der Username wird dem UserDao übergeben, dieser überprüft ob der User bereits in der Usertabelle vorhanden ist. Falls ja wird er als Sessionattribut gesetzt. Falls nicht wird ein neuer User angelegt und in der Usertabelle gespeichert.

```
20 @WebServlet("/login")
21 public class LoginServlet extends HttpServlet {
22     private static final long serialVersionUID = 1L;
23
24     // Injected DAO EJB:
25     @EJB
26     UserDao userDao;
27
28     @Override
29     protected void doGet(
30         HttpServletRequest request, HttpServletResponse response)
31         throws ServletException, IOException {
32         // display "verwalten"
33         request.getRequestDispatcher("/verwalten").forward(request, response);
34     }
35
36     @Override
37     protected void doPost(
38         HttpServletRequest request, HttpServletResponse response)
39         throws ServletException, IOException {
40
41         // get the current session
42         HttpSession session = request.getSession();
43
44         // create or log in an user and set user as session attribute
45         String userName = request.getParameter("user");
46         if (userDao.findUserByName(userName).size() == 1) {
47             User user = userDao.findUserByName(userName).get(0);
48             session.setAttribute("user", user);
49         } else {
50             User user = new User(request.getParameter("user"));
51             userDao.persist(user);
52             session.setAttribute("user", user);
53         }
54
55         // display "verwalten"
56         doGet(request, response);
57     }
}
```

## Alle registrierten Geräte

Gerätetyp	Geräteerkennung	Optionen
	AB12345678	<a href="#">Zur Detailsansicht</a>
	KA93726312	<a href="#">Zur Detailsansicht</a>

## Login

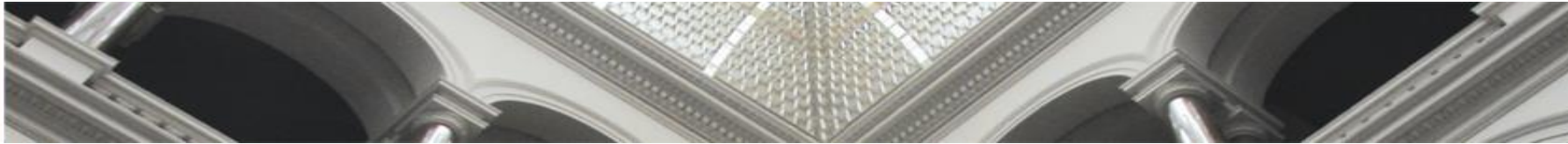


Um neue Geräte hinzufügen und Ablesungen vornehmen zu können, müssen Sie sich anmelden.

**Nutzerkennung:**

[Anmelden](#)





## Detailansicht - LogoutServlet

Das LogoutServlet übernimmt den Logout von Usern, der Zeitpunkt des Logouts wird als lastLogin vom UserDao gespeichert.

```
19 @WebServlet("/logout")
20 public class LogoutServlet extends HttpServlet {
21     private static final long serialVersionUID = 1L;
22
23     // Injected DAO EJB:
24     @EJB
25     UserDao userDao;
26
27     @Override
28     protected void doGet(
29         HttpServletRequest request, HttpServletResponse response)
30         throws ServletException, IOException {
31         // display "verwalten"
32         request.getRequestDispatcher("/verwalten").forward(request, response);
33     }
34
35     @Override
36     protected void doPost(
37         HttpServletRequest request, HttpServletResponse response)
38         throws ServletException, IOException {
39
40         // get the current session
41         HttpSession session = request.getSession();
42
43         // logout an user if logout is requested and save last logged in date
44         session.invalidate();
45         Long userId = Long.parseLong(request.getParameter("logout"));
46         userDao.lastLogin(userId);
47
48         // display "verwalten"
49         doGet(request, response);
50     }
```

# Verwaltung: Smart Meter

Eine einfache Web Applikation zum verwalten und überwachen von Smart Meters

[Zurück zur Übersicht](#)

**Geräteerkennung: AB12345678**

Maximale Belastung: 60.0 Ampere

Spannung: 234.3 Volt

Stromstärke: 62.4 Ampere

**WARNING:**

Die Stromstärke liegt oberhalb der zulässigen Maximalbelastung!

## Alle Ablesungen

Nutzerkennung	Verbrauchswert (in kWh)	Uhrzeit, Datum
Admin	1500.0	12:26:36, 24.06.2017

## Detailansicht - DetailServlet

Die doGet() Methode des DetailServlets ist zuständig für die Darstellung eines einzelnen Smartmeters. Zunächst wird dem SmartmeterDao angeforderte ID übergeben, dieser gibt dann das gesuchte Smartmeter aus der Smartmetertabelle zurück. Da sich die Zufallswerte für Strom und Spannung des Smartmeters bei jedem Aufruf ändern sollen, werden als Nächstes die entsprechenden Berechnungsfunktionen des Smartmeters aufgerufen. Danach wird das Smartmeter als Sessionattribut gesetzt.

Bevor der User Ablesungen für das Smartmeter vornehmen kann, wird überprüft ob er angemeldet ist. Ist dies der Fall, wird die SmartmeterId als Hilfsvariable gesetzt. Nun kann es losgehen.

```

23 @WebServlet("/detail")
24 public class DetailServlet extends HttpServlet {
25     private static final long serialVersionUID = 1L;
26
27     // Injected DAO EJB:
28     @EJB
29     SmartMeterDao smartmeterDao;
30     @EJB
31     RecordDao recordDao;
32     @EJB
33     UserDao userDao;
34
35     /**
36      * display a smart meter with all its properties
37      */
38     @Override
39     protected void doGet(HttpServletRequest request, HttpServletResponse response)
40         throws ServletException, IOException {
41
42         // get the correct smart meter
43         Long smartmeterId = Long.parseLong(request.getParameter("id"));
44         SmartMeter smartmeter = smartmeterDao.findSmartmeterById(smartmeterId);
45         // generate random values for "Spannung" & "Strom"
46         smartmeter.randomSpannung();
47         smartmeter.randomStrom();
48         // set correct smart meter object as the attribute
49         request.setAttribute("smartmeter", smartmeter);
50
51         // check whether user is logged in, if true set user as attribute
52         HttpSession session = request.getSession(false);
53         User user = (User) session.getAttribute("user");
54         if (user != null) {
55             user.setSmartmeterId(smartmeterId);
56             request.setAttribute("user", user);
57         }
58
59         // display smart meter with all its properties
60         request.getRequestDispatcher("/detail.jsp").forward(request, response);
61     }
62 }

```



In der doPost() Methode wird die Erstellung von neuen Ablesungen veranlasst. Der eingegebene Verbrauchswert, das zugehörige Smartmeter und der ablesende User werden dem Konstruktor der Klasse Record übergeben und es wird ein neues Record Objekt erstellt.

Dieses wird vom RecordDao gespeichert und der Liste von Records des Smartmeters hinzugefügt.

```
66 @Override
67 protected void doPost(
68     HttpServletRequest request, HttpServletResponse response)
69     throws ServletException, IOException {
70
71     // create new "Verbrauchswert" record
72     Long smartmeterId = Long.parseLong(request.getParameter("id"));
73     SmartMeter smartmeter = smartmeterDao.findSmartmeterById(smartmeterId);
74     String user = request.getParameter("user");
75     double verbrauchswert = Double.parseDouble(request.getParameter("verbrauchswert"));
76     Record record = new Record(smartmeter, user, verbrauchswert);
77     recordDao.persist(record);
78     smartmeter.getSmartmeterRecords().add(record);
79
80     // display smart meter with all its properties
81     doGet(request, response);
82 }
```



Gerätekennung: AB12345678

Maximale Belastung: 60.0 Ampere  
Spannung: 234.3 Volt  
Stromstärke: 42.1 Ampere

Aktueller Status:

Alles läuft nach Plan: Keine weiteren Informationen vorhanden.

Ablesen

Nutzerkennung:  
Sie sind eingeloggt als: Admin

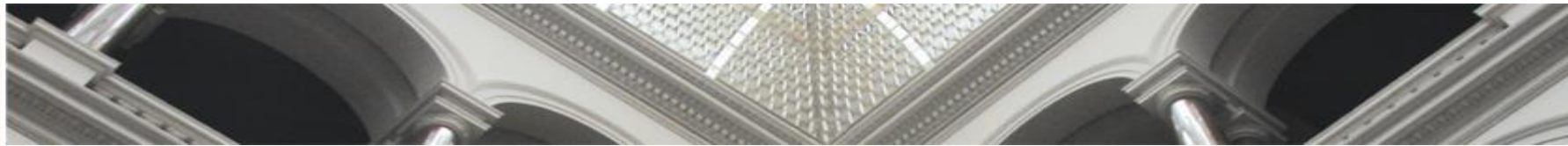
Verbrauchswert:

Ablesen

Alle Ablesungen

Nutzerkennung	Verbrauchswert (in kWh)	Uhrzeit, Datum
Admin	1500.0	12:26:36, 24.06.2017





## JSPs – index.jsp

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3
4 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
5 <html lang="de">
6
7 <head>
8   <title>AnwSys DemoApp</title>
9   <link rel="stylesheet" type="text/css" href="./style.css">
10 </head>
11
12 <body>
13
14   <!-- Navigation Menu -->
15   <div class="menuBg">
16     <div class="menu">
17       <div class="LogoFont">Group 46</div>
18       <ul>
19         <li><a href="./verwalten">Verwaltung</a></li>
20         <li><a class="activ" href="#">Startseite</a></li>
21       </ul>
22     </div>
23   </div>
24
25   <!-- Header -->
26   <div class="heading">
27     <p style="font-size: 44px">Anwendungssysteme Übung 1 SS 2017</p>
28     <div class="subHeading">
29       <div class="subHLeft">Eine einfache Web Applikation zum
30         verwalten und überwachen von Smart Meters</div>
31       <div class="subHRight">
32         <a class="button" href="./verwalten">Zur Verwaltung</a>
33       </div>
34     </div>
35   </div>
36
37   <!-- Content Box -->
38   <div class="main">
39     <div class="content">
40       <center>
41         <h1>POWERFUL TOOLS TO HELP YOU MANAGE YOUR INTERESTS</h1>
42       </center>
43       <table class="Landing">
```

```
44
45       <td></img></td>
46       <td class="Landing">Ihre Webapplikation soll sowohl über eine
47         Gesamtübersicht aller eingetragenen Smart Meter, als auch
48         individuelle Detailsichten für jeden Smart Meter verfügen.
49         Zwischen den individuellen Ansichten und der Übersichtseite muss
50         navigiert werden können. Legen Sie mindestens zwei Detailsichten
51         an. <br /> Auf der individuellen Detailsicht eines Smart Meters
52         sollen das Produktbild des Geräts und die statischen Eigenschaften
53         dargestellt sein. Beim Aufruf einer Detailsicht soll außerdem
54         der aktuelle Status des Geräts angezeigt werden, d.h. anliegende
55         Spannung und Stromstärke. Diese Werte sollen zufallsgeneriert
56         (gleichverteilt) sein und bei jedem Aufruf neu berechnet werden.
57         Die Spannung (Genauigkeit: 0.1 Volt) soll zwischen 220 und 240
58         Volt liegen, die Stromstärke (Genauigkeit: 0.1 Ampere) zwischen 0
59         und (der jeweiligen maximalen Belastung +5 Ampere). Sofern die
60         Stromstärke oberhalb der zulässigen Maximalbelastung liegt, soll
61         auf der Smart Meter-Detailseite eine Warnung angezeigt werden.
62       </td>
63     </tr>
64     <tr>
65       <td></img></td>
66       <td class="Landing">Beim Ablesen trägt der Nutzer Ihrer
67         Webapplikation dabei seine Nutzerkennung und einen Verbrauchswert
68         (in kWh) in ein Formular ein. Die Nutzerkennung besteht aus den
69         Initialen des Namens des Nutzers und den letzten beiden Ziffern
70         seiner Matrikelnummer. Ablesungen sollen nach dem Abschicken durch
71         den Nutzer automatisch mit dem Datum des Ablesezeitpunkts versehen
72         und auf der Detailsicht des jeweiligen Smart Meters in eine
73         Liste oder Tabelle aufgenommen werden. <br /> Sie müssen keine
74         Zugriffskontrolle oder ähnliches implementieren. Auch persistente
75         Datenhaltung muss nicht implementiert werden, d.h. wenn Sie eine
76         Detailsicht im Browser neu laden kann die Liste der Ablesungen
77         wieder leer sein.
78       </td>
79     </tr>
```



```

80      </table>
81      <br />
82      <h2>Vorgehensweise</h2>
83      <hr />
84      <p>
85          <b>Schritt 1</b><br /> Stellen Sie konzeptionell dar, welche Seiten
86          ihre Webapplikation umfassen wird, welche Funktionen jeweils auf
87          diesen zur Verfügung stehen und welche Informationen und
88          Interaktionselemente darauf vorhanden sein werden. Erläutern Sie
89          zudem an welcher Stelle sie welche Technologie einsetzen werden.
90          Überlegen Sie sich außerdem ein einfaches Design für Ihre
91          Applikation. <br /> <br /> <b>Schritt 2</b><br /> Beschreiben Sie
92          Ihre Seiten als HTML-Dokumente. Wählen Sie geeignete Elemente um
93          Seitenbestandteile sinnvoll auszuzeichnen. Begründen Sie kurz, warum
94          Sie sich für die gewählten Elemente entschieden haben. <br /> <br />
95          <b>Schritt 3</b><br /> Kompletтировать Sie das in Schritt 1
96          entworfene Design mit Hilfe von CSS. Dokumentieren Sie ihr Vorgehen.
97          <br /> <br /> <b>Schritt 4</b><br /> Implementieren Sie
98          dynamische Bestandteile und Funktionen mit Hilfe von JavaScript.
99          Gerne dürfen Sie jQuery verwenden. Erläutern Sie die so realisierten
100          Features und ihren Ansatz für die Umsetzung. <br /> <br /> <b>Schritt
101          5</b><br /> Dokumentieren Sie die Ergebnisse Ihrer Arbeit in einer
102          Präsentation (Powerpoint o.ä.). Diese Präsentation muss für das
103          Gesamtverständnis Ihrer Lösung genügen, daher sollten relevante
104          Codeauszüge enthalten sein. In der Präsentation müssen außerdem
105          mindestens vier (4) unterschiedliche Nutzerkennungen (vgl.
106          Aufgabenstellung) zu sehen sein, die Ihre tatsächlichen
107          Namen/Matrikelnummern der Gruppe enthalten. <br /> <br /> <b>Schritt
108          6</b><br /> Exportieren Sie die Präsentation als PDF-Dokument. Packen
109          Sie ihren kompletten Code und alle zugehörige Artefakte (z.B.
110          Bilder) zudem als .zip oder .7z Datei. Reichen Sie die Präsentation
111          und die .zip-Datei mit Hilfe des ISIS-Systems ein. Bewertet wird nur
112          diese Präsentation! Der Quellcode dient dabei als Referenz und
113          Nachweis der Leistung. Es genügt, wenn die Einreichung durch ein
114          Gruppenmitglied erfolgt.<br /> Deadline für die Abgabe per Upload:
115          <b>28.05.2017, 23:55</b>.
116      </p>
117  </div>
118 </div>

```

```

119
120 <!-- Footer -->
121 <div class="footerBg">
122     <div class="footer">
123         <div class="copyright">
124             © Copyright 2017 by <b class="footerFont">Group 46</b>
125         </div>
126         <div class="toTop">
127             <a class="ttop" href="#">Go to top</a>
128         </div>
129     </div>
130 </div>
131
132 </body>
133
134 </html>
135

```

## JSPs – user.jsp

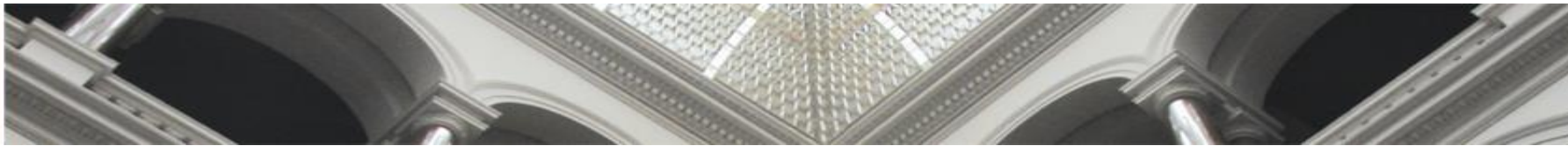
```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3
4 <%@page import="java.util.*,de.tub.as.smm.models.User"%>
5
6 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
7 <html>
8 <head>
9 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
10 <title>User Management</title>
11 </head>
12
13 <body>
14   <form method="POST" action="user">
15     Name: <input type="text" name="name" /> <input type="submit"
16       value="Add" />
17   </form>
18
19   <hr>
20   <ol>
21     <%
22       @SuppressWarnings("unchecked")
23       List<User> users = (List<User>) request.getAttribute("user");
24       if (users != null) {
25         for (User user : users) {
26           %>
27           <li><%=user%></li>
28         <%
29         }
30       }
31     %>
32   </ol>
33   <hr>
34
35 </body>
36 </html>
```



## JSPs – verwalten.jsp

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3
4 <%@page import="java.util.*,de.tub.as.smm.models.*"%>
5
6 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
7 <html lang="de">
8
9 <head>
10 <title>AnwSys DemoApp</title>
11 <link rel="stylesheet" type="text/css" href="/style.css">
12 </head>
13
14 <body>
15
16 <!-- Navigation Menu -->
17 <div class="menuBg">
18 <div class="menu">
19 <div class="LogoFont">Group 46</div>
20 <ul>
21 <li><a class="activ" href="#">Verwaltung</a></li>
22 <li><a href="#">Startseite</a></li>
23 </ul>
24 </div>
25 </div>
26
27 <!-- Header -->
28 <div class="heading">
29 <p style="font-size: 44px">Verwaltung: Übersicht</p>
30 <div class="subHeading">
31 <div class="subHLeft">Eine einfache Web Applikation zum
32   verwalten und überwachen von Smart Meters</div>
33 <div class="subHRight">
34 <a class="button" href="#">Zurück zur Startseite</a>
35 </div>
36 </div>
37 </div>
38
39 <!-- Content Box -->
40 <div class="main">
41 <div class="content">
42 <h2>Alle registrierten Geräte</h2>
43 <hr />
```

```
44 <table class="overview">
45 <tr>
46 <th>Gerätetyp</th>
47 <th>Geräteerkennung</th>
48 <th>Optionen</th>
49 </tr>
50 <%
51 @SuppressWarnings("unchecked")
52 List<SmartMeter> smartmeters = (List<SmartMeter>) request.getAttribute("smartmeter");
53 if (smartmeters != null) {
54   for (SmartMeter smartmeter : smartmeters) {
55
56 <tr class="overview">
57 <td></img></td>
58 <td><%=smartmeter.getGeraeterkennung()%></td>
59 <td>
60 <form method="GET" action="detail">
61 <input type="hidden" name="id" value="<%=smartmeter.getId()%>">
62 <a href="/detail"><input class="inButton" type="submit"
63   value="Zur Detailansicht" /></a>
64 </form>
65 </td>
66 </tr>
67 <%
68   }
69 }
70 <%>
71 </table>
72 <%
73 if (request.getAttribute("user") == null) {
74 <%=User.loginForm()%>
75 <%
76   }
77 <%
78 if (request.getAttribute("user") != null) {
79   User user = (User) request.getAttribute("user");
80 <%=user.addDevices()%>
81 <%=user.logout()%>
82 <%>
83 <%>
84 <%>
```



```
85      <%  
86      }  
87      %>  
88      </div>  
89      </div>  
90  
91  
92      <!-- Footer -->  
93      <div class="footerBg">  
94          <div class="footer">  
95              <div class="copyright">  
96                  © Copyright 2017 by <b class="footerFont">Group 46</b>  
97              </div>  
98              <div class="toTop">  
99                  <a class="ttop" href="#">Go to top</a>  
100              </div>  
101          </div>  
102      </div>  
103  
104  
105 </body>  
106  
107 </html>
```



## JSPs – detail.jsp

```

1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@page import="java.util.*,de.tub.as.smm.models.*"%>
4
5 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
6 <html lang="de">
7
8 <head>
9   <title>AnwSys DemoApp</title>
10  <link rel="stylesheet" type="text/css" href="./style.css">
11 </head>
12
13 <body>
14
15   <!-- Navigation Menu -->
16   <div class="menuBg">
17     <div class="menu">
18       <div class="LogoFont">Group 46</div>
19       <ul>
20         <li><a class="activ" href="./verwalten">Verwaltung</a></li>
21         <li><a href="./">Startseite</a></li>
22       </ul>
23     </div>
24   </div>
25
26   <!-- Header -->
27   <div class="heading">
28     <p style="font-size: 44px">Verwaltung: Smart Meter</p>
29     <div class="subHeading">
30       <div class="subHLeft">Eine einfache Web Applikation zum
31         verwalten und überwachen von Smart Meters</div>
32       <div class="subHRight">
33         <a class="button" href="./verwalten">Zurück zur Übersicht</a>
34       </div>
35     </div>
36   </div>
37   <%
38     SmartMeter smartmeter = (SmartMeter) request.getAttribute("smartmeter");
39     %>

```

```

40   <!-- Content Box -->
41   <div class="main">
42     <div class="content">
43       <div class="detail">
44         <tr class="detail">
45           <td></td>
46           <td>
47             <table class="status">
48               <tr>
49                 <td><h2>Geräteerkennung: <%=smartmeter.getGeraetekennung()%></h2>
50                 <td><h2>Maximale Belastung:</h2>
51                 <td><%=smartmeter.getMaxBelastung()%> Ampere</td>
52               </tr>
53               <tr>
54                 <td><h2>Spannung:</h2>
55                 <td><%=smartmeter.getSpannung()%> Volt</td>
56               </tr>
57               <tr>
58                 <td><h2>Stromstärke:</h2>
59                 <td><%=smartmeter.getStrom()%> Ampere</td>
60               </tr>
61             </table>
62             <%=smartmeter.currentStatus()%>
63           </td>
64         </tr>
65       </div>
66       <%
67         if (request.getAttribute("user") != null) {
68           User user = (User) request.getAttribute("user");
69           %>
70           <%=user.ablesenForm()%>
71           <%
72         }
73         %>
74       <br />
75       <h2>Alle Ablesungen</h2>
76       <hr />
77       <table class="event">
78         <tr>
79           <th class="event">Nutzererkennung</th>
80           <th class="event">Verbrauchswert (in kWh)</th>
81           <th class="event">Uhrzeit, Datum</th>
82         </tr>

```



```

83      <%
84          List<Record> allRecords = smartmeter.getSmartmeterRecords();
85          ListIterator<Record> iter = allRecords.listIterator(allRecords.size());
86          while (iter.hasPrevious()) {
87              Record record = iter.previous();
88          }
89      <tr>
90          <td><%=record.getUser()%></td>
91          <td><%=record.getRecord()%></td>
92          <td><%=record.getFormattedDate()%></td>
93      </tr>
94      <%
95          }
96      <%
97  </table>
98  </div>
99  </div>
100
101
102  <!-- Footer -->
103  <div class="footerBg">
104      <div class="footer">
105          <div class="copyright">
106              © Copyright 2017 by <b class="footerFont">Group 46</b>
107          </div>
108          <div class="toTop">
109              <a class="ttop" href="#">Go to top</a>
110          </div>
111      </div>
112  </div>
113
114 </body>
115
116 </html>

```