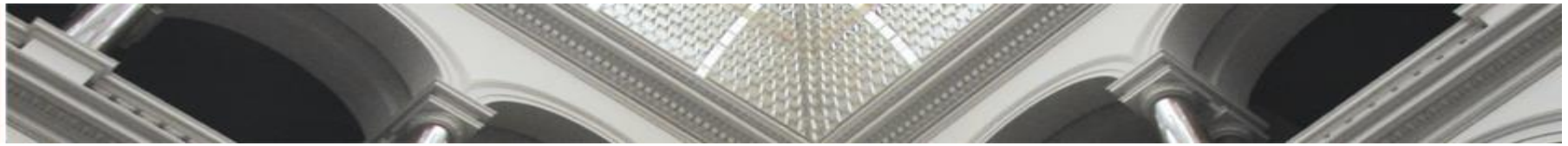
Technische Universität Berlin

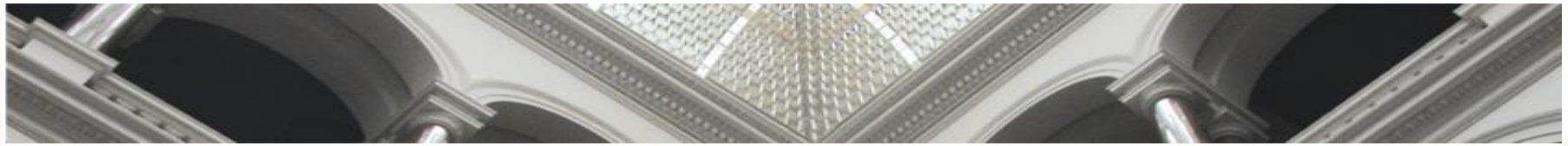
Java EE – Übung 2

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



Technische Universität Berlin

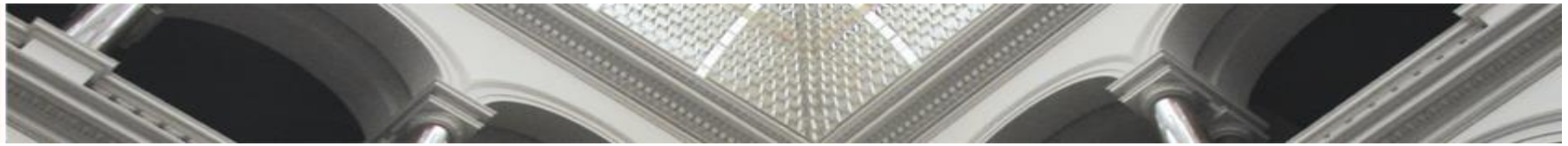
Content

Struktur des Projekts

- -EJB Models
- -Daos
- -Servlets

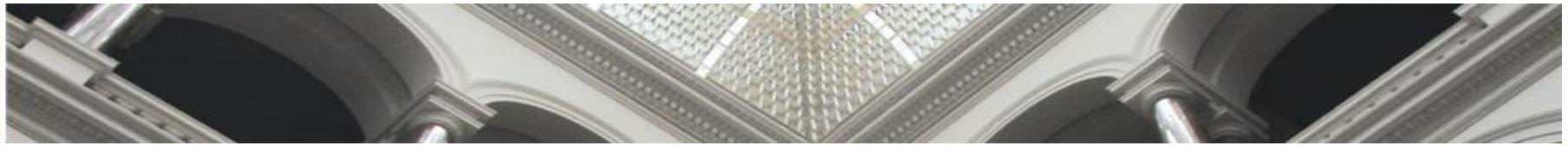
Detailansichten der einzelnen Klassen mit Code

- -EJB Models
- -Daos
- -Servlets



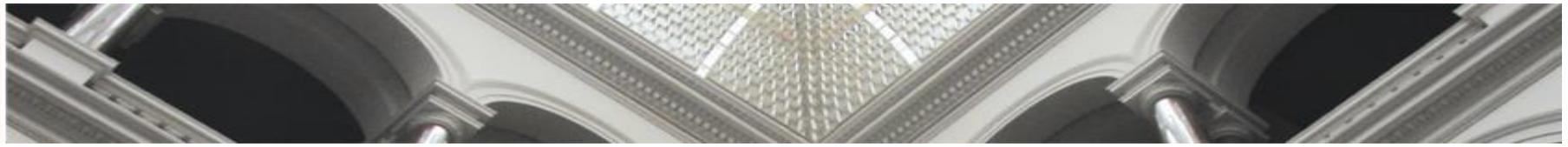
Struktur des Projekts – EJB Models

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.



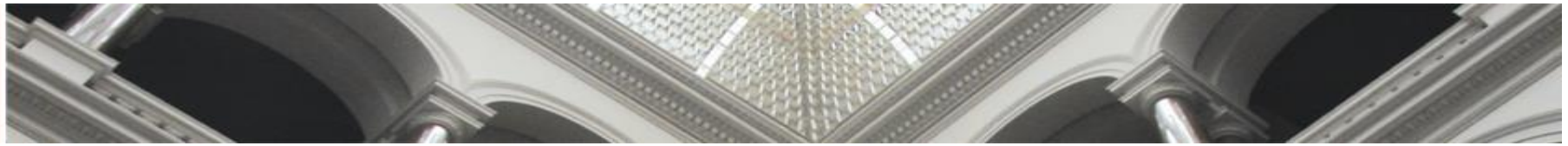
Struktur des Projekts - Servlets

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.

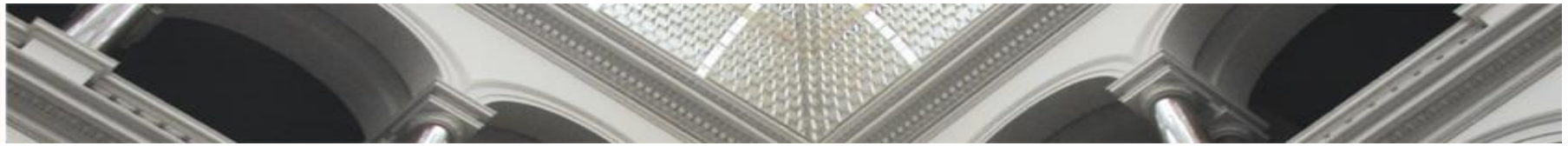


Technische Universität Berlin

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     }
```

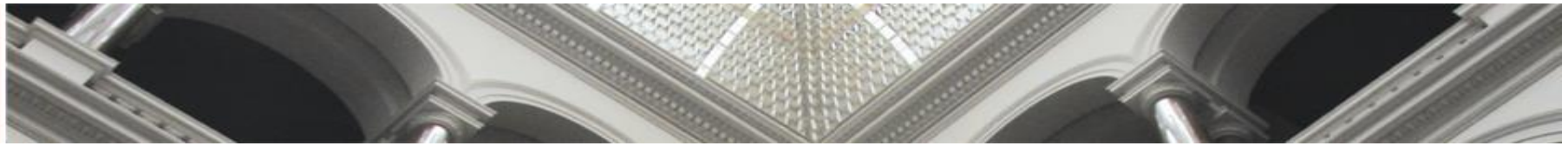


Technische Universität Berlin

Zur Anmeldung der User wird eine Loginform verwendet.

Es gibt eine addDevicesform, um neue Smartmeters einzutragen.

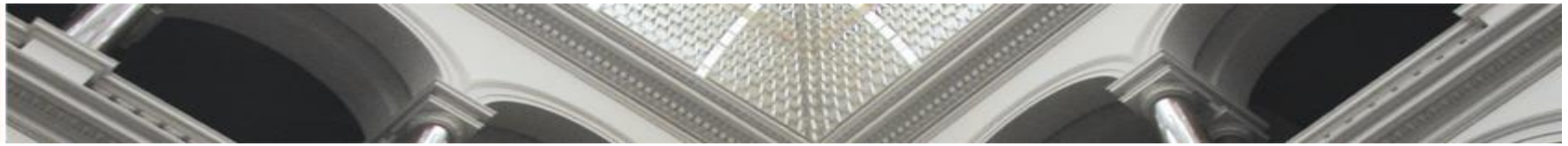
```
77      // forms
78      public static String loginForm() {
79          return "<br />" +
80                  "<h2>Login</h2>" +
81                  "<hr />" +
82                  "<table>" +
83                  "<tr>" +
84                  "<td><img src=\"./media/unlocked.png\" height=\"200\" width=\"200\"></img></td>" +
85                  "<td>Um neue GerÃ¤te hinzufÃ¼gen und Ablesungen vornehmen zu kÃ¶nnen, mÃ¼ssen 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>GerÃ¤te hinzufÃ¼gen</h2>" +
103                 "<hr />" +
104                 "<form method=\"POST\" action=\"verwalten\">" +
105                 "<b>GerÃ¤tekennung:</b>" +
106                 "<input pattern=\"[A-Z]{2}[0-9]{8}\" required title=\"Zwei GroÃŸbuchstaben 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=\"HinzufÃ¼gen\" />" +
111                 "</form>";
112     }
```



Technische Universität Berlin

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

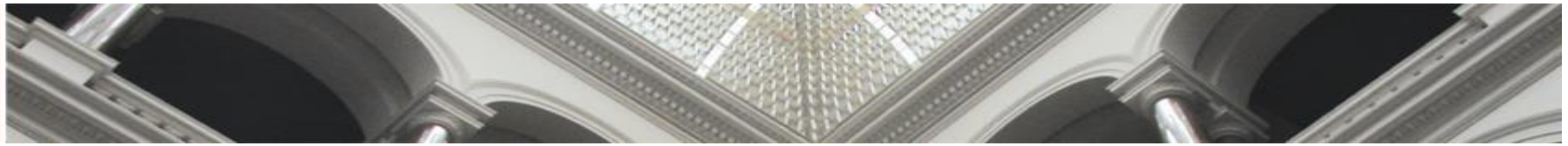
```
114    @Transient
115    public String logout() {
116        return "<br />" +
117                "<h2>Logout</h2>" +
118                "<hr />" +
119                "<table>" +
120                "<tr>" +
121                "<td><img src=\"./media/locked.png\" height=\"200\" width=\"200\"></img></td>" +
122                "<td>Wenn Sie sich abmelden, kÃ¶nnen Sie keine neuen GerÃ¤te hinzufÃ¼gen 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, geraetekennung, maxBelastung und eine Liste von smartmeterRecords. Dazu kommen noch die nicht peristenten Attribute spannung und strom, die bei jedem Aufruf neu berechnet werden.

```java
package de.tub.as.smm.models;

import java.io.Serializable;

@Entity
public class SmartMeter implements Serializable {

    private static final long serialVersionUID = -2250177305743873329L;
    // persistent fields
    private Long id;
    private String geraetekennung;
    private double maxBelastung;
    private List<Record> smartmeterRecords;
    // non persistent fields
    private double spannung;
    private double strom;

    // constructors
    public SmartMeter() {
    }

    public SmartMeter(String geraetekennung, double maxBelastung) {
        this.geraetekennung = geraetekennung;
        this.maxBelastung = maxBelastung;
        this.smartmeterRecords = new ArrayList<Record>();
    }

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    public Long getId() {
        return id;
    }
```



Technische Universität Berlin

Die Smartmeter stehen in einer OneToMany Beziehung zu den Records.

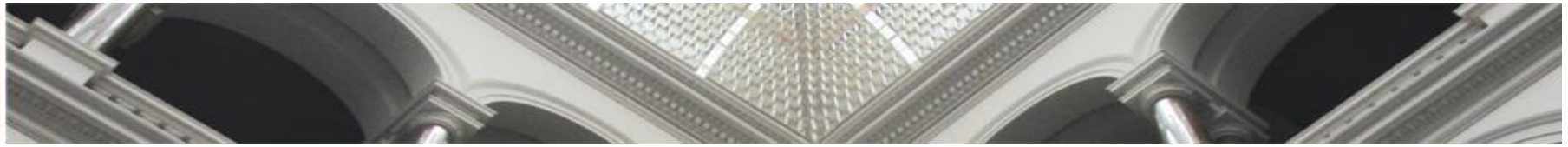
```
@OneToMany(cascade = { CascadeType.PERSIST,
        CascadeType.MERGE }, targetEntity = Record.class, mappedBy = "smartmeter", fetch = FetchType.EAGER)
public List<Record> getSmartmeterRecords() {
    return smartmeterRecords;
}

public void setSmartmeterRecords(List<Record> smartmeterRecords) {
    this.smartmeterRecords = smartmeterRecords;
}

@Transient
public double getSpannung() {
    return spannung;
}
```

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

```
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;
}
```

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

```
@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>";
}
```

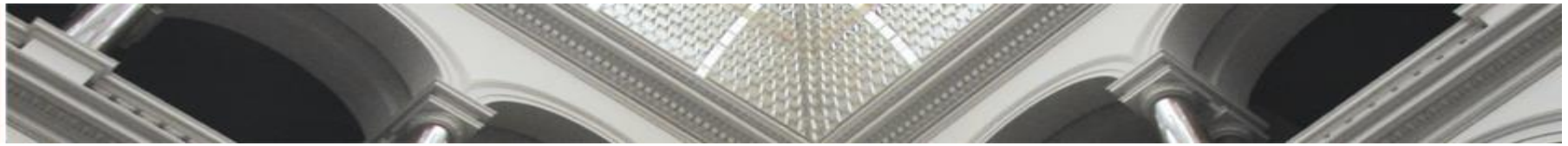


Detailansicht - Records

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     }
```

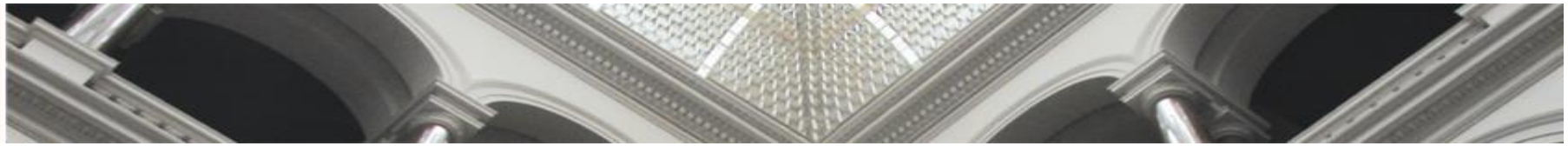


Technische Universität Berlin

Detailansicht - UserDao

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

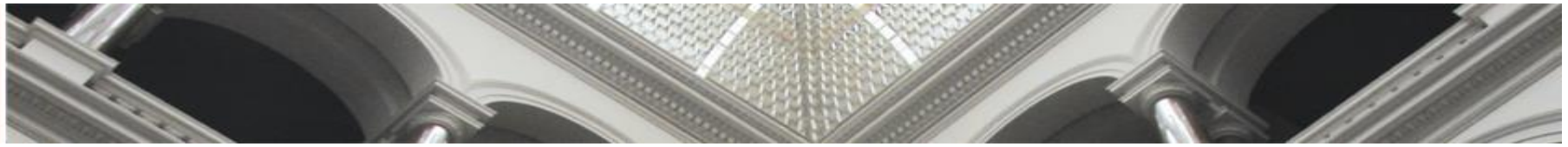
```java
package de.tub.as.smm.dao;

import java.util.Date;

/**
 * Session Bean implementation class UserDaoEJB
 */
@Stateless
public class UserDao {

    // injected database connection
    @PersistenceContext
    private EntityManager em;

    // store a new user
    public void persist(User user) {
        em.persist(user);
    }

    // retrieve all the users
    public List<User> getAllUsers() {
        TypedQuery<User> query = em.createQuery("SELECT u FROM User u ORDER BY u.id", User.class);
        return query.getResultList();
    }

    // find a user
    public List<User> findUserByName(String name) {
        TypedQuery<User> query = em.createQuery("SELECT u FROM User u WHERE u.name LIKE :userName", User.class)
                .setMaxResults(1).setParameter("userName", name);
        return query.getResultList();
    }

    // update last login date
    public void lastLogin(Long userId) {
        em.createQuery("UPDATE User SET lastLogin = :date WHERE id LIKE :userId")
                .setParameter("userId", userId).setParameter("date", new Date(System.currentTimeMillis()))
                .executeUpdate();
    }
```



Detailansicht - SmartMeterDao

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

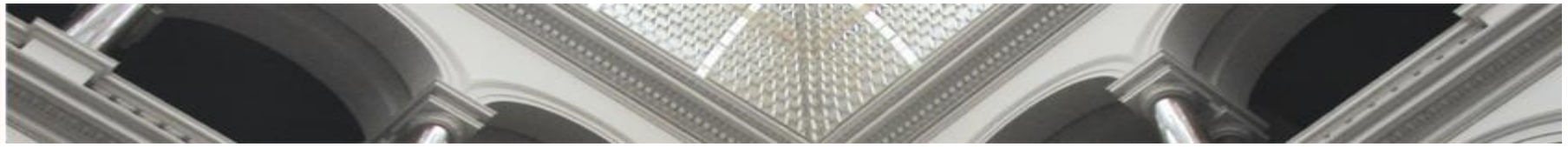
```java
package de.tub.as.smm.dao;

import java.util.List;

/**
 * Session Bean implementation class SmartMeterDaoEJB
 */
@Stateless
public class SmartMeterDao {

    // injected database connection
    @PersistenceContext
    private EntityManager em;

    // store a new smart meter
    public void persist(SmartMeter smartMeter) {
        em.persist(smartMeter);
    }

    // retrieve all smart meters
    public List<SmartMeter> getAllSmartMeters() {
        TypedQuery<SmartMeter> query = em.createQuery("SELECT u FROM SmartMeter u ORDER BY u.id", SmartMeter.class);
        return query.getResultList();
    }

    public SmartMeter findSmartmeterById(Long smartmeterId) {
        TypedQuery<SmartMeter> query = em
                .createQuery("SELECT u FROM SmartMeter u WHERE u.id LIKE :sId", SmartMeter.class)
                .setParameter("sId", smartmeterId).setMaxResults(1);
        return query.getSingleResult();
    }

}
```



Detailansicht - RecordDao

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

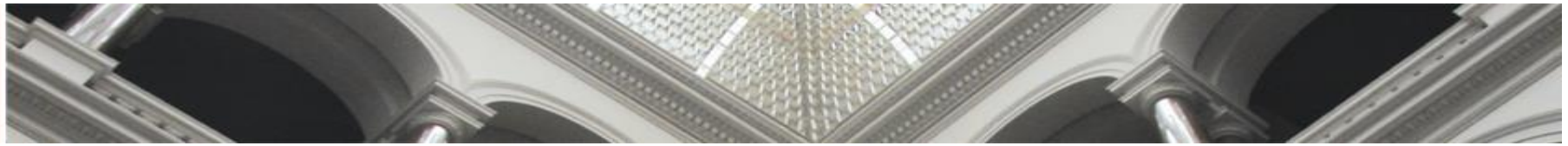
```java
package de.tub.as.smm.dao;

import java.util.List;

/**
 * Session Bean implementation class RecordDao
 */
@Stateless
public class RecordDao {

    // injected database connection
    @PersistenceContext
    private EntityManager em;

    // store a new record
    public void persist(Record record) {
        em.merge(record);
    }

    // retrieve all records
    public List<Record> getAblesungen() {
        TypedQuery<Record> query = em.createQuery("SELECT u FROM Record u ORDER BY u.id DESC", Record.class);
        return query.getResultList();
    }
```



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.

```java
package de.tub.as.smm;

import java.io.IOException;

/**
 * Servlet implementation class SmartMeterServlet
 */
@WebServlet("/verwalten")
public class SmartMeterServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    // Injected DAO EJB:
    @EJB
    SmartMeterDao smartmeterDao;
    @EJB
    UserDao userDao;

    /**
     * display the list of smart meters
     */
    @Override
    protected void doGet(
            HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        // set a single user as the attribute if someone is logged in
        HttpSession session = request.getSession(false);
        if (session != null) {
            User user = (User) session.getAttribute("user");
            request.setAttribute("user", user);
        }

        // set a list of all smart meters as the attribute
        request.setAttribute("smartmeter", smartmeterDao.getAllSmartMeters());

        // display the list of smart meters
        request.getRequestDispatcher("/verwalten.jsp").forward(request, response);
    }
```



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):

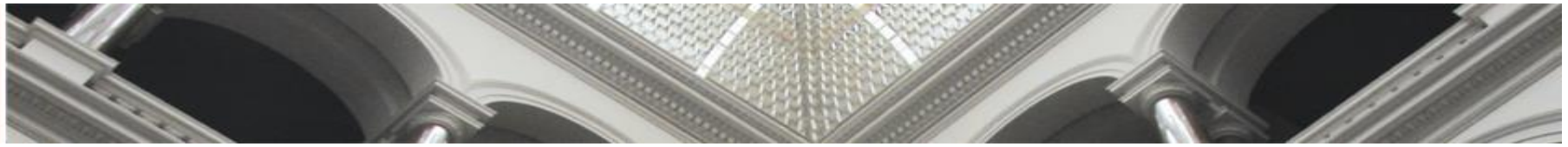
Hinzufügen

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	Gerätekennung	Optionen
	AB12345678	Zur Detailansicht
	KA93726312	Zur Detailansicht

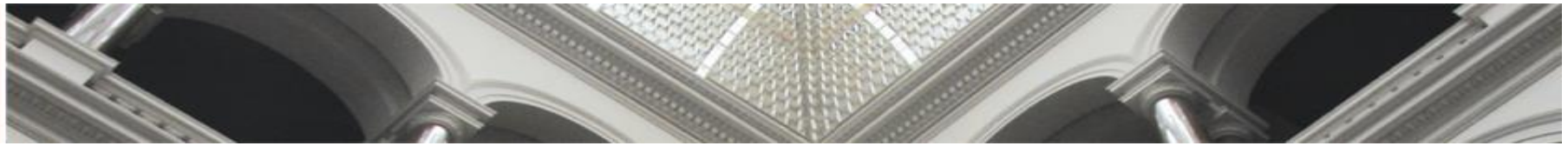
Login

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

Nutzerkennung:

Anmelden

Go to top



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

Gerätekennung: AB12345678

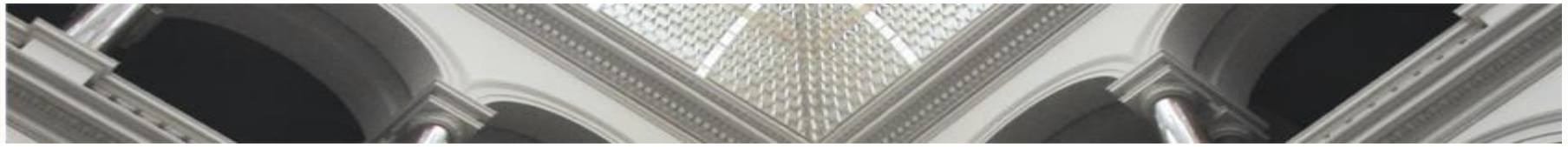
Maximale Belastung:	60.0 Ampere
Spannung:	234.3 Volt
Stromstärke:	62.4 Ampere

WARNUNG:
Die Stromstärke liegt oberhalb der zulässigen Maximalbelastung!

Alle Ablesungen

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

Go to top

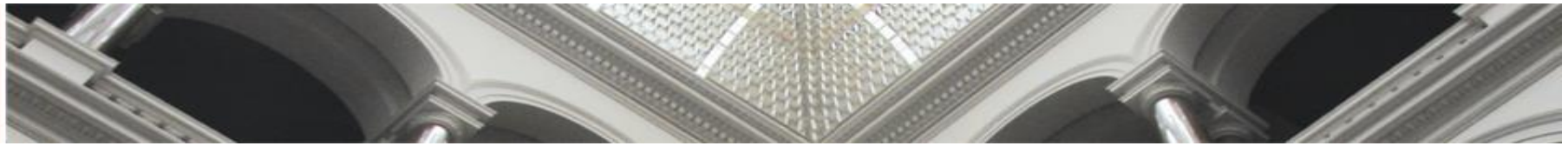


Detailansicht - DetailServlet

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" & "Stom"
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.setSmartmerterId(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     }
```



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

Go to top