Moveatis 1.0.0 Source Code

Moveatis Project

June 16, 2016

Contents

Chapter 1

Java source code

1.1 abstracts/AbstractBean.java

```
1  package com.moveatis.abstracts;
2
3  import com.moveatis.timezone.TimeZoneInformation;
4  import java.util.Calendar;
5  import java.util.List;
6  import javax.persistence.EntityManager;
7  import javax.persistence.Query;
8  import javax.persistence.TypedQuery;
9  import javax.persistence.criteria.CriteriaBuilder;
10 import javax.persistence.criteria.CriteriaQuery;
11 import javax.persistence.criteria.Root;
12
13 /**
14  * The super class to the enterprise beans manages the persistent connection and
        entities.
15  *
16  * @author Sami Kallio <phinaliumz at outlook.com>
17  * @param <T> The entity the child of this bean uses
18  *
19  */
20 public abstract class AbstractBean<T extends BaseEntity> {
21
22    private Class<T> entityClass;
23
24    public AbstractBean(Class<T> entityClass) {
25       this.entityClass = entityClass;
26    }
27
28    protected abstract EntityManager getEntityManager();
29
30    /**
31     * Creates a new entity.
32     * @param entity The entity to be created.
33     */
34    public void create(T entity) {
35       getEntityManager().persist(entity);
36    }
37
38    /**
39     * Edits the entity.
```

```
40      * @param entity The entity to be edited.
41      */
42     public void edit(T entity) {
43        getEntityManager().merge(entity);
44     }
45
46     /**
47      * Removes the entity.
48      * @param entity The entity to be removed.
49      */
50     public void remove(T entity) {
51        entity.setRemoved(); //entity is not actually removed, only the removed-date
              is set
52        getEntityManager().merge(entity);
53     }
54
55     /**
56      * Finds an entity and returns it if it's not out of date.
57      * @param id The id of the entity to be found.
58      * @return The entity, if it is found. Otherwise null.
59      */
60     public T find(Object id) {
61        T entity = (T)getEntityManager().find(entityClass, id);
62
63        if(entity.getRemoved() != null) {
64           Calendar calendar = Calendar.getInstance(TimeZoneInformation.getTimeZone
                 ());
65           Calendar entityCalendar = Calendar.getInstance(TimeZoneInformation.
                 getTimeZone());
66           entityCalendar.setTime(entity.getRemoved());
67
68           if(entityCalendar.before(calendar)) {
69              return entity;
70           } else {
71              return null;
72           }
73        } else {
74           return entity;
75        }
76     }
77
78     /**
79      * Finds all the entities, whose type matches the requested entity.
80      * @return A list of all the entities of the requested entity type.
81      */
82     public List<T> findAll() {
83        CriteriaBuilder cb = getEntityManager().getCriteriaBuilder();
84        CriteriaQuery<T> cq = cb.createQuery(entityClass);
85
86        Root<T> rt = cq.from(entityClass);
87        CriteriaQuery<T> all = cq.select(rt);
88
89        TypedQuery<T> allQuery = getEntityManager().createQuery(all);
90        return allQuery.getResultList();
91     }
92
93     /**
94      * Finds and returns the list of entities in the specified range.
```

```
95      * The range array has two elements: the minimum and the maximum of the range.
96      *
97      * @param range An array with two elements.
98      * @return A list of the entities in the range.
99      */
100    public List<T> findRange(int[] range) {
101       CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();
102       cq.select(cq.from(entityClass));
103       Query q = getEntityManager().createQuery(cq);
104       q.setMaxResults(range[1] - range[0] + 1);
105       q.setFirstResult(range[0]);
106       return q.getResultList();
107    }
108
109    /**
110     * Counts how many entities there are of the requested type.
111     * @return The count of the entities.
112     */
113    public int count() {
114       CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();
115       Root<T> rt = cq.from(entityClass);
116       cq.select(getEntityManager().getCriteriaBuilder().count(rt));
117       Query q = getEntityManager().createQuery(cq);
118       return ((Long) q.getSingleResult()).intValue();
119    }
120
121 }
```

1.2 abstracts/BaseEntity.java

```
1  package com.moveatis.abstracts;
2
3  import com.moveatis.timezone.TimeZoneInformation;
4  import java.util.Calendar;
5  import java.util.Date;
6  import javax.persistence.GeneratedValue;
7  import javax.persistence.GenerationType;
8  import javax.persistence.Id;
9  import javax.persistence.MappedSuperclass;
10 import javax.persistence.PrePersist;
11 import javax.persistence.Temporal;
12 import javax.persistence.TemporalType;
13 import javax.validation.constraints.NotNull;
14
15 /**
16  * The entity is the base of all the entities of the application. It has the
17  * id and the dates for creation and removal. Like the MappedSuperclass, it's
18  * not persisted to the database itself.
19  * @author Sami Kallio <phinaliumz at outlook.com>
20  */
21
22 @MappedSuperclass
23 public abstract class BaseEntity {
24
25    @Id
```

```
26      @GeneratedValue(strategy = GenerationType.AUTO)
27      protected Long id;
28
29      @Temporal(TemporalType.TIMESTAMP)
30      @NotNull
31      private Date created;
32
33      @Temporal(TemporalType.TIMESTAMP)
34      private Date removed;
35
36      @PrePersist
37      public void setCreated() {
38         Calendar calendar = Calendar.getInstance(TimeZoneInformation.getTimeZone());
39         created = calendar.getTime();
40      }
41
42      public Date getCreated() {
43         return this.created;
44      }
45
46      protected void setRemoved() {
47         Calendar calendar = Calendar.getInstance(TimeZoneInformation.getTimeZone());
48         removed = calendar.getTime();
49      }
50
51      protected Date getRemoved() {
52         return this.removed;
53      }
54
55      protected Long getId() {
56         return id;
57      }
58
59      protected void setId(Long id) {
60         this.id = id;
61      }
62
63
64
65  }
```

1.3 application/ApplicationBean.java

```
1   package com.moveatis.application;
2
3   import com.moveatis.abstracts.AbstractBean;
4   import javax.persistence.EntityManager;
5   import javax.persistence.PersistenceContext;
6   import javax.ejb.Singleton;
7   import javax.ejb.Startup;
8   import com.moveatis.interfaces.Application;
9   import org.slf4j.Logger;
10  import org.slf4j.LoggerFactory;
11
12  /**
```

```
 * The Application enterprise bean controls access to the application entity.
 * It is a singleton bean, so only one instance is running at any time, and it is
 * automatically started when the application is started.
 *
 * @author Sami Kallio <phinaliumz at outlook.com>
 */
@Singleton
@Startup
public class ApplicationBean extends AbstractBean<ApplicationEntity> implements
    Application {

   private static final Logger LOGGER = LoggerFactory.getLogger(ApplicationBean.
       class);

   @PersistenceContext(unitName = "MOVEATIS_PERSISTENCE")
   private EntityManager em;

   public ApplicationBean() {
      super(ApplicationEntity.class);
   }

   @Override
   protected EntityManager getEntityManager() {
      return em;
   }

   /**
    * Checks if the application has been installed.
    */
   @Override
   public boolean checkInstalled() {
      return super.findAll().size() == 1;
   }

   /**
    * Gets the singleton application entity.
    */
   @Override
   public ApplicationEntity getApplicationEntity() {
      return super.findAll().get(0);
   }
}
```

1.4 application/ApplicationEntity.java

```
package com.moveatis.application;

import com.moveatis.abstracts.BaseEntity;
import com.moveatis.roles.SuperUserRoleEntity;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import javax.persistence.Entity;
import javax.persistence.OneToMany;
```

```
11 import javax.persistence.Table;
12 import javax.persistence.Temporal;
13 import javax.persistence.TemporalType;
14 import javax.validation.constraints.NotNull;
15
16 /**
17  * The entity represents the data of the application in the database.
18  *
19  * @author Sami Kallio <phinaliumz at outlook.com>
20  */
21 @Table(name="APPLICATION")
22 @Entity
23 public class ApplicationEntity extends BaseEntity implements Serializable {
24
25    private static final long serialVersionUID = 1L;
26
27    @Temporal(TemporalType.DATE)
28    @NotNull
29    private Date applicationInstalled;
30
31    @OneToMany
32    private List<SuperUserRoleEntity> superUsers;
33
34    private String reportEmail;
35
36    /**
37     * Gets the list of the users with the superuser role.
38     */
39    public List<SuperUserRoleEntity> getSuperUsers() {
40       if(superUsers == null) {
41          superUsers = new ArrayList<>();
42       }
43       return superUsers;
44    }
45
46    public void setSuperUsers(List<SuperUserRoleEntity> superUsers) {
47       this.superUsers = superUsers;
48    }
49
50    /**
51     * Returns the date when the application was installed.
52     */
53    public Date getApplicationInstalled() {
54       return applicationInstalled;
55    }
56
57    /**
58     * Sets the date when the application was installed.
59     */
60    public void setApplicationInstalled(Date applicationInstalled) {
61       this.applicationInstalled = applicationInstalled;
62    }
63
64    public String getReportEmail() {
65       return reportEmail;
66    }
67
68    public void setReportEmail(String reportEmail) {
```

```
69          this.reportEmail = reportEmail;
70      }
71
72      @Override
73      public int hashCode() {
74          int hash = 0;
75          hash += (id != null ? id.hashCode() : 0);
76          return hash;
77      }
78
79      @Override
80      public boolean equals(Object object) {
81          if (!(object instanceof ApplicationEntity)) {
82              return false;
83          }
84          ApplicationEntity other = (ApplicationEntity) object;
85          return !((this.id == null && other.id != null) || (this.id != null && !this.
                id.equals(other.id)));
86      }
87
88      @Override
89      public String toString() {
90          return "com.moveatis.application.ApplicationEntity[ id=" + id + " ]";
91      }
92
93  }
```

1.5 application/InstallationBean.java

```
1
2   package com.moveatis.application;
3
4   import com.moveatis.enums.ApplicationStatusCode;
5   import com.moveatis.interfaces.AnonUser;
6   import com.moveatis.interfaces.Application;
7   import com.moveatis.interfaces.Role;
8   import com.moveatis.interfaces.Session;
9   import com.moveatis.user.AnonUserEntity;
10  import java.io.Serializable;
11  import java.util.Calendar;
12  import javax.ejb.Stateless;
13  import javax.inject.Inject;
14  import org.slf4j.Logger;
15  import org.slf4j.LoggerFactory;
16
17  /**
18   * The class does the installation task of the Moveatis application.
19   *
20   * @author Sami Kallio <phinaliumz at outlook.com>
21   */
22  @Stateless
23  public class InstallationBean implements Serializable {
24
25      private static final Logger LOGGER = LoggerFactory.getLogger(InstallationBean.
            class);
```

```
26      private static final String REPORT_EMAIL="sami.m.j.kallio@student.jyu.fi";
27
28      private ApplicationEntity applicationEntity;
29
30      @Inject
31      private Application applicationEJB;
32
33      @Inject
34      private Role roleEJB;
35
36      @Inject
37      private AnonUser anonUserEJB;
38
39      @Inject
40      private Session sessionEJB;
41
42      public InstallationBean() {
43
44      }
45
46      /**
47       * The method creates the application, which includes setting the
48       * installation date, adding the user to superusers, and adding the
49       * reporting email address, where possible error reports are sent.
50       *
51       * @return enum, which result of the installation.
52       */
53      public ApplicationStatusCode createApplication() {
54
55          if(!applicationEJB.checkInstalled()) {
56
57              applicationEntity = new ApplicationEntity();
58              applicationEntity.setApplicationInstalled(Calendar.getInstance().getTime
                    ());
59              applicationEntity.setSuperUsers(roleEJB.listSuperusers());
60              applicationEntity.setReportEmail(REPORT_EMAIL);
61              applicationEJB.create(applicationEntity);
62
63              AnonUserEntity anonEntity = new AnonUserEntity();
64              anonEntity.setCreator(sessionEJB.getLoggedIdentifiedUser());
65              anonUserEJB.create(anonEntity);
66
67              return ApplicationStatusCode.INSTALLATION_OK;
68          }
69
70          return ApplicationStatusCode.ALREADY_INSTALLED;
71      }
72  }
```

1.6 application/RedirectURLs.java

```
1
2   package com.moveatis.application;
3
4   /**
```

```
 * The class has the URIs for redirection within the Jyväskylä University login system.
 * @author Sami Kallio <phinaliumz at outlook.com>
 */
public class RedirectURLs {

    public static final String HOME_URI = "https://moveatis.sport.jyu.fi";
    public static final String SHIBBOLETH_REDIRECT_SECURE_URI = "https://moveatis.sport.jyu.fi/moveatis/secure";
    public static final String LOCALHOST_REDIRECT_SECURE_URI = "http://localhost:8080/moveatis/jyutesting/";
    public static final String LOCALHOST_HOME_URI = "http://localhost:8080/moveatis";
    public static final String SHIBBOLET_LOGOUT_URL =
          "https://moveatis.sport.jyu.fi/Shibboleth.sso/Logout?return="
          + "https://login.jyu.fi/sso/logout.php?return="
          + "https%3A%2F%2Fmoveatis.sport.jyu.fi";

    public static final String CONTROL_PAGE_URI = "https://moveatis.sport.jyu.fi/app/control/";
    public static final String ERROR_PAGE_URI = "https://moveatis.sport.jyu.fi/error";
    public static final String SMTP_HOST="localhost";

    public RedirectURLs() {

    }
}
```

1.7 category/CategoryBean.java

```
package com.moveatis.category;

import com.moveatis.abstracts.AbstractBean;
import com.moveatis.interfaces.Category;
import com.moveatis.interfaces.CategorySet;
import javax.ejb.Stateless;
import javax.inject.Inject;
import javax.persistence.EntityManager;
import javax.persistence.NoResultException;
import javax.persistence.PersistenceContext;
import javax.persistence.TypedQuery;

/**
 * The EJB manages the Category entity.
 * @author Sami Kallio <phinaliumz at outlook.com>
 */
@Stateless
public class CategoryBean extends AbstractBean<CategoryEntity> implements Category {

    @PersistenceContext(unitName = "MOVEATIS_PERSISTENCE")
    private EntityManager em;

    @Inject
```

```
24     private CategorySet categorySetEJB;
25
26     @Override
27     protected EntityManager getEntityManager() {
28        return em;
29     }
30
31     public CategoryBean() {
32        super(CategoryEntity.class);
33     }
34
35     /**
36      * The method is used for finding the category with the certain label.
37      * @param label The label to find the category by.
38      * @return The found category or null.
39      */
40     @Override
41     public CategoryEntity findByLabel(String label) {
42        TypedQuery<CategoryEntity> query = em.createNamedQuery("Category.findByLabel
              ", CategoryEntity.class);
43        query.setParameter("label", label);
44        try {
45           CategoryEntity categoryEntity = query.getSingleResult();
46           return categoryEntity;
47        } catch(NoResultException nre) {
48           return null;
49        }
50     }
51
52     /**
53      * Removes the category from the category set.
54      *
55      * @param whichCategorySet The category set the category belongs to.
56      * @param whichCategory The category to be removed.
57      */
58     @Override
59     public void removeFromCategorySet(CategorySetEntity whichCategorySet,
           CategoryEntity whichCategory) {
60        super.remove(whichCategory);
61        categorySetEJB.removeCategoryFromCategorySet(whichCategorySet, whichCategory
              );
62        whichCategory.setCategorySet(null);
63        super.edit(whichCategory);
64     }
65 }
```

1.8 category/CategoryEntity.java

```
1  package com.moveatis.category;
2
3  import com.moveatis.abstracts.BaseEntity;
4  import com.moveatis.label.LabelEntity;
5  import java.io.Serializable;
6  import static javax.persistence.CascadeType.MERGE;
7  import static javax.persistence.CascadeType.PERSIST;
```

```
8  import javax.persistence.Entity;
9  import static javax.persistence.FetchType.EAGER;
10 import javax.persistence.ManyToOne;
11 import javax.persistence.NamedQuery;
12 import javax.persistence.Table;
13
14 /**
15  * The entity represents the categories of an observation in the database.
16  * @author Sami Kallio <phinaliumz at outlook.com>
17  */
18 @Table(name="CATEGORY")
19 @Entity
20 @NamedQuery(name="Category.findByLabel", query="SELECT category FROM
       CategoryEntity category WHERE category.label = :label")
21 public class CategoryEntity extends BaseEntity implements Serializable {
22
23    private static final long serialVersionUID = 1L;
24
25    @ManyToOne(fetch=EAGER, cascade={PERSIST, MERGE})
26    private LabelEntity label;
27
28    private String description;
29
30    @ManyToOne
31    private CategorySetEntity categorySet;
32
33    private Integer orderNumber;
34    private Boolean canOverlap;
35    private CategoryType categoryType;
36
37    @Override
38    public Long getId() {
39       return id;
40    }
41
42    @Override
43    public void setId(Long id) {
44       this.id = id;
45    }
46
47    public LabelEntity getLabel() {
48       return label;
49    }
50
51    public void setLabel(LabelEntity label) {
52       this.label = label;
53    }
54
55    public CategorySetEntity getCategorySet() {
56       return categorySet;
57    }
58
59    public void setCategorySet(CategorySetEntity categorySet) {
60       this.categorySet = categorySet;
61    }
62
63    public Boolean getCanOverlap() {
64       return canOverlap;
```

```
65      }
66
67      public void setCanOverlap(Boolean canOverlap) {
68          this.canOverlap = canOverlap;
69      }
70
71      public String getDescription() {
72          return description;
73      }
74
75      public void setDescription(String description) {
76          this.description = description;
77      }
78
79      public Integer getOrderNumber() {
80          return orderNumber;
81      }
82
83      public void setOrderNumber(Integer orderNumber) {
84          this.orderNumber = orderNumber;
85      }
86
87      public CategoryType getCategoryType() {
88          return categoryType;
89      }
90
91      public void setCategoryType(CategoryType categoryType) {
92          this.categoryType = categoryType;
93      }
94
95      @Override
96      public int hashCode() {
97          int hash = 0;
98          hash += (id != null ? id.hashCode() : 0);
99          return hash;
100     }
101
102     @Override
103     public boolean equals(Object object) {
104         return object instanceof CategoryEntity;
105     }
106
107     @Override
108     public String toString() {
109         return "com.moveatis.category.Category[ id=" + id + " ]";
110     }
111
112 }
```

1.9 category/CategorySetBean.java

```
1  package com.moveatis.category;
2
3  import com.moveatis.event.EventGroupEntity;
4  import com.moveatis.abstracts.AbstractBean;
```

```
5  import com.moveatis.interfaces.AnonUser;
6  import javax.ejb.Stateless;
7  import javax.persistence.EntityManager;
8  import javax.persistence.PersistenceContext;
9  import com.moveatis.interfaces.CategorySet;
10 import com.moveatis.interfaces.EventGroup;
11 import java.util.HashSet;
12 import java.util.List;
13 import java.util.Map;
14 import java.util.Set;
15 import javax.inject.Inject;
16 import org.slf4j.Logger;
17 import org.slf4j.LoggerFactory;
18
19 /**
20  * The EJB manages the CategorySet entities.
21  *
22  * @author Sami Kallio <phinaliumz at outlook.com>
23  */
24 @Stateless
25 public class CategorySetBean extends AbstractBean<CategorySetEntity> implements
       CategorySet {
26
27    private static final Logger LOGGER = LoggerFactory.getLogger(CategorySetBean.
          class);
28
29    @PersistenceContext(unitName = "MOVEATIS_PERSISTENCE")
30    private EntityManager em;
31
32    @Inject
33    private AnonUser anonUserEJB;
34
35    @Inject
36    private EventGroup eventGroupEJB;
37
38    @Override
39    protected EntityManager getEntityManager() {
40       return em;
41    }
42
43    public CategorySetBean() {
44       super(CategorySetEntity.class);
45    }
46
47    /**
48     * Sets the category set removal date and removes the category set from event
            groups.
49     * @param categorySetEntity The category set entity to be removed.
50     */
51    @Override
52    public void remove(CategorySetEntity categorySetEntity) {
53       super.remove(categorySetEntity);
54       eventGroupEJB.removeCategorySetEntityFromEventGroups(categorySetEntity);
55       categorySetEntity.setEventGroupEntity(null);
56       super.edit(categorySetEntity);
57    }
58
59    /**
```

```
60      * Finds the category sets that are set as public.
61      * @return Set of all public category set entities.
62      */
63     @Override
64     public Set<CategorySetEntity> findPublicCategorySets() {
65
66        List<EventGroupEntity> publicEventGroups = eventGroupEJB.
              findAllForPublicUser();
67        Set<CategorySetEntity> publicCategorySets = new HashSet<>();
68
69        for(EventGroupEntity eventGroupEntity : publicEventGroups) {
70           Set<CategorySetEntity> eventGroupCategorySets = eventGroupEntity.
                 getCategorySets();
71           publicCategorySets.addAll(eventGroupCategorySets);
72        }
73
74        return publicCategorySets;
75     }
76
77     /**
78      * Removes the category from the category set.
79      * @param categorySet The category set from which the category is removed from.
80      * @param categoryEntity The category to be removed from the category set.
81      */
82     @Override
83     public void removeCategoryFromCategorySet(CategorySetEntity categorySet,
           CategoryEntity categoryEntity) {
84        Map<Integer, CategoryEntity> categories = categorySet.getCategoryEntitys();
85        Integer keyFound = -1;
86
87        for(Integer key : categories.keySet()) {
88           if(categories.get(key).getId().equals(categoryEntity.getId())) {
89              keyFound = key;
90           }
91        }
92
93        if(keyFound > -1) {
94           categories.remove(keyFound);
95        }
96
97        categorySet.setCategoryEntitys(categories);
98        super.edit(categorySet);
99     }
100 }
```

1.10 category/CategorySetEntity.java

```
1  package com.moveatis.category;
2
3  import com.moveatis.abstracts.BaseEntity;
4  import com.moveatis.event.EventGroupEntity;
5  import com.moveatis.user.IdentifiedUserEntity;
6  import java.io.Serializable;
7  import java.util.Map;
8  import static javax.persistence.CascadeType.MERGE;
```

```
9  import static javax.persistence.CascadeType.PERSIST;
10 import javax.persistence.CollectionTable;
11 import javax.persistence.Column;
12 import javax.persistence.Entity;
13 import javax.persistence.FetchType;
14 import javax.persistence.ManyToOne;
15 import javax.persistence.MapKey;
16 import javax.persistence.OneToMany;
17 import javax.persistence.Table;
18
19 /**
20  * The entity represents a category set in the database.
21  * @author Sami Kallio <phinaliumz at outlook.com>
22  */
23 @Entity
24 @Table(name="CATEGORYSET")
25 public class CategorySetEntity extends BaseEntity implements Serializable {
26
27    private static final long serialVersionUID = 1L;
28
29    @OneToMany(mappedBy = "categorySet", cascade = {PERSIST, MERGE}, fetch =
          FetchType.LAZY, targetEntity = CategoryEntity.class)
30    @CollectionTable(name="CATEGORYENTITIES")
31    @MapKey(name="orderNumber")
32    @Column(name="CATEGORYENTITY_ORDERNUMBER")
33    private Map<Integer, CategoryEntity> categoryEntitys;
34
35    @ManyToOne
36    private IdentifiedUserEntity creator;
37
38    @ManyToOne
39    private EventGroupEntity eventGroupEntity;
40
41    private String label;
42    private String description;
43
44    @Override
45    public Long getId() {
46       return id;
47    }
48
49    @Override
50    public void setId(Long id) {
51       this.id = id;
52    }
53
54    public Map<Integer, CategoryEntity> getCategoryEntitys() {
55       return categoryEntitys;
56    }
57
58    public void setCategoryEntitys(Map<Integer, CategoryEntity> categoryEntitys) {
59       this.categoryEntitys = categoryEntitys;
60    }
61
62    public IdentifiedUserEntity getCreator() {
63       return creator;
64    }
65
```

```
66      public void setCreator(IdentifiedUserEntity creator) {
67          this.creator = creator;
68      }
69
70      public EventGroupEntity getEventGroupEntity() {
71          return eventGroupEntity;
72      }
73
74      public void setEventGroupEntity(EventGroupEntity eventGroupEntity) {
75          this.eventGroupEntity = eventGroupEntity;
76      }
77
78      public String getLabel() {
79          return label;
80      }
81
82      public void setLabel(String label) {
83          this.label = label;
84      }
85
86      public String getDescription() {
87          return description;
88      }
89
90      public void setDescription(String description) {
91          this.description = description;
92      }
93
94      @Override
95      public int hashCode() {
96          int hash = 0;
97          hash += (id != null ? id.hashCode() : 0);
98          return hash;
99      }
100
101     @Override
102     public boolean equals(Object object) {
103         if (!(object instanceof CategorySetEntity)) {
104             return false;
105         }
106         CategorySetEntity other = (CategorySetEntity) object;
107         return !((this.id == null && other.id != null) || (this.id != null && !this.
                id.equals(other.id)));
108     }
109
110     @Override
111     public String toString() {
112         return "com.moveatis.category.CategoryTemplateEntity[ id=" + id + " ]";
113     }
114
115 }
```

1.11 category/CategoryType.java

```
1
```

```
2  package com.moveatis.category;
3
4  /**
5   * The enum represents the information of a category. A category can be either
6   * a timed category with a time value, or a counted category with an amount value.
7   * @author Sami Kallio <phinaliumz at outlook.com>
8   */
9  public enum CategoryType {
10    TIMED, COUNTED
11 }
```

1.12 devel/DevelJYULoginBean.java

```
1
2  package com.moveatis.devel;
3
4  import com.moveatis.application.InstallationBean;
5  import com.moveatis.enums.ApplicationStatusCode;
6  import com.moveatis.identityprovider.IdentityProviderBean;
7  import com.moveatis.identityprovider.IdentityProviderInformationEntity;
8  import com.moveatis.interfaces.Application;
9  import com.moveatis.interfaces.Role;
10 import com.moveatis.interfaces.Session;
11 import com.moveatis.interfaces.User;
12 import com.moveatis.user.IdentifiedUserEntity;
13 import java.io.IOException;
14 import javax.inject.Named;
15 import javax.enterprise.context.RequestScoped;
16 import javax.faces.context.FacesContext;
17 import javax.inject.Inject;
18 import javax.servlet.http.HttpServletRequest;
19 import org.slf4j.Logger;
20 import org.slf4j.LoggerFactory;
21
22 /**
23  * The bean is a dummy login bean for development purposes, as it mocks
24  * the Sibboleth identity provider system of Jyväskylä University.
25  *
26  * @author Sami Kallio <phinaliumz at outlook.com>
27  */
28 @Named(value="develJYULoginBean")
29 @RequestScoped
30 public class DevelJYULoginBean {
31
32    private static final Logger LOGGER = LoggerFactory.getLogger(DevelJYULoginBean.
      class);
33
34    private IdentifiedUserEntity userEntity;
35
36    @Inject
37    private Session sessionBean;
38    @Inject
39    private IdentityProviderBean ipBean;
40    @Inject
41    private User userEJB;
```

```
42    @Inject
43    private Role roleEJB;
44    @Inject
45    private InstallationBean installationBean;
46    @Inject
47    private Application applicationEJB;
48
49    private String username;
50    private String givenName;
51    private String affiliation;
52    private final Boolean isLocalhost;
53
54    /** Creates a new instance of DevelJYULoginBean. */
55    public DevelJYULoginBean() {
56
57       isLocalhost = ((HttpServletRequest) FacesContext.getCurrentInstance().
          getExternalContext().getRequest())
58             .getRequestURL().toString().contains("localhost");
59    }
60
61    public String getUsername() {
62       return username;
63    }
64
65    public void setUsername(String username) {
66       this.username = username;
67    }
68
69    public String getGivenName() {
70       return givenName;
71    }
72
73    public void setGivenName(String givenName) {
74       this.givenName = givenName;
75    }
76
77    public String getAffiliation() {
78       return affiliation;
79    }
80
81    public void setAffiliation(String affiliation) {
82       this.affiliation = affiliation;
83    }
84
85    public Boolean getIsLocalhost() {
86       return isLocalhost;
87    }
88
89    public String doLogin() {
90       IdentityProviderInformationEntity ipInformationEntity = ipBean.
            findIpEntityByUsername(username);
91
92       if(ipInformationEntity != null) {
93          userEntity = ipInformationEntity.getIdentifiedUserEntity();
94          sessionBean.setIdentityProviderUser(userEntity);
95       } else {
96          return "fail?faces-redirect=true";
97       }
```

```
98          if(sessionBean.getReturnUri() != null) {
99             try {
100               FacesContext.getCurrentInstance().getExternalContext().redirect(
                     sessionBean.getReturnUri());
101            } catch (IOException ex) {
102               LOGGER.debug("Error in doLogin", ex);
103            }
104         } else {
105            return "/app/control/index?faces-redirect=true";
106         }
107
108         return "fail?faces-redirect=true";
109
110      }
111
112      public String doRegistration() {
113         userEntity = new IdentifiedUserEntity();
114
115         IdentityProviderInformationEntity identityProviderInformationEntity = new
              IdentityProviderInformationEntity();
116         identityProviderInformationEntity.setUsername(username);
117         identityProviderInformationEntity.setAffiliation(affiliation);
118
119         userEntity.setIdentityProviderInformationEntity(
              identityProviderInformationEntity);
120         userEntity.setGivenName(givenName);
121
122         identityProviderInformationEntity.setUserEntity(userEntity);
123
124         userEJB.create(userEntity);
125
126         sessionBean.setIdentityProviderUser(userEntity);
127
128         if(applicationEJB.checkInstalled()) {
129            return "/app/control/index?faces-redirect=true";
130         } else {
131            return "fail?faces-redirect=true";
132         }
133      }
134
135      public String doSuperUserRegistration() {
136         userEntity = new IdentifiedUserEntity();
137
138         IdentityProviderInformationEntity identityProviderInformationEntity = new
              IdentityProviderInformationEntity();
139         identityProviderInformationEntity.setUsername(username);
140         identityProviderInformationEntity.setAffiliation(affiliation);
141
142         userEntity.setIdentityProviderInformationEntity(
              identityProviderInformationEntity);
143         userEntity.setGivenName(givenName);
144
145         identityProviderInformationEntity.setUserEntity(userEntity);
146
147         userEJB.create(userEntity);
148         roleEJB.addSuperuserRoleToUser(userEntity);
149
150         sessionBean.setIdentityProviderUser(userEntity);
```

```
151
152         if(installationBean.createApplication() == ApplicationStatusCode.
                INSTALLATION_OK) {
153            return "/app/control/index?faces-redirect=true";
154         } else {
155            return "fail?faces-redirect=true";
156         }
157      }
158   }
```

1.13 enums/ApplicationStatusCode.java

```
1  package com.moveatis.enums;
2
3  /**
4   * The enum contains the information of the status of the application. The
        application
5   * can be already installed, the installation went fine, or there was an error.
6   *
7   * @author Sami Kallio <phinaliumz at outlook.com>
8   */
9  public enum ApplicationStatusCode {
10
11    ALREADY_INSTALLED, INSTALLATION_OK, INSTALLATION_FAILED
12
13 }
```

1.14 enums/MailStatus.java

```
1  package com.moveatis.enums;
2
3  /**
4   * The enum represents the status of the mail that was sent. The mail was either
5   * sent successfully or the sending failed.
6   *
7   * @author Sami Kallio <phinaliumz at outlook.com>
8   */
9  public enum MailStatus {
10    MAIL_SENT_OK, MAIL_SENT_FAILED
11 }
```

1.15 error/ErrorHandler.java

```
1  package com.moveatis.error;
2
3  import com.moveatis.interfaces.Application;
4  import com.moveatis.interfaces.Mailer;
5  import java.io.IOException;
6  import java.text.MessageFormat;
7  import java.util.Locale;
```

```
8  import java.util.ResourceBundle;
9  import javax.inject.Inject;
10 import javax.servlet.ServletException;
11 import javax.servlet.annotation.WebServlet;
12 import javax.servlet.http.HttpServlet;
13 import javax.servlet.http.HttpServletRequest;
14 import javax.servlet.http.HttpServletResponse;
15
16 /**
17  *
18  * The servlet handles the errors, that might be thrown in the course of
19  * executing the software. It mails the error to the superuser mail account and
20  * redirects the user to an error page.
21  *
22  * @author Sami Kallio <phinaliumz at outlook.com>
23  */
24 @WebServlet(name = "ErrorHandler", urlPatterns = {"/ErrorHandler"})
25 public class ErrorHandler extends HttpServlet {
26
27     @Inject
28     private Mailer mailerBean;
29     @Inject
30     private Application applicationEJB;
31
32     /**
33      * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
34      * methods.
35      *
36      * @param request servlet request.
37      * @param response servlet response.
38      * @throws ServletException if a servlet-specific error occurs.
39      * @throws IOException if an I/O error occurs.
40      */
41     protected void processRequest(HttpServletRequest request, HttpServletResponse
            response)
42             throws ServletException, IOException {
43
44         String recipients[] = {applicationEJB.getApplicationEntity().getReportEmail
                ()};
45         ResourceBundle resourceBundle = ResourceBundle.getBundle("errormessages",
                new Locale("fi", "FI"));
46
47         String subject = resourceBundle.getString("emailErrorSubject");
48         String exception = ((Throwable)request.getAttribute("javax.servlet.error.
                exception")).toString();
49         String errorMessage = MessageFormat.format(resourceBundle.getString("
                emailErrorText"), exception);
50
51         mailerBean.sendEmail(recipients, subject, errorMessage);
52         response.sendRedirect("/error");
53     }
54
55     /**
56      * Handles the HTTP <code>GET</code> method.
57      *
58      * @param request servlet request.
59      * @param response servlet response.
60      * @throws ServletException if a servlet-specific error occurs.
```

```
     * @throws IOException if an I/O error occurs.
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
       processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request.
     * @param response servlet response.
     * @throws ServletException if a servlet-specific error occurs.
     * @throws IOException if an I/O error occurs.
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
       processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description.
     */
    @Override
    public String getServletInfo() {
       return "Errorhandler servlet to send error reports to admin and redirect
          user to error page";
    }

}
```

1.16 event/EventBean.java

```
package com.moveatis.event;

import com.moveatis.abstracts.AbstractBean;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import com.moveatis.user.IdentifiedUserEntity;
import javax.ejb.Stateful;
import javax.persistence.TypedQuery;
import com.moveatis.interfaces.Event;
import com.moveatis.observation.ObservationEntity;
import com.moveatis.observation.ObservationEntity_;
import java.util.Set;
import javax.persistence.NoResultException;
import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Predicate;
import javax.persistence.criteria.Root;
```

```
19 import javax.persistence.criteria.SetJoin;
20
21 /**
22  * The EJB manages the events of an user.
23  * @author Sami Kallio <phinaliumz at outlook.com>
24  */
25 @Stateful
26 public class EventBean extends AbstractBean<EventEntity> implements Event {
27
28    @PersistenceContext(unitName = "MOVEATIS_PERSISTENCE")
29    private EntityManager em;
30
31    private EventEntity eventEntity;
32
33    @Override
34    protected EntityManager getEntityManager() {
35       return em;
36    }
37
38    public EventBean() {
39       super(EventEntity.class);
40    }
41
42    /**
43     * Returns a list of the events belonging to the requested user.
44     * @param user The user for which the events should be searched for.
45     * @return A list of the user's events.
46     */
47    @Override
48    public List<EventEntity> findEventsForUser(IdentifiedUserEntity user) {
49       TypedQuery<EventEntity> query = em.createNamedQuery("SceneEntity.findByUser"
             , EventEntity.class);
50       query.setParameter("owner", user);
51       return query.getResultList();
52    }
53
54    /**
55     * Gets the event that is currently associated with the instance of the
           eventBean.
56     * @return the EventEntity associated with the instance.
57     */
58    @Override
59    public EventEntity getEventEntity() {
60       return eventEntity;
61    }
62
63    /**
64     * The method removes the observations from the event of the instance of the
           eventBean.
65     * @param observationEntity The observation to be removed from the event.
66     */
67    @Override
68    public void removeObservation(ObservationEntity observationEntity) {
69
70       CriteriaBuilder cb = em.getCriteriaBuilder();
71       CriteriaQuery<EventEntity> cq = cb.createQuery(EventEntity.class);
72
73       Root<EventEntity> groupRoot = cq.from(EventEntity.class);
```

```
74        SetJoin<EventEntity, ObservationEntity> observationJoin = groupRoot.join(
              EventEntity_.observations);
75        Predicate p = cb.equal(observationJoin.get(ObservationEntity_.id),
              observationEntity.getId());
76
77        cq.select(groupRoot).where(p);
78        TypedQuery<EventEntity> query = em.createQuery(cq);
79        try {
80           EventEntity event = query.getSingleResult();
81           Set<ObservationEntity> observationSets = event.getObservations();
82           observationSets.remove(observationEntity);
83           event.setObservations(observationSets);
84           super.edit(event);
85        } catch(NoResultException nre) {
86           //NoResultException is not an actual exception IMHO ...
87        }
88     }
89  }
```

1.17 event/EventEntity.java

```
1  package com.moveatis.event;
2
3  import com.moveatis.abstracts.BaseEntity;
4  import com.moveatis.observation.ObservationEntity;
5  import com.moveatis.user.AbstractUser;
6  import java.io.Serializable;
7  import java.util.Set;
8  import javax.persistence.Entity;
9  import javax.persistence.FetchType;
10 import javax.persistence.ManyToOne;
11 import javax.persistence.NamedQueries;
12 import javax.persistence.NamedQuery;
13 import javax.persistence.OneToMany;
14 import javax.persistence.OneToOne;
15 import javax.persistence.Table;
16
17 /**
18  * The entity represents the information of the events.
19  * @author Sami Kallio <phinaliumz at outlook.com>
20  */
21 @Entity
22 @Table(name="EVENT")
23 @NamedQueries(
24       @NamedQuery(name="EventEntity.findByCreator", query="SELECT event FROM
              EventEntity event WHERE event.creator = :user")
25 )
26 public class EventEntity extends BaseEntity implements Serializable {
27
28    @OneToOne
29    private EventGroupEntity eventGroup;
30
31    private static final long serialVersionUID = 1L;
32
33    @ManyToOne
```

```
34      private AbstractUser creator;
35
36      @OneToMany(mappedBy = "event", fetch = FetchType.LAZY)
37      private Set<ObservationEntity> observations;
38
39      private String description;
40      private String label;
41
42      @Override
43      public Long getId() {
44          return id;
45      }
46
47      @Override
48      public void setId(Long id) {
49          this.id = id;
50      }
51
52      public EventGroupEntity getEventGroup() {
53          return eventGroup;
54      }
55
56      public void setEventGroup(EventGroupEntity eventGroup) {
57          this.eventGroup = eventGroup;
58      }
59
60      public AbstractUser getCreator() {
61          return creator;
62      }
63
64      public void setCreator(AbstractUser creator) {
65          this.creator = creator;
66      }
67
68      public String getDescription() {
69          return description;
70      }
71
72      public void setDescription(String description) {
73          this.description = description;
74      }
75
76      public String getLabel() {
77          return label;
78      }
79
80      public void setLabel(String label) {
81          this.label = label;
82      }
83
84      public Set<ObservationEntity> getObservations() {
85          return observations;
86      }
87
88      public void setObservations(Set<ObservationEntity> observations) {
89          this.observations = observations;
90      }
91
```

```
92     @Override
93     public int hashCode() {
94        int hash = 0;
95        hash += (id != null ? id.hashCode() : 0);
96        return hash;
97     }
98
99     @Override
100    public boolean equals(Object object) {
101       if (!(object instanceof EventEntity)) {
102          return false;
103       }
104       EventEntity other = (EventEntity) object;
105       return !((this.id == null && other.id != null) || (this.id != null && !this.
              id.equals(other.id)));
106    }
107
108    @Override
109    public String toString() {
110       return "com.moveatis.event.EventEntity[ id=" + id + " ]";
111    }
112
113 }
```

1.18 event/EventGroupBean.java

```
1  package com.moveatis.event;
2
3  import com.moveatis.abstracts.AbstractBean;
4  import com.moveatis.category.CategorySetEntity;
5  import com.moveatis.category.CategorySetEntity_;
6  import com.moveatis.interfaces.AnonUser;
7  import javax.ejb.Stateless;
8  import javax.persistence.EntityManager;
9  import javax.persistence.PersistenceContext;
10 import com.moveatis.interfaces.EventGroup;
11 import com.moveatis.user.AbstractUser;
12 import com.moveatis.user.AbstractUser_;
13 import java.util.List;
14 import java.util.Set;
15 import javax.inject.Inject;
16 import javax.persistence.TypedQuery;
17 import javax.persistence.criteria.CriteriaBuilder;
18 import javax.persistence.criteria.CriteriaQuery;
19 import javax.persistence.criteria.Predicate;
20 import javax.persistence.criteria.Root;
21 import javax.persistence.criteria.SetJoin;
22 import org.slf4j.Logger;
23 import org.slf4j.LoggerFactory;
24
25 /**
26  * The EJB manages the event group entities.
27  *
28  * @author Sami Kallio <phinaliumz at outlook.com>
29  */
```

```
30 @Stateless
31 public class EventGroupBean extends AbstractBean<EventGroupEntity> implements
       EventGroup {
32
33    private static final Logger LOGGER = LoggerFactory.getLogger(EventGroupBean.
          class);
34
35    @PersistenceContext(unitName = "MOVEATIS_PERSISTENCE")
36    private EntityManager em;
37
38    @Inject
39    private AnonUser anonUserEJB;
40
41    @Override
42    protected EntityManager getEntityManager() {
43       return em;
44    }
45
46    public EventGroupBean() {
47       super(EventGroupEntity.class);
48    }
49
50    /**
51     * Finds and returns a list of the event groups belonging to the given user.
52     * @param owner The user whose event groups are searched for.
53     * @return A list of the event groups.
54     */
55    @Override
56    public List<EventGroupEntity> findAllForOwner(AbstractUser owner) {
57       TypedQuery<EventGroupEntity> query = em.createNamedQuery("
             findEventGroupByOwner", EventGroupEntity.class);
58       query.setParameter("ownerEntity", owner);
59       return query.getResultList();
60    }
61
62    /**
63     * Finds and returns a list of event groups, which the given user has access to.
64     * @param user The user whose event groups are searched for.
65     * @return A list of the event groups.
66     */
67    @Override
68    public List<EventGroupEntity> findAllForUser(AbstractUser user) {
69       return findAllForAbstractUser(user);
70    }
71
72    /**
73     * Finds and returns the event groups that are allowed for public use.
74     * @return A list of the event groups.
75     */
76    @Override
77    public List<EventGroupEntity> findAllForPublicUser() {
78       return findAllForAbstractUser(anonUserEJB.find());
79    }
80
81    /**
82     * The method finds and returns the event groups, which the given user
83     * has access to.
84     * @param user The user to search event groups for.
```

```
85      * @return A list of the event groups.
86      */
87     private List<EventGroupEntity> findAllForAbstractUser(AbstractUser user) {
88        CriteriaBuilder cb = em.getCriteriaBuilder();
89        CriteriaQuery<EventGroupEntity> cq = cb.createQuery(EventGroupEntity.class);
90
91        Root<EventGroupEntity> groupRoot = cq.from(EventGroupEntity.class);
92
93        SetJoin<EventGroupEntity, AbstractUser> userJoin = groupRoot.join(
              EventGroupEntity_.users);
94        Predicate p = cb.equal(userJoin.get(AbstractUser_.id), user.getId());
95
96        cq.select(groupRoot).where(p);
97
98        TypedQuery<EventGroupEntity> query = em.createQuery(cq);
99
100       return query.getResultList();
101    }
102    /**
103     * The method removes the given category set from all of the event
104     * groups that have the category set.
105     * @param categorySetEntity The category set to be removed from the event groups
            .
106     */
107    @Override
108    public void removeCategorySetEntityFromEventGroups(CategorySetEntity
          categorySetEntity) {
109       CriteriaBuilder cb = em.getCriteriaBuilder();
110       CriteriaQuery<EventGroupEntity> cq = cb.createQuery(EventGroupEntity.class);
111
112       Root<EventGroupEntity> groupRoot = cq.from(EventGroupEntity.class);
113       SetJoin<EventGroupEntity, CategorySetEntity> categoryJoin = groupRoot.join(
              EventGroupEntity_.categorySets);
114       Predicate p = cb.equal(categoryJoin.get(CategorySetEntity_.id),
              categorySetEntity.getId());
115
116       cq.select(groupRoot).where(p);
117       TypedQuery<EventGroupEntity> query = em.createQuery(cq);
118       List<EventGroupEntity> eventGroups = query.getResultList();
119       if (!eventGroups.isEmpty()) {
120          for (EventGroupEntity eventGroup : eventGroups) {
121             Set<CategorySetEntity> categorySets = eventGroup.getCategorySets();
122             categorySets.remove(categorySetEntity);
123             eventGroup.setCategorySets(categorySets);
124             super.edit(eventGroup);
125          }
126       }
127    }
128 }
```

1.19 event/EventGroupEntity.java

```
1  package com.moveatis.event;
2
3  import com.moveatis.abstracts.BaseEntity;
```

```
4  import com.moveatis.category.CategorySetEntity;
5  import com.moveatis.groupkey.GroupKeyEntity;
6  import com.moveatis.user.AbstractUser;
7  import java.io.Serializable;
8  import java.util.Set;
9  import static javax.persistence.CascadeType.PERSIST;
10 import javax.persistence.Entity;
11 import static javax.persistence.FetchType.EAGER;
12 import javax.persistence.ManyToOne;
13 import javax.persistence.NamedQueries;
14 import javax.persistence.NamedQuery;
15 import javax.persistence.OneToMany;
16 import javax.persistence.OneToOne;
17 import javax.persistence.Table;
18
19 /**
20  * The entity represents the information for the event groups of the application.
21  * The event group has just one event, but the class could be extended to support
22  * multiple events. The event group can be identified with a groupkey, which
23  * allows semi-public usage.
24  * @author Sami Kallio <phinaliumz at outlook.com>
25  */
26 @Entity
27 @NamedQueries({
28    @NamedQuery(
29          name = "findEventGroupByOwner",
30          query = "SELECT eventGroup FROM EventGroupEntity eventGroup WHERE
                 eventGroup.owner=:ownerEntity "
31          + "AND eventGroup.removed IS NULL"
32    )
33 })
34 @Table(name = "EVENTGROUP")
35 public class EventGroupEntity extends BaseEntity implements Serializable {
36
37    private static final long serialVersionUID = 1L;
38
39    @OneToMany(mappedBy = "eventGroupEntity", cascade = PERSIST, fetch = EAGER)
40    private Set<CategorySetEntity> categorySets;
41
42    @OneToOne(mappedBy = "eventGroup", cascade = PERSIST)
43    private EventEntity event;
44
45    @OneToOne(cascade = PERSIST)
46    private GroupKeyEntity groupKey;
47
48    @ManyToOne
49    private AbstractUser owner;
50
51    @OneToMany
52    private Set<AbstractUser> users;
53
54    private String label;
55    private String description;
56
57    @Override
58    public Long getId() {
59       return id;
60    }
```

```
61
62      @Override
63      public void setId(Long id) {
64          this.id = id;
65      }
66
67      public AbstractUser getOwner() {
68          return owner;
69      }
70
71      public void setOwner(AbstractUser owner) {
72          this.owner = owner;
73      }
74
75      public Set<AbstractUser> getUsers() {
76          return users;
77      }
78
79      public void setUsers(Set<AbstractUser> users) {
80          this.users = users;
81      }
82
83      public Set<CategorySetEntity> getCategorySets() {
84          return categorySets;
85      }
86
87      public void setCategorySets(Set<CategorySetEntity> categorySets) {
88          this.categorySets = categorySets;
89      }
90
91      public String getLabel() {
92          return label;
93      }
94
95      public void setLabel(String label) {
96          this.label = label;
97      }
98
99      public String getDescription() {
100         return description;
101     }
102
103     public void setDescription(String description) {
104         this.description = description;
105     }
106
107     public EventEntity getEvent() {
108         return event;
109     }
110
111     public void setEvent(EventEntity event) {
112         this.event = event;
113     }
114
115     public GroupKeyEntity getGroupKey() {
116         return groupKey;
117     }
118
```

```
119     public void setGroupKey(GroupKeyEntity groupKey) {
120        this.groupKey = groupKey;
121     }
122
123     @Override
124     public int hashCode() {
125        int hash = 0;
126        hash += (id != null ? id.hashCode() : 0);
127        return hash;
128     }
129
130     @Override
131     public boolean equals(Object object) {
132        if (!(object instanceof EventGroupEntity)) {
133           return false;
134        }
135        EventGroupEntity other = (EventGroupEntity) object;
136        return !((this.id == null && other.id != null) || (this.id != null && !this.
              id.equals(other.id)));
137     }
138
139     @Override
140     public String toString() {
141        return "com.moveatis.event.EventGroupEntity[ id=" + id + " ]";
142     }
143
144 }
```

1.20 exception/InstallationRedirectException.java

```
1  package com.moveatis.exception;
2
3  /**
4   * The exception informs the developers or the admin that something went
5   * wrong with the installation.
6   *
7   * @author Sami Kallio <phinaliumz at outlook.com>
8   */
9  public class InstallationRedirectException extends RuntimeException {
10
11    private static final long serialVersionUID=1L;
12
13    /**
14     * Creates a new instance of <code>InstallationRedirectException</code>
15     * without a detailed message.
16     */
17    public InstallationRedirectException() {
18    }
19
20    /**
21     * Constructs an instance of <code>InstallationRedirectException</code> with
22     * the specified detailed message.
23     *
24     * @param msg the detailed message.
25     */
```

```
26     public InstallationRedirectException(String msg) {
27        super(msg);
28     }
29
30     public InstallationRedirectException(Throwable cause) {
31        super(cause);
32     }
33
34     public InstallationRedirectException(String msg, Throwable cause) {
35        super(msg, cause);
36     }
37
38     public InstallationRedirectException(String msg, Throwable cause, boolean
           enableSuppression,
39            boolean writableStackTrace) {
40        super(msg, cause, enableSuppression, writableStackTrace);
41     }
42 }
```

1.21 export/CSVBuilder.java

```
1  package com.moveatis.export;
2
3  import java.io.IOException;
4  import java.io.OutputStream;
5  import java.io.OutputStreamWriter;
6  import java.nio.charset.StandardCharsets;
7  import org.apache.commons.lang3.StringEscapeUtils;
8
9  /**
10  * Builds CSV formatted data to OutputStream.
11  * @author Ilari Paananen
12  */
13 public class CSVBuilder {
14    private OutputStreamWriter out;
15    private String sep;
16    private boolean atLineBegin;
17
18    /**
19     * Constructs a builder with the given stream and separator.
20     * @param output The stream to write the CSV data to.
21     * @param separator The separator to be used between the fields.
22     */
23    public CSVBuilder(OutputStream output, String separator) {
24       out = new OutputStreamWriter(output, StandardCharsets.UTF_8);
25       sep = separator;
26       atLineBegin = true;
27    }
28
29    /**
30     * Adds the long field to the stream.
31     * @param value The field value.
32     * @return The instance of the CSVBuilder for convenience.
33     * @throws IOException
34     */
```

```
35      public CSVBuilder add(Long value) throws IOException {
36         writeSep();
37         write(value.toString());
38         return this;
39      }
40
41      /**
42       * Adds the long field followed by a percent character (%) to the stream.
43       * @param value Field value.
44       * @return The instance of the CSVBuilder for convenience.
45       * @throws IOException
46       */
47      public CSVBuilder addPercent(Long value) throws IOException {
48         writeSep();
49         write(value + "%");
50         return this;
51      }
52
53      /**
54       * Escapes the string field and adds it to the stream.
55       * @param value The field value.
56       * @return The instance of the CSVBuilder for convenience.
57       * @throws IOException
58       */
59      public CSVBuilder add(String value) throws IOException {
60         writeSep();
61         if (value != null)
62            write("\"" + StringEscapeUtils.escapeCsv(value) + "\"");
63         else
64            write("\"\"");
65         return this;
66      }
67
68      /**
69       * Adds a CSV new line to the stream.
70       * @return The instance of the CSVBuilder for convenience.
71       * @throws IOException
72       */
73      public CSVBuilder newLine() throws IOException {
74         write("\r\n");
75         atLineBegin = true;
76         return this;
77      }
78
79      /**
80       * Closes the writer that uses the OutputStream given in the constructor.
81       * @throws IOException
82       */
83      public void close() throws IOException {
84         out.close();
85      }
86
87      /**
88       * Writes a string to the output stream. Makes it easier to replace member
89       * OutputStreamWriter with something else if needed.
90       * @param s String to write.
91       * @throws IOException
92       */
```

```
93      private void write(String s) throws IOException {
94         out.write(s);
95      }
96
97      /**
98       * Writes separator if we aren't at the begin of a line.
99       * @throws IOException
100      */
101     private void writeSep() throws IOException {
102        if (atLineBegin)
103           atLineBegin = false;
104        else
105           write(sep);
106     }
107  }
```

1.22 export/CSVFileBuilder.java

```
1   package com.moveatis.export;
2
3   import com.moveatis.observation.ObservationCategory;
4   import com.moveatis.observation.ObservationEntity;
5   import com.moveatis.records.RecordEntity;
6   import java.io.IOException;
7   import java.io.OutputStream;
8   import java.util.Comparator;
9   import java.util.List;
10  import java.util.Map;
11  import java.util.TreeMap;
12  import org.slf4j.Logger;
13  import org.slf4j.LoggerFactory;
14
15  /**
16   * Builds a CSV file from an observation.
17   * @author Ilari Paananen
18   */
19  public class CSVFileBuilder {
20
21     private static final Logger LOGGER = LoggerFactory.getLogger(CSVFileBuilder.
          class);
22
23     private long totalCount;
24     private long totalDuration;
25
26     /**
27      * Builds a CSV file from the given observation and writes it to the output
           stream.
28      * @param out The stream to write the CSV data to.
29      * @param obs The observation to be built.
30      * @param separator The separator used between the CSV fields.
31      * @throws IOException
32      */
33     public void buildCSV(OutputStream out, ObservationEntity obs, String separator)
           throws IOException {
34
```

```
35        Long obsDuration = obs.getDuration();
36        Map<ObservationCategory, CategorySummaryItem> summaryItems =
37              computeCategorySummaryItems(obs);
38        List<RecordEntity> records = obs.getRecords();
39
40        CSVBuilder csv = new CSVBuilder(out, separator);
41
42        csv.add("Observation info").newLine();
43        csv.newLine();
44
45        csv.add("Attribute").add("Value").newLine();
46
47        csv.add("name").add(obs.getName()).newLine();
48        csv.add("target").add(obs.getTarget()).newLine();
49        csv.add("description").add(obs.getDescription()).newLine();
50        csv.add("duration (ms)").add(obsDuration).newLine();
51        csv.add("records").add(totalCount).newLine();
52
53        csv.newLine();
54        csv.newLine();
55
56        csv.add("Summary").newLine();
57        csv.newLine();
58
59        csv.add("Category").add("Count").add("Count %").add("Duration (ms)").add("
              Duration %").newLine();
60
61        for (Map.Entry<ObservationCategory, CategorySummaryItem> entry :
              summaryItems.entrySet()) {
62           String category = entry.getKey().getName();
63           CategorySummaryItem item = entry.getValue();
64           long countPercent = (long)(item.count * 100.0 / totalCount + 0.5);
65           long durationPercent = (long)(item.duration * 100.0 / obsDuration + 0.5);
66           csv.add(category).add(item.count).addPercent(countPercent).add(item.
                 duration).addPercent(durationPercent).newLine();
67        }
68
69        csv.newLine();
70        csv.newLine();
71
72        csv.add("Recordings").newLine();
73        csv.newLine();
74
75        csv.add("Category").add("Start time (ms)").add("End time (ms)").add("
              Duration (ms)").newLine();
76
77        for (RecordEntity record : records) {
78           String category = record.getCategory().getName();
79           Long startTime = record.getStartTime();
80           Long endTime = record.getEndTime();
81           csv.add(category).add(startTime).add(endTime).add(endTime - startTime).
                 newLine();
82        }
83
84        csv.close();
85     }
86
87     /**
```

```
88      * Computes category summary items from observation.
89      * @param obs Observation
90      * @return Summary items mapped by category.
91      */
92     private Map<ObservationCategory, CategorySummaryItem>
           computeCategorySummaryItems(ObservationEntity obs) {
93
94        // TODO: Categories should be in the same order as when the
95        // observation was conducted.
96        // Observation should contain this info, but does not yet.
97        Map<ObservationCategory, CategorySummaryItem> summaryItems = new TreeMap<>(
98              new Comparator<ObservationCategory>() {
99                 @Override
100                public int compare(ObservationCategory c1, ObservationCategory c2)
                       {
101                   return c1.getTag().compareTo(c2.getTag());
102                }
103             }
104       );
105
106       List<RecordEntity> records = obs.getRecords();
107
108       totalCount = 0;
109       totalDuration = 0; // NOTE: Not used anywhere.
110
111       for (RecordEntity record : records) {
112          ObservationCategory category = record.getCategory();
113          Long deltaTime = record.getEndTime() - record.getStartTime();
114
115          CategorySummaryItem item = summaryItems.get(category);
116          if (item == null) {
117             item = new CategorySummaryItem();
118             summaryItems.put(category, item);
119          }
120
121          item.count++;
122          item.duration += deltaTime;
123
124          totalCount++;
125          totalDuration += deltaTime;
126       }
127
128       return summaryItems;
129    }
130
131    /**
132     * Private class for category summary info.
133     */
134    private static class CategorySummaryItem {
135       public long count = 0;
136       public long duration = 0;
137    }
138 }
```

1.23 filters/ControlFilter.java

```
1  package com.moveatis.filters;
2
3  import com.moveatis.application.RedirectURLs;
4  import com.moveatis.interfaces.Session;
5  import java.io.IOException;
6  import java.util.Locale;
7  import java.util.ResourceBundle;
8  import javax.inject.Inject;
9  import javax.servlet.Filter;
10 import javax.servlet.FilterChain;
11 import javax.servlet.FilterConfig;
12 import javax.servlet.ServletException;
13 import javax.servlet.ServletRequest;
14 import javax.servlet.ServletResponse;
15 import javax.servlet.annotation.WebFilter;
16 import javax.servlet.http.HttpServletRequest;
17 import javax.servlet.http.HttpServletResponse;
18 import org.slf4j.Logger;
19 import org.slf4j.LoggerFactory;
20
21 /**
22  * The filter controls access to the control page only for the identified users.
23  * @author Sami Kallio <phinaliumz at outlook.com>
24  */
25 @WebFilter(filterName = "ControlFilter", urlPatterns = {"/app/control/*"})
26 public class ControlFilter implements Filter {
27
28    private static final Logger LOGGER = LoggerFactory.getLogger(ControlFilter.
          class);
29
30    // The filter configuration object we are associated with. If
31    // this value is null, this filter instance is not currently
32    // configured.
33    private FilterConfig filterConfig = null;
34
35    @Inject
36    private Session sessionBean;
37
38    public ControlFilter() {
39    }
40
41    private void doBeforeProcessing(ServletRequest request, ServletResponse
          response)
42          throws IOException, ServletException {
43
44    }
45
46    private void doAfterProcessing(ServletRequest request, ServletResponse response
          )
47          throws IOException, ServletException {
48
49    }
50
51    /**
52     *
53     * @param request The servlet request to be processed.
54     * @param response The servlet response to be created.
```

```
55      * @param chain The filter chain to be processed.
56      *
57      * @exception IOException if an input or output error occurs.
58      * @exception ServletException if a servlet error occurs.
59      */
60     @Override
61     public void doFilter(ServletRequest request, ServletResponse response,
62           FilterChain chain)
63           throws IOException, ServletException {
64
65        doBeforeProcessing(request, response);
66
67        if(sessionBean.isIdentifiedUser()) {
68
69        } else {
70
71           Locale locale = ((HttpServletRequest)request).getLocale();
72           ResourceBundle messages = ResourceBundle.getBundle("com.moveatis.messages
                .Messages", locale);
73
74           ((HttpServletResponse)response).setStatus(HttpServletResponse.
                SC_FORBIDDEN);
75           ((HttpServletResponse)response).sendError(HttpServletResponse.
                SC_FORBIDDEN, messages.getString("filter.forbidden"));
76           return;
77        }
78
79        Throwable problem = null;
80        try {
81           chain.doFilter(request, response);
82        } catch (Throwable t) {
83           // If an exception is thrown somewhere down the filter chain,
84           // we still want to execute our after processing, and then
85           // rethrow the problem after that.
86           problem = t;
87           t.printStackTrace();
88        }
89
90        doAfterProcessing(request, response);
91
92        // If there was a problem, we want to rethrow it if it is
93        // a known type, otherwise log it.
94        if (problem != null) {
95           if (problem instanceof ServletException) {
96              throw (ServletException) problem;
97           }
98           if (problem instanceof IOException) {
99              throw (IOException) problem;
100          }
101          sendProcessingError(problem, response);
102       }
103    }
104
105    /**
106     * Returns the filter configuration object for the filter.
107     */
108    public FilterConfig getFilterConfig() {
109       return (this.filterConfig);
```

```
110     }
111
112     /**
113      * Sets the filter configuration object for the filter.
114      *
115      * @param filterConfig The filter configuration object.
116      */
117     public void setFilterConfig(FilterConfig filterConfig) {
118        this.filterConfig = filterConfig;
119     }
120
121     /**
122      * Destroys the filter.
123      */
124     public void destroy() {
125     }
126
127     /**
128      * Initializes the filter.
129      */
130     @Override
131     public void init(FilterConfig filterConfig) {
132        this.filterConfig = filterConfig;
133        if (filterConfig != null) {
134
135        }
136     }
137
138     /**
139      * Returns a string representation of the object.
140      */
141     @Override
142     public String toString() {
143        if (filterConfig == null) {
144           return ("ControlFilter()");
145        }
146        StringBuilder sb = new StringBuilder("ControlFilter(");
147        sb.append(filterConfig);
148        sb.append(")");
149        return (sb.toString());
150     }
151
152     private void sendProcessingError(Throwable t, ServletResponse response) {
153        LOGGER.error("Error in controlpage filtering", t);
154
155        try {
156           ((HttpServletResponse)response).sendRedirect(RedirectURLs.ERROR_PAGE_URI)
                ;
157        } catch (IOException ex) {
158           LOGGER.error("Error in redirecting", ex);
159        }
160     }
161
162     public void log(String msg) {
163        filterConfig.getServletContext().log(msg);
164     }
165 }
```

1.24 filters/LoginFilter.java

```
1  package com.moveatis.filters;
2
3  import com.moveatis.application.RedirectURLs;
4  import com.moveatis.interfaces.Session;
5  import java.io.IOException;
6  import javax.inject.Inject;
7  import javax.servlet.Filter;
8  import javax.servlet.FilterChain;
9  import javax.servlet.FilterConfig;
10 import javax.servlet.ServletContext;
11 import javax.servlet.ServletException;
12 import javax.servlet.ServletRequest;
13 import javax.servlet.ServletResponse;
14 import javax.servlet.annotation.WebFilter;
15 import javax.servlet.http.HttpServletResponse;
16 import org.slf4j.Logger;
17 import org.slf4j.LoggerFactory;
18
19 /**
20  *
21  * The filter checks that the pages in /app folder are accessed only
22  * through the frontpage.
23  *
24  * @author Sami Kallio <phinaliumz at outlook.com>
25  *
26  */
27 @WebFilter(filterName = "LoginFilter", urlPatterns = {"/app/*"})
28 public class LoginFilter implements Filter {
29
30    private static final Logger LOGGER = LoggerFactory.getLogger(LoginFilter.class)
          ;
31
32    private static final boolean DEBUG = false;
33    private FilterConfig filterConfig = null;
34    private ServletContext context = null;
35
36    @Inject
37    private Session sessionBean;
38
39    public LoginFilter() {
40    }
41
42    private void doBeforeProcessing(ServletRequest request, ServletResponse
          response)
43          throws IOException, ServletException {
44
45    }
46
47    private void doAfterProcessing(ServletRequest request, ServletResponse response
          )
48          throws IOException, ServletException {
49
50    }
51
52    /**
53     *
```

```
54      * @param request The servlet request to be processed.
55      * @param response The servlet response to be created.
56      * @param chain The filter chain to be processed.
57      *
58      * @exception IOException if an input or output error occurs.
59      * @exception ServletException if a servlet error occurs.
60      */
61     @Override
62     public void doFilter(ServletRequest request, ServletResponse response,
63           FilterChain chain)
64           throws IOException, ServletException {
65
66        doBeforeProcessing(request, response);
67
68        if(!sessionBean.isLoggedIn()) {
69           ((HttpServletResponse)response).sendRedirect("/" + context.getContextPath
                 () + "/index.xhtml");
70           return;
71        }
72
73        Throwable problem = null;
74        try {
75           chain.doFilter(request, response);
76        } catch (IOException | ServletException t) {
77           problem = t;
78        }
79
80        doAfterProcessing(request, response);
81
82        // If there was a problem, we want to rethrow it if it is
83        // a known type, otherwise log it.
84        if (problem != null) {
85           if (problem instanceof ServletException) {
86              throw (ServletException) problem;
87           }
88           if (problem instanceof IOException) {
89              throw (IOException) problem;
90           }
91           sendProcessingError(problem, response);
92        }
93     }
94
95     /**
96      * Returns the filter configuration object for the filter.
97      */
98     public FilterConfig getFilterConfig() {
99        return (this.filterConfig);
100    }
101
102    /**
103     * Sets the filter configuration object for the filter.
104     *
105     * @param filterConfig The filter configuration object.
106     */
107    public void setFilterConfig(FilterConfig filterConfig) {
108       this.filterConfig = filterConfig;
109    }
110
```

```
111     /**
112      * Destroys the filter.
113      */
114     @Override
115     public void destroy() {
116     }
117 
118     /**
119      * Initializes the filter.
120      */
121     @Override
122     public void init(FilterConfig filterConfig) {
123        this.filterConfig = filterConfig;
124        if (filterConfig != null) {
125           if (DEBUG) {
126              log("LoginFilter:Initializing filter");
127              this.context = filterConfig.getServletContext();
128           }
129        }
130     }
131 
132     /**
133      * Returns a string representation of the object.
134      */
135     @Override
136     public String toString() {
137        if (filterConfig == null) {
138           return ("LoginFilter()");
139        }
140        StringBuilder sb = new StringBuilder("LoginFilter(");
141        sb.append(filterConfig);
142        sb.append(")");
143        return (sb.toString());
144     }
145 
146     private void sendProcessingError(Throwable t, ServletResponse response) {
147        LOGGER.error("Error in login filtering", t);
148 
149        try {
150           ((HttpServletResponse)response).sendRedirect(RedirectURLs.ERROR_PAGE_URI)
                ;
151        } catch (IOException ex) {
152           LOGGER.error("Error in redirecting", ex);
153        }
154     }
155 
156     public void log(String msg) {
157        filterConfig.getServletContext().log(msg);
158     }
159 }
```

1.25 filters/SuperUserFilter.java

```
1 package com.moveatis.filters;
2 
```

```
3  import com.moveatis.application.RedirectURLs;
4  import com.moveatis.interfaces.Role;
5  import com.moveatis.interfaces.Session;
6  import java.io.IOException;
7  import java.util.Locale;
8  import java.util.ResourceBundle;
9  import javax.inject.Inject;
10 import javax.servlet.Filter;
11 import javax.servlet.FilterChain;
12 import javax.servlet.FilterConfig;
13 import javax.servlet.ServletException;
14 import javax.servlet.ServletRequest;
15 import javax.servlet.ServletResponse;
16 import javax.servlet.annotation.WebFilter;
17 import javax.servlet.http.HttpServletRequest;
18 import javax.servlet.http.HttpServletResponse;
19 import org.slf4j.Logger;
20 import org.slf4j.LoggerFactory;
21
22 /**
23  * The filter allows only identified users that have the superuser role
24  * to access the superuser page.
25  *
26  * @author Sami Kallio <phinaliumz at outlook.com>
27  */
28 @WebFilter(filterName = "SuperUserFilter", urlPatterns = {"/app/superuser/*"})
29 public class SuperUserFilter implements Filter {
30
31    private static final Logger LOGGER = LoggerFactory.getLogger(SuperUserFilter.
      class);
32
33    // The filter configuration object we are associated with. If
34    // this value is null, this filter instance is not currently
35    // configured.
36    private FilterConfig filterConfig = null;
37
38    @Inject
39    private Session sessionBean;
40    @Inject
41    private Role roleBean;
42
43    public SuperUserFilter() {
44    }
45
46    private void doBeforeProcessing(ServletRequest request, ServletResponse
      response)
47          throws IOException, ServletException {
48
49    }
50
51    private void doAfterProcessing(ServletRequest request, ServletResponse response
      )
52          throws IOException, ServletException {
53    }
54
55    /**
56     *
57     * @param request The servlet request to be processed.
```

```
58      * @param response The servlet response to be created.
59      * @param chain The filter chain to be processed.
60      *
61      * @exception IOException if an input or output error occurs.
62      * @exception ServletException if a servlet error occurs.
63      */
64     public void doFilter(ServletRequest request, ServletResponse response,
65           FilterChain chain)
66           throws IOException, ServletException {
67
68        doBeforeProcessing(request, response);
69
70        if(sessionBean.isIdentifiedUser()) {
71           if(roleBean.checkIfUserIsSuperUser(sessionBean.getLoggedIdentifiedUser())
                 ) {
72
73           } else {
74              Locale locale = ((HttpServletRequest)request).getLocale();
75              ResourceBundle messages = ResourceBundle.getBundle("com.moveatis.
                    messages.Messages", locale);
76
77              ((HttpServletResponse)response).setStatus(HttpServletResponse.
                    SC_FORBIDDEN);
78              ((HttpServletResponse)response).sendError(HttpServletResponse.
                    SC_FORBIDDEN, messages.getString("filter.forbidden"));
79              return;
80           }
81        } else {
82
83           Locale locale = ((HttpServletRequest)request).getLocale();
84           ResourceBundle messages = ResourceBundle.getBundle("com.moveatis.messages
                 .Messages", locale);
85
86           ((HttpServletResponse)response).setStatus(HttpServletResponse.
                 SC_FORBIDDEN);
87           ((HttpServletResponse)response).sendError(HttpServletResponse.
                 SC_FORBIDDEN, messages.getString("filter.forbidden"));
88           return;
89        }
90
91        Throwable problem = null;
92        try {
93           chain.doFilter(request, response);
94        } catch (Throwable t) {
95           // If an exception is thrown somewhere down the filter chain,
96           // we still want to execute our after processing, and then
97           // rethrow the problem after that.
98           problem = t;
99           t.printStackTrace();
100       }
101
102       doAfterProcessing(request, response);
103
104       // If there was a problem, we want to rethrow it if it is
105       // a known type, otherwise log it.
106       if (problem != null) {
107          if (problem instanceof ServletException) {
108             throw (ServletException) problem;
```

```
109             }
110             if (problem instanceof IOException) {
111                 throw (IOException) problem;
112             }
113             sendProcessingError(problem, response);
114         }
115     }
116 
117     /**
118      * Returns the filter configuration object for the filter.
119      */
120     public FilterConfig getFilterConfig() {
121         return (this.filterConfig);
122     }
123 
124     /**
125      * Sets the filter configuration object for the filter.
126      *
127      * @param filterConfig The filter configuration object.
128      */
129     public void setFilterConfig(FilterConfig filterConfig) {
130         this.filterConfig = filterConfig;
131     }
132 
133     /**
134      * Destroys the filter.
135      */
136     @Override
137     public void destroy() {
138     }
139 
140     /**
141      * Initializes the filter.
142      */
143     @Override
144     public void init(FilterConfig filterConfig) {
145         this.filterConfig = filterConfig;
146         if (filterConfig != null) {
147 
148         }
149     }
150 
151     /**
152      * Returns a string representation of the object.
153      */
154     @Override
155     public String toString() {
156         if (filterConfig == null) {
157             return ("SuperUserFilter()");
158         }
159         StringBuilder sb = new StringBuilder("SuperUserFilter(");
160         sb.append(filterConfig);
161         sb.append(")");
162         return (sb.toString());
163     }
164 
165     private void sendProcessingError(Throwable t, ServletResponse response) {
166       LOGGER.error("Error in superuserpage filtering", t);
```

```
167
168         try {
169             ((HttpServletResponse)response).sendRedirect(RedirectURLs.ERROR_PAGE_URI)
                    ;
170         } catch (IOException ex) {
171             LOGGER.error("Error in redirecting", ex);
172         }
173     }
174     public void log(String msg) {
175         filterConfig.getServletContext().log(msg);
176     }
177
178 }
```

1.26 groupkey/GroupKeyBean.java

```
1
2  import com.moveatis.abstracts.AbstractBean;
3  import com.moveatis.interfaces.GroupKey;
4  import javax.ejb.Stateless;
5  import javax.persistence.EntityManager;
6  import javax.persistence.NoResultException;
7  import javax.persistence.PersistenceContext;
8  import javax.persistence.TypedQuery;
9
10 /**
11  * The EJB manages group keys that are used to access the event groups in a "semi-
        public" fashion.
12  * @author Sami Kallio <phinaliumz at outlook.com>
13  */
14 @Stateless
15 public class GroupKeyBean extends AbstractBean<GroupKeyEntity> implements GroupKey
        {
16
17     @PersistenceContext(unitName = "MOVEATIS_PERSISTENCE")
18     private EntityManager em;
19
20     private GroupKeyEntity groupKeyEntity;
21
22     public GroupKeyBean() {
23         super(GroupKeyEntity.class);
24     }
25
26     @Override
27     protected EntityManager getEntityManager() {
28         return em;
29     }
30
31     /**
32      * The method returns the group key, which currently is associated with the
            instance.
33      * @return The group key associated with the instance for groupkeyEJB.
34      */
35     @Override
36     public GroupKeyEntity getGroupKeyEntity() {
```

```
37          return groupKeyEntity;
38       }
39       /**
40        * The method finds and returns the group key, which has the same value
41        * as the specified parameter.
42        * @param key String-representation of the key.
43        * @return the group key or null.
44        */
45       @Override
46       public GroupKeyEntity findByKey(String key) {
47          TypedQuery<GroupKeyEntity> query = em.createNamedQuery("findKey",
               GroupKeyEntity.class);
48          query.setParameter("key", key);
49
50          try {
51             groupKeyEntity = query.getSingleResult();
52          } catch(NoResultException nre) {
53             return null;
54          }
55
56          return groupKeyEntity;
57
58       }
59
60       /**
61        * The method removes the group keys permanently. Usually the entities are not
              removed
62        * from the database, as only their removal date is set. If the group key need
              to be reused,
63        * they need to be removed permanently before reuse.
64        * @param groupKeyEntity The group key to be removed.
65        */
66       @Override
67       public void removePermanently(GroupKeyEntity groupKeyEntity) {
68          em.remove(em.merge(groupKeyEntity));
69       }
70    }
```

1.27 groupkey/GroupKeyEntity.java

```
1  package com.moveatis.groupkey;
2
3  import com.moveatis.abstracts.BaseEntity;
4  import com.moveatis.event.EventGroupEntity;
5  import com.moveatis.user.IdentifiedUserEntity;
6  import com.moveatis.user.TagUserEntity;
7  import java.io.Serializable;
8  import static javax.persistence.CascadeType.MERGE;
9  import static javax.persistence.CascadeType.PERSIST;
10 import static javax.persistence.CascadeType.REMOVE;
11 import javax.persistence.Column;
12 import javax.persistence.Entity;
13 import javax.persistence.ManyToOne;
14 import javax.persistence.NamedQueries;
15 import javax.persistence.NamedQuery;
```

```
16 import javax.persistence.OneToOne;
17 import javax.persistence.Table;
18 import javax.validation.constraints.NotNull;
19
20 /**
21  * The entity presents the group key that the event groups can be identified with
        in the database.
22  * @author Sami Kallio <phinaliumz at outlook.com>
23  */
24 @Entity
25 @NamedQueries({
26       @NamedQuery(
27          name="findKey",
28          query="SELECT groupkey FROM GroupKeyEntity groupkey WHERE groupkey.
               groupKey=:key"
29       )
30 })
31 @Table(name="GROUPKEY")
32 public class GroupKeyEntity extends BaseEntity implements Serializable {
33
34    private static final long serialVersionUID = 1L;
35
36    @OneToOne
37    private EventGroupEntity eventGroup;
38
39    @OneToOne(cascade={PERSIST, MERGE, REMOVE})
40    private TagUserEntity tagUser;
41
42    @Column(unique=true)
43    @NotNull
44    private String groupKey;
45
46    @ManyToOne
47    private IdentifiedUserEntity creator;
48
49    private String label;
50
51
52    public EventGroupEntity getEventGroup() {
53       return eventGroup;
54    }
55
56    public void setEventGroup(EventGroupEntity eventGroup) {
57       this.eventGroup = eventGroup;
58    }
59
60    public String getGroupKey() {
61       return groupKey;
62    }
63
64    public void setGroupKey(String groupKey) {
65       this.groupKey = groupKey;
66    }
67
68    public IdentifiedUserEntity getCreator() {
69       return creator;
70    }
71
```

```
72     public void setCreator(IdentifiedUserEntity creator) {
73        this.creator = creator;
74     }
75
76     public String getLabel() {
77        return label;
78     }
79
80     public void setLabel(String label) {
81        this.label = label;
82     }
83
84     public TagUserEntity getTagUser() {
85        return tagUser;
86     }
87
88     public void setTagUser(TagUserEntity tagUser) {
89        this.tagUser = tagUser;
90     }
91
92     @Override
93     public int hashCode() {
94        int hash = 0;
95        hash += (id != null ? id.hashCode() : 0);
96        return hash;
97     }
98
99     @Override
100    public boolean equals(Object object) {
101       if (!(object instanceof GroupKeyEntity)) {
102          return false;
103       }
104       GroupKeyEntity other = (GroupKeyEntity) object;
105       return !((this.id == null && other.id != null) || (this.id != null && !this.
              id.equals(other.id)));
106    }
107
108    @Override
109    public String toString() {
110       return "com.moveatis.groupkey.GroupKeyEntity[ id=" + id + " ]";
111    }
112
113 }
```

1.28 helpers/GroupKeyGenerator.java

```
1  package com.moveatis.helpers;
2
3  import java.util.Random;
4
5  /**
6   * The helper class could be used to generate a random group key, if the user
       cannot
7   * come up with one by herself.
8   * @author Sami Kallio <phinaliumz at outlook.com>
```

```
 */
public class GroupKeyGenerator {

   private static final int EVENGROUP_KEY_LENGTH = 8;
   private static final int LETTERS_COUNT = 25;
   private static final int NUMBERS_COUNT = 10;

   private static final int CHOOSE_LETTER = 0;

   private static final char[] LETTERS = {'A','B','C','D','E','F','G','H',
         'I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y'};
   private static final int[] NUMBERS = {0,1,2,3,4,5,6,7,8,9};

   public GroupKeyGenerator() {

   }

   public static String getGroupKey() {
      Random randomizer = new Random();
      StringBuilder stb = new StringBuilder();
      int letterOrNumber;

      for(int i = 0; i <= EVENGROUP_KEY_LENGTH; i++) {
         letterOrNumber = randomizer.nextInt(2);
         if(letterOrNumber % 2 == CHOOSE_LETTER) {
            stb.append(LETTERS[randomizer.nextInt(LETTERS_COUNT)]);
         } else {
            stb.append(NUMBERS[randomizer.nextInt(NUMBERS_COUNT)]);
         }
      }

      return stb.toString();
   }

}
```

1.29 helpers/Validation.java

```
package com.moveatis.helpers;

/**
 * The helper class contains the static validation function(s).
 * @author Ilari Paananen
 */
public class Validation {

   /**
    * Validates the given string so that it can be placed as a JavaScript string
    * in an HTML script tag.
    * @param s The string to be validated.
    * @return The validated string that contains only accepted characters.
    */
   public static String validateForJsAndHtml(String s) {
      // TODO: Allow some other chars? Don't allow some of these?
      // Maybe only blacklist characters shown here:
```

```
18        // http://benv.ca/2012/10/02/you-are-probably-misusing-DOM-text-methods/
19        String validChars = " ,.-;:_!?*/+()[]{}|=#";
20
21        StringBuilder sb = new StringBuilder();
22
23        for (int i = 0; i < s.length(); ) {
24           int codePoint = s.codePointAt(i);
25           if (Character.isLetterOrDigit(codePoint) ||
26              (validChars.indexOf(codePoint) >= 0)) {
27              sb.appendCodePoint(codePoint);
28           }
29           i += Character.charCount(codePoint);
30        }
31
32        return sb.toString();
33     }
34  }
```

1.30 identityprovider/IdentityProvider.java

```
1  package com.moveatis.identityprovider;
2
3  import com.moveatis.user.IdentifiedUserEntity;
4
5  /**
6   * The interface for the identity provider service that must be implemented, if
7   * the identity provider service needs to be customized outside Jyväskylä
        University.
8   * @author Sami Kallio <phinaliumz at outlook.com>
9   */
10 public interface IdentityProvider {
11    IdentifiedUserEntity getIdentifiedUserEntity();
12 }
```

1.31 identityprovider/IdentityProviderBean.java

```
1  package com.moveatis.identityprovider;
2
3  import com.moveatis.abstracts.AbstractBean;
4  import javax.ejb.Stateless;
5  import javax.persistence.EntityManager;
6  import javax.persistence.NoResultException;
7  import javax.persistence.PersistenceContext;
8  import javax.persistence.TypedQuery;
9
10 /**
11  * An example EJB that implements the custom identity provider service.
12  * @author Sami Kallio <phinaliumz at outlook.com>
13  */
14 @Stateless
15 public class IdentityProviderBean extends AbstractBean<
       IdentityProviderInformationEntity> {
16
```

```
17     @PersistenceContext(unitName = "MOVEATIS_PERSISTENCE")
18     private EntityManager em;
19
20     public IdentityProviderBean() {
21        super(IdentityProviderInformationEntity.class);
22     }
23
24     @Override
25     protected EntityManager getEntityManager() {
26        return em;
27     }
28
29     public IdentityProviderInformationEntity findIpEntityByUsername(String userName
           ) {
30        TypedQuery<IdentityProviderInformationEntity> query =
31              em.createNamedQuery("findIdentityProviderEntityByUsername",
                     IdentityProviderInformationEntity.class);
32        try {
33           IdentityProviderInformationEntity entity = query.setParameter("username",
                  userName).getSingleResult();
34           return entity;
35        } catch (NoResultException noResult) {
36           return null;
37        }
38     }
39
40  }
```

1.32 identityprovider/IdentityProviderInformationEntity.java

```
1
2  package com.moveatis.identityprovider;
3
4  import com.moveatis.abstracts.BaseEntity;
5  import com.moveatis.user.IdentifiedUserEntity;
6  import java.io.Serializable;
7  import javax.persistence.Entity;
8  import javax.persistence.NamedQueries;
9  import javax.persistence.NamedQuery;
10 import javax.persistence.OneToOne;
11 import javax.persistence.Table;
12
13 /**
14  * The entity represents the data needed to identify an existing user.
15  * The example entity must be rewritten, if a custom identity service is to be used
        .
16  * @author Sami Kallio <phinaliumz at outlook.com>
17  */
18 @Entity
19 @NamedQueries({
20       @NamedQuery(
21          name="findIdentityProviderEntityByUsername",
22          query="SELECT entity FROM IdentityProviderInformationEntity entity WHERE
                "
23                + "entity.username=:username"
```

```
      )
})
@Table(name="IDENTITY_PROVIDER_INFORMATION")
public class IdentityProviderInformationEntity extends BaseEntity implements
    IdentityProvider, Serializable {

   private static final long serialVersionUID = 1L;

   @OneToOne
   private IdentifiedUserEntity userEntity;

   private String username;
   private String affiliation;

   public IdentityProviderInformationEntity() {

   }

   public void setUserEntity(IdentifiedUserEntity userEntity) {
      this.userEntity = userEntity;
   }

   @Override
   public IdentifiedUserEntity getIdentifiedUserEntity() {
      return this.userEntity;
   }

   public String getUsername() {
      return username;
   }

   public void setUsername(String username) {
      this.username = username;
   }

   public String getAffiliation() {
      return affiliation;
   }

   public void setAffiliation(String affiliation) {
      this.affiliation = affiliation;
   }
}
```

1.33 identityprovider/IdentityProviderRegistrationBean.java

```

package com.moveatis.identityprovider;

import com.moveatis.application.RedirectURLs;
import com.moveatis.user.IdentifiedUserEntity;
import java.io.IOException;
import java.io.Serializable;
import javax.enterprise.context.RequestScoped;
import javax.faces.context.FacesContext;
```

```
10 import javax.faces.event.ActionEvent;
11 import javax.inject.Named;
12 import org.slf4j.Logger;
13 import org.slf4j.LoggerFactory;
14
15 /**
16  * An example managed bean for customizing an identity service.
17  * @author Sami Kallio <phinaliumz at outlook.com>
18  */
19 @Named(value="identityProviderRegistrationBean")
20 @RequestScoped
21 public class IdentityProviderRegistrationBean implements IdentityProvider,
       Serializable {
22
23    private static final long serialVersionUID = 1L;
24    private static final Logger LOGGER = LoggerFactory.getLogger(
          IdentityProviderRegistrationBean.class);
25    private IdentifiedUserEntity userEntity;
26    private IdentityProviderInformationEntity identityProviderInformationEntity;
27
28
29    public IdentityProviderRegistrationBean() {
30
31    }
32
33    public void registerSuperUser(ActionEvent actionEvent) {
34
35       try {
36          FacesContext.getCurrentInstance().getExternalContext().redirect(
               RedirectURLs.SHIBBOLETH_REDIRECT_SECURE_URI);
37       } catch (IOException ex) {
38          LOGGER.debug("Error in registration", ex);
39       }
40    }
41
42    @Override
43    public IdentifiedUserEntity getIdentifiedUserEntity() {
44       userEntity = new IdentifiedUserEntity();
45       identityProviderInformationEntity = new IdentityProviderInformationEntity();
46
47       userEntity.setIdentityProviderInformationEntity(
            identityProviderInformationEntity);
48       identityProviderInformationEntity.setUserEntity(userEntity);
49
50       return userEntity;
51    }
52
53 }
```

1.34 interfaces/AnonUser.java

```
1 package com.moveatis.interfaces;
2
3 import com.moveatis.user.AnonUserEntity;
4 import java.util.List;
```

```
5
6  /**
7   * The interface to manage public users within the Moveatis.
8   * @author Sami Kallio <phinaliumz at outlook.com>
9   */
10 public interface AnonUser {
11
12    void create(AnonUserEntity userEntity);
13
14    void edit(AnonUserEntity userEntity);
15
16    void remove(AnonUserEntity userEntity);
17
18    AnonUserEntity find();
19
20    List<AnonUserEntity> findAll();
21
22 }
```

1.35 interfaces/Application.java

```
1  package com.moveatis.interfaces;
2
3  import com.moveatis.application.ApplicationEntity;
4  import javax.ejb.Local;
5
6  /**
7   * The interface to manage the application entity.
8   *
9   * @author Sami Kallio <phinaliumz at outlook.com>
10  */
11 @Local(Application.class)
12 public interface Application {
13
14    void create(ApplicationEntity applicationEntity);
15
16    void edit(ApplicationEntity applicationEntity);
17
18    void remove(ApplicationEntity applicationEntity);
19
20    ApplicationEntity getApplicationEntity();
21
22    int count();
23
24    boolean checkInstalled();
25
26 }
```

1.36 interfaces/Category.java

```
1  package com.moveatis.interfaces;
2
3  import com.moveatis.category.CategoryEntity;
```

```
4  import com.moveatis.category.CategorySetEntity;
5  import java.util.List;
6  import javax.ejb.Local;
7
8  /**
9   * The interface to manage the category entity.
10  * The category specifies the type or class of an observation record.
11  * For example, a teacher giving feedback to a student could be categorized
12  * as "Giving feedback".
13  *
14  * @author Sami Kallio <phinaliumz at outlook.com>
15  */
16 @Local(Category.class)
17 public interface Category {
18
19    void create(CategoryEntity categoryEntity);
20
21    void edit(CategoryEntity categoryEntity);
22
23    void remove(CategoryEntity categoryEntity);
24
25    void removeFromCategorySet(CategorySetEntity whichCategorySet, CategoryEntity
          whichCategory);
26
27    CategoryEntity find(Object id);
28
29    CategoryEntity findByLabel(String label);
30
31    List<CategoryEntity> findAll();
32
33    List<CategoryEntity> findRange(int[] range);
34
35    int count();
36
37 }
```

1.37 interfaces/CategorySet.java

```
1  package com.moveatis.interfaces;
2
3  import com.moveatis.category.CategoryEntity;
4  import com.moveatis.category.CategorySetEntity;
5  import java.util.List;
6  import java.util.Set;
7  import javax.ejb.Local;
8
9  /**
10  *
11  * The interface to manage the category set entity.
12  * The category set contains different categories that can be thought as a group.
13  * For example, a category set named "Teacher's activities" could contain
        categories
14  * like "Organizing", "Observing", and "Giving feedback".
15  *
16  * @author Sami Kallio <phinaliumz at outlook.com>
```

```
 */
@Local
public interface CategorySet {

   void create(CategorySetEntity categoryTemplateEntity);

   void edit(CategorySetEntity categoryTemplateEntity);

   void remove(CategorySetEntity categoryTemplateEntity);

   void removeCategoryFromCategorySet(CategorySetEntity categorySet,
      CategoryEntity categoryEntity);

   CategorySetEntity find(Object id);

   List<CategorySetEntity> findAll();

   List<CategorySetEntity> findRange(int[] range);

   Set<CategorySetEntity> findPublicCategorySets();

   int count();

}
```

1.38 interfaces/Event.java

```
package com.moveatis.interfaces;

import com.moveatis.event.EventEntity;
import com.moveatis.observation.ObservationEntity;
import com.moveatis.user.IdentifiedUserEntity;
import java.util.List;
import javax.ejb.Local;

/**
 * The interface to manage the event entity.
 * Event specifies the situation that is observed.
 * For example, the event could be "Teaching situation number 4".
 * @author Sami Kallio <phinaliumz at outlook.com>
 */
@Local(Event.class)
public interface Event {

   void create(EventEntity eventEntity);

   void edit(EventEntity eventEntity);

   void remove(EventEntity eventEntity);

   EventEntity find(Object id);

   List<EventEntity> findAll();

   List<EventEntity> findRange(int[] range);
```

```
29
30      void removeObservation(ObservationEntity observationEntity);
31
32      List<EventEntity> findEventsForUser(IdentifiedUserEntity user);
33
34      EventEntity getEventEntity();
35
36      int count();
37
38  }
```

1.39 interfaces/EventGroup.java

```
1   package com.moveatis.interfaces;
2
3   import com.moveatis.category.CategorySetEntity;
4   import com.moveatis.event.EventGroupEntity;
5   import com.moveatis.user.AbstractUser;
6   import java.util.List;
7   import javax.ejb.Local;
8
9   /**
10   * The interface to manage the event group entity.
11   * The event group contains different events that can be thought as a group.
12   * For example, an event group named "Exploratory teacher" could contain events
13   * like "Teaching situation number 1", "Teaching situation number 2", etc.
14   * @author Sami Kallio <phinaliumz at outlook.com>
15   */
16  @Local(EventGroup.class)
17  public interface EventGroup {
18
19      void create(EventGroupEntity eventGroupEntity);
20
21      void edit(EventGroupEntity eventGroupEntity);
22
23      void remove(EventGroupEntity eventGroupEntity);
24
25      EventGroupEntity find(Object id);
26
27      List<EventGroupEntity> findAll();
28
29      List<EventGroupEntity> findAllForOwner(AbstractUser owner);
30
31      List<EventGroupEntity> findAllForUser(AbstractUser user);
32
33      List<EventGroupEntity> findAllForPublicUser();
34
35      List<EventGroupEntity> findRange(int[] range);
36
37      void removeCategorySetEntityFromEventGroups(CategorySetEntity categorySetEntity
            );
38
39      int count();
40
41  }
```

1.40 interfaces/GroupKey.java

```
1
2  import com.moveatis.groupkey.GroupKeyEntity;
3  import java.util.List;
4  import javax.ejb.Local;
5
6  /**
7   * The interface to manage the group key entity.
8   * A group key is used to identify an event group for "semi-public" usage.
9   * Anyone who knowns the group key for a particular event group, can use the event
        group.
10  * @author Sami Kallio <phinaliumz at outlook.com>
11  */
12 @Local(GroupKey.class)
13 public interface GroupKey {
14
15    void create(GroupKeyEntity groupKeyEntity);
16
17    void edit(GroupKeyEntity groupKeyEntity);
18
19    void remove(GroupKeyEntity groupKeyEntity);
20
21    void removePermanently(GroupKeyEntity groupKeyEntity);
22
23    GroupKeyEntity find(Object id);
24
25    GroupKeyEntity findByKey(String key);
26
27    List<GroupKeyEntity> findAll();
28
29    List<GroupKeyEntity> findRange(int[] range);
30
31    GroupKeyEntity getGroupKeyEntity();
32
33    int count();
34 }
```

1.41 interfaces/Label.java

```
1
2  import com.moveatis.label.LabelEntity;
3  import java.util.List;
4  import javax.ejb.Local;
5
6  /**
7   * The interface to manage the label entity.
8   * A label is a name for a category. If there are several similarly named
        categories,
9   * like "Giving feedback", they can all use the same label.
10  *
11  * @author Sami Kallio <phinaliumz at outlook.com>
12  */
13 @Local(Label.class)
14 public interface Label {
```

```
15
16      void create(LabelEntity labelEntity);
17
18      void edit(LabelEntity labelEntity);
19
20      void remove(LabelEntity labelEntity);
21
22      LabelEntity find(Object id);
23
24      LabelEntity findByLabel(String label);
25
26      List<LabelEntity> findAll();
27
28      List<LabelEntity> findRange(int[] range);
29
30      int count();
31  }
```

1.42 interfaces/Mailer.java

```
1   package com.moveatis.interfaces;
2
3   import com.moveatis.enums.MailStatus;
4   import java.io.File;
5   import javax.ejb.Local;
6
7   /**
8    * The interface to send mails.
9    * @author Sami Kallio <phinaliumz at outlook.com>
10   */
11  @Local
12  public interface Mailer {
13
14      MailStatus sendEmail(final String[] recipients, final String subject, final
            String message);
15      MailStatus sendEmailWithAttachment(final String[] recipient, final String
            subject, final String message, final File[] attachmentFile);
16
17  }
```

1.43 interfaces/MessageBundle.java

```
1   package com.moveatis.interfaces;
2
3   import java.lang.annotation.Documented;
4   import static java.lang.annotation.ElementType.FIELD;
5   import static java.lang.annotation.ElementType.METHOD;
6   import static java.lang.annotation.ElementType.PARAMETER;
7   import static java.lang.annotation.ElementType.TYPE;
8   import java.lang.annotation.Retention;
9   import static java.lang.annotation.RetentionPolicy.RUNTIME;
10  import java.lang.annotation.Target;
11  import javax.inject.Qualifier;
```

```
12
13 /**
14  * The qualifier for the message bundle is needed to provide a resource bundle
15  * as injected resource for transient managed beans.
16  * @author Sami Kallio <phinaliumz at outlook.com>
17  */
18 @Qualifier
19 @Target({TYPE,METHOD,PARAMETER,FIELD })
20 @Retention(RUNTIME)
21 @Documented
22 public @interface MessageBundle {
23
24 }
```

1.44 interfaces/Observation.java

```
1  package com.moveatis.interfaces;
2
3  import com.moveatis.observation.ObservationEntity;
4  import com.moveatis.records.RecordEntity;
5  import com.moveatis.user.AbstractUser;
6  import java.util.List;
7  import javax.ejb.Local;
8
9  /**
10  * The interface to manage the observation entity.
11  * The observation contains the records a user made when
12  * he or she was observing some event.
13  *
14  * @author Sami Kallio <phinaliumz at outlook.com>
15  */
16 @Local(Observation.class)
17 public interface Observation {
18
19    void create(ObservationEntity observationEntity);
20
21    void edit(ObservationEntity observationEntity);
22
23    void remove(ObservationEntity observationEntity);
24
25    void removeUnsavedObservation(ObservationEntity observationEntity);
26
27    ObservationEntity find(Object id);
28
29    List<ObservationEntity> findAll();
30
31    List<ObservationEntity> findAllByObserver(AbstractUser observer);
32
33    List<ObservationEntity> findWithoutEvent(AbstractUser observer);
34
35    List<ObservationEntity> findByEventsNotOwned(AbstractUser observer);
36
37    List<ObservationEntity> findRange(int[] range);
38
39    List<RecordEntity> findRecords(Object id);
```

```
40
41      int count();
42  }
```

1.45 interfaces/Record.java

```
1  package com.moveatis.interfaces;
2
3  import com.moveatis.records.RecordEntity;
4  import java.util.List;
5  import javax.ejb.Local;
6
7  /**
8   * The interface to manage the record entity. A record is a single instance of
9   * a category in an observation. For example, if during an observation a teacher
10  * gives feedback for 5 minutes, there would be a record of the category
11  * "Giving feedback" with a duration of 5 minutes.
12  * @author Sami Kallio <phinaliumz at outlook.com>
13  */
14 @Local(Record.class)
15 public interface Record {
16
17     void create(RecordEntity recordEntity);
18
19     void edit(RecordEntity recordEntity);
20
21     void remove(RecordEntity recordEntity);
22
23     RecordEntity find(Object id);
24
25     List<RecordEntity> findAll();
26
27     List<RecordEntity> findRange(int[] range);
28
29     int count();
30
31 }
```

1.46 interfaces/Role.java

```
1  package com.moveatis.interfaces;
2
3  import com.moveatis.roles.SuperUserRoleEntity;
4  import com.moveatis.user.IdentifiedUserEntity;
5  import java.util.Date;
6  import java.util.List;
7
8  /**
9   * The interface to manage the role entity.
10  * A role grants more finegrained access rights to the users.
11  * For now, the only role is the superuser role.
12  * @author Sami Kallio <phinaliumz at outlook.com>
13  */
```

```
14 public interface Role {
15
16    void addSuperuserRoleToUser(IdentifiedUserEntity user);
17    void addSuperuserRoleToUser(IdentifiedUserEntity user, Date startDate, Date
          endDate);
18    void removeSuperuserRoleFromUser(IdentifiedUserEntity user);
19    List<SuperUserRoleEntity> listSuperusers();
20    boolean checkIfUserIsSuperUser(IdentifiedUserEntity user);
21
22 }
```

1.47 interfaces/Session.java

```
1  package com.moveatis.interfaces;
2
3  import com.moveatis.groupkey.GroupKeyEntity;
4  import com.moveatis.observation.ObservationCategorySet;
5  import com.moveatis.user.AbstractUser;
6  import com.moveatis.user.IdentifiedUserEntity;
7  import com.moveatis.user.TagUserEntity;
8  import java.util.List;
9  import java.util.SortedSet;
10 import java.util.TimeZone;
11 import javax.ejb.Local;
12
13 /**
14  * The interface to manage the session entity.
15  * The session is a context, in which a particular user is using Moveatis.
16  * The session has information, that is meaningful in the current context,
17  * such as whether the user has been identified or not.
18  * @author Sami Kallio <phinaliumz at outlook.com>
19  */
20 @Local(Session.class)
21 public interface Session {
22
23     void setTagUser(TagUserEntity tagUser);
24     void setIdentityProviderUser(IdentifiedUserEntity user);
25     void setAnonymityUser();
26     boolean isLoggedIn();
27     boolean isIdentifiedUser();
28     boolean isSaveable();
29     SortedSet<Long> getSessionObservationsIds();
30     void setSessionObservations(SortedSet<Long> observationsIds);
31     AbstractUser getLoggedInUser();
32     IdentifiedUserEntity getLoggedIdentifiedUser();
33     GroupKeyEntity getGroupKey();
34     TimeZone getSessionTimeZone();
35     void setSessionTimeZone(TimeZone timeZone);
36     boolean getIsLocalhost();
37     void setReturnUri(String uri);
38     String getReturnUri();
39     void setCategorySetsInUse(List<ObservationCategorySet> categorySets);
40     List<ObservationCategorySet> getCategorySetsInUse();
41 }
```

1.48 interfaces/TagUser.java

```
1  package com.moveatis.interfaces;
2
3  import com.moveatis.groupkey.GroupKeyEntity;
4  import com.moveatis.user.TagUserEntity;
5  import java.util.List;
6
7  /**
8   * The interface to manage the tag user entity.
9   * A tag user is the user account for group keys, which are used to identify
10  * event groups.
11  * @author Sami Kallio <phinaliumz at outlook.com>
12  */
13 public interface TagUser {
14
15    void create(TagUserEntity userEntity);
16
17    void edit(TagUserEntity userEntity);
18
19    void remove(TagUserEntity userEntity);
20
21    TagUserEntity find(Object id);
22
23    TagUserEntity findByKey(GroupKeyEntity groupkey);
24
25    List<TagUserEntity> findAll();
26
27    List<TagUserEntity> findRange(int[] range);
28
29    int count();
30
31 }
```

1.49 interfaces/User.java

```
1  package com.moveatis.interfaces;
2
3  import com.moveatis.user.IdentifiedUserEntity;
4  import java.util.List;
5  import javax.ejb.Local;
6
7  /**
8   * The interface to manage the user entity.
9   * A user is the identified user, which means that they are recognized by
10  * the identity provider service.
11  * @author Sami Kallio <phinaliumz at outlook.com>
12  */
13 @Local(User.class)
14 public interface User {
15
16    void create(IdentifiedUserEntity userEntity);
17
18    void edit(IdentifiedUserEntity userEntity);
19
```

```
20     void remove(IdentifiedUserEntity userEntity);
21 
22     IdentifiedUserEntity find(Object id);
23 
24     List<IdentifiedUserEntity> findAll();
25 
26     List<IdentifiedUserEntity> findRange(int[] range);
27 
28     int count();
29 
30 }
```

1.50 label/LabelBean.java

```
1 
2  import com.moveatis.abstracts.AbstractBean;
3  import com.moveatis.interfaces.Label;
4  import javax.ejb.Stateless;
5  import javax.persistence.EntityManager;
6  import javax.persistence.PersistenceContext;
7  import javax.persistence.TypedQuery;
8 
9  /**
10  * The EJB manages the labels.
11  * @author Sami Kallio <phinaliumz at outlook.com>
12  */
13 @Stateless
14 public class LabelBean extends AbstractBean<LabelEntity> implements Label {
15 
16     @PersistenceContext(unitName = "MOVEATIS_PERSISTENCE")
17     private EntityManager em;
18 
19     private LabelEntity labelEntity;
20 
21     public LabelBean() {
22         super(LabelEntity.class);
23     }
24 
25     @Override
26     protected EntityManager getEntityManager() {
27         return em;
28     }
29 
30     /**
31      * Finds and returns the label entity with the given string as its value.
32      * @param label The string to search for.
33      * @return the LabelEntity or null.
34      */
35     @Override
36     public LabelEntity findByLabel(String label) {
37         TypedQuery<LabelEntity> query = em.createNamedQuery("findByText",
               LabelEntity.class);
38         query.setParameter("text", label);
39         if(query.getResultList().size() > 0) {
40             return query.getSingleResult();
```

```
41        } else {
42           return null;
43        }
44     }
45  }
```

1.51 label/LabelEntity.java

```
1  package com.moveatis.label;
2
3  import com.moveatis.abstracts.BaseEntity;
4  import com.moveatis.category.CategoryEntity;
5  import java.io.Serializable;
6  import java.util.List;
7  import javax.persistence.Entity;
8  import javax.persistence.FetchType;
9  import javax.persistence.NamedQueries;
10 import javax.persistence.NamedQuery;
11 import javax.persistence.OneToMany;
12 import javax.persistence.Table;
13 import javax.validation.constraints.NotNull;
14
15 /**
16  * The entity represents the information for the category's label in the database.
17  * If there are several categories with a similar name, like "Teaching", they
18  * can all have the same label entity.
19  * @author Sami Kallio <phinaliumz at outlook.com>
20  */
21 @Table(name="LABEL")
22 @Entity
23 @NamedQueries(
24       @NamedQuery(name="findByText", query="SELECT l FROM LabelEntity l WHERE l.
          text = :text")
25 )
26 public class LabelEntity extends BaseEntity implements Serializable {
27
28    private static final long serialVersionUID = 1L;
29
30    @OneToMany(mappedBy = "label", orphanRemoval = true, fetch = FetchType.LAZY)
31    private List<CategoryEntity> categoryEntities;
32
33    @NotNull
34    private String text;
35
36    @Override
37    public int hashCode() {
38       int hash = 0;
39       hash += (id != null ? id.hashCode() : 0);
40       return hash;
41    }
42
43    public String getText() {
44       return text;
45    }
46
```

```
47      public void setText(String text) {
48          this.text = text;
49      }
50
51      public List<CategoryEntity> getCategoryEntities() {
52          return categoryEntities;
53      }
54
55      public void setCategoryEntities(List<CategoryEntity> categoryEntities) {
56          this.categoryEntities = categoryEntities;
57      }
58
59      @Override
60      public boolean equals(Object object) {
61          if (!(object instanceof LabelEntity)) {
62              return false;
63          }
64          LabelEntity other = (LabelEntity) object;
65          return !((this.id == null && other.id != null) || (this.id != null && !this.
                id.equals(other.id)));
66      }
67
68      @Override
69      public String toString() {
70          return "com.moveatis.category.LabelEntity[ id=" + id + " ]";
71      }
72
73  }
```

1.52 mail/MailerBean.java

```
1   package com.moveatis.mail;
2
3   import com.moveatis.application.RedirectURLs;
4   import com.moveatis.enums.MailStatus;
5   import javax.ejb.Stateless;
6   import com.moveatis.interfaces.Mailer;
7   import java.io.File;
8   import java.util.Properties;
9   import javax.activation.DataHandler;
10  import javax.activation.DataSource;
11  import javax.activation.FileDataSource;
12  import javax.mail.Message;
13  import javax.mail.MessagingException;
14  import javax.mail.Multipart;
15  import javax.mail.Session;
16  import javax.mail.Transport;
17  import javax.mail.internet.MimeBodyPart;
18  import javax.mail.internet.MimeMessage;
19  import javax.mail.internet.MimeMultipart;
20  import org.slf4j.Logger;
21  import org.slf4j.LoggerFactory;
22
23  /**
24   * The class implements the Mailer interface, and takes care of mailing to the
        users
```

```
25  * the requested information.
26  *
27  * @author Sami Kallio <phinaliumz at outlook.com>
28  */
29 @Stateless
30 public class MailerBean implements Mailer {
31
32    private static final Logger LOGGER = LoggerFactory.getLogger(MailerBean.class);
33
34    private static final String FROM = "donotreply@moveatis.sport.jyu.fi";
35
36    private static final String CHARSET = "UTF-8";
37
38
39    public MailerBean() {
40
41    }
42
43    /**
44     * Sends email to the recipients.
45     *
46     * @param recipients An array of the users to whom the mail is sent.
47     * @param subject The subject of the mail.
48     * @param message The message for the mail.
49     * @return enum that states if the mail was sent successfully or not.
50     */
51    @Override
52    public MailStatus sendEmail(final String[] recipients, final String subject,
          final String message) {
53
54       try {
55          MimeMessage msg = setMessage(recipients, subject);
56          MimeBodyPart textPart = new MimeBodyPart();
57          textPart.setText(message, CHARSET);
58
59          MimeMultipart multipart = new MimeMultipart();
60          multipart.addBodyPart(textPart);
61
62          msg.setContent(multipart);
63          Transport.send(msg);
64
65          return MailStatus.MAIL_SENT_OK;
66       } catch (MessagingException ex) {
67          LOGGER.error("Error in email", ex);
68       }
69
70       return MailStatus.MAIL_SENT_FAILED;
71    }
72
73    /**
74     * Sends email with the given attachment files.
75     *
76     * @param recipients An array of the users to whom the mail is sent.
77     * @param subject The subject of the mail.
78     * @param message The message for the mail.
79     * @param attachmentFiles An array of the files to be attached to the mail.
80     * @return enum that states if the mail was sent successfully or not.
81     */
```

```
82      @Override
83      public MailStatus sendEmailWithAttachment(final String[] recipients, final
            String subject, final String message, final File[] attachmentFiles) {
84
85          try {
86              MimeMessage msg = setMessage(recipients,subject);
87              MimeBodyPart textPart = new MimeBodyPart();
88              textPart.setText(message, CHARSET);
89
90              Multipart multipart = new MimeMultipart();
91              multipart.addBodyPart(textPart);
92
93              for(File f:attachmentFiles) {
94                  DataSource source = new FileDataSource(f);
95                  MimeBodyPart attachPart = new MimeBodyPart();
96                  attachPart.setDataHandler(new DataHandler(source));
97                  attachPart.setFileName(f.getName());
98
99                  multipart.addBodyPart(attachPart);
100             }
101
102             msg.setContent(multipart);
103             Transport.send(msg);
104
105             return MailStatus.MAIL_SENT_OK;
106
107         } catch(MessagingException ex) {
108             LOGGER.error("Error in email", ex);
109         }
110
111         return MailStatus.MAIL_SENT_FAILED;
112     }
113
114     /**
115      * Creates the mime message, which is sent to the recipients.
116      *
117      * @param recipients An array of the users to whom the mail is sent.
118      * @param subject The subject of the mail.
119      * @param message The message for the mail.
120      * @return The MimeMessage containing the necessary information for sending the
             email.
121      * @throws MessagingException If there is an error in sending the email.
122      */
123     private MimeMessage setMessage(final String[] recipients, final String subject)
            throws MessagingException {
124         Properties props = System.getProperties();
125         props.setProperty("mail.smtp.host", RedirectURLs.SMTP_HOST);
126         props.setProperty("mail.mime.charset", CHARSET);
127         Session session = Session.getDefaultInstance(props);
128         MimeMessage msg = new MimeMessage(session);
129
130         msg.setFrom(FROM);
131         for(String recipient:recipients) {
132             msg.addRecipients(Message.RecipientType.TO, recipient);
133         }
134
135         msg.setSubject(subject, CHARSET);
136
```

```
137         return msg;
138     }
139 }
```

1.53 managedbeans/ApplicationManagedBean.java

```
1  package com.moveatis.managedbeans;
2
3  import com.moveatis.interfaces.Application;
4  import com.moveatis.interfaces.Session;
5  import javax.faces.bean.ManagedBean;
6  import javax.faces.bean.RequestScoped;
7  import javax.inject.Inject;
8
9  /**
10  * The bean that serves the application installation view.
11  * @author Sami Kallio <phinaliumz at outlook.com>
12  */
13 @ManagedBean(name="applicationManagedBean")
14 @RequestScoped
15 public class ApplicationManagedBean {
16
17     @Inject
18     private Application applicationEJB;
19
20     @Inject
21     private Session sessionBean;
22
23     public ApplicationManagedBean() {
24
25     }
26
27     /**
28      * Returns true if the application has been installed, otherwise false.
29      */
30     public Boolean getInstalled() {
31         return applicationEJB.checkInstalled();
32     }
33
34     /**
35      * Redirects the user to the installation URI if the application has not
36      * been installed yet. If the application is running on localhost server
37      * it redirects to the localhost version of the installation URI.
38      */
39     public String doInstall() {
40         if(applicationEJB.checkInstalled()) {
41             return "index?faces-redirect=true";
42         } else {
43             if(sessionBean.getIsLocalhost()) {
44                 return "jyutesting/index.xhtml?faces-redirect=true";
45             } else {
46                 return "install?faces-redirect=true";
47             }
48         }
49     }
```

```
50 }
```

1.54 managedbeans/CategoryManagedBean.java

```
1  package com.moveatis.managedbeans;
2
3  import com.moveatis.category.CategoryEntity;
4  import com.moveatis.category.CategorySetEntity;
5  import com.moveatis.interfaces.Category;
6  import com.moveatis.interfaces.Label;
7  import com.moveatis.label.LabelEntity;
8  import java.io.Serializable;
9  import java.util.Map;
10 import java.util.TreeMap;
11 import javax.faces.event.ActionEvent;
12 import javax.inject.Named;
13 import javax.faces.view.ViewScoped;
14 import javax.inject.Inject;
15 import org.slf4j.Logger;
16 import org.slf4j.LoggerFactory;
17
18 /**
19  * The bean that serves category management in the appropriate views.
20  * @author Sami Kallio <phinaliumz at outlook.com>
21  */
22 @Named(value = "categoryManagedBean")
23 @ViewScoped
24 public class CategoryManagedBean implements Serializable {
25
26    private static final Logger LOGGER = LoggerFactory.getLogger(
          CategoryManagedBean.class);
27
28    private static final long serialVersionUID = 1L;
29
30    @Inject
31    private Category categoryEJB;
32    @Inject
33    private Label labelEJB;
34    @Inject
35    private ControlManagedBean controlManagedBean;
36
37    private String label;
38    private String description;
39    private Boolean canOverlap = false;
40
41    /**
42     * Creates a new instance of CategoryManagedBean.
43     */
44    public CategoryManagedBean() {
45
46    }
47
48    public String getLabel() {
49       return label;
50    }
```

```
51
52      public void setLabel(String label) {
53         this.label = label;
54      }
55
56      public String getDescription() {
57         return description;
58      }
59
60      public void setDescription(String description) {
61         this.description = description;
62      }
63
64      public Boolean getCanOverlap() {
65         return canOverlap;
66      }
67
68      public void setCanOverlap(Boolean canOverlap) {
69         this.canOverlap = canOverlap;
70      }
71
72      public void addCategory(ActionEvent event) {
73         LOGGER.debug("Category added");
74      }
75
76      /**
77       * Creates a new category entity and adds it to the given category set.
78       */
79      public void createNewCategory(CategorySetEntity categorySetEntity) {
80         CategoryEntity categoryEntity = new CategoryEntity();
81
82         LabelEntity labelEntity = labelEJB.findByLabel(label);
83
84         if (labelEntity == null) {
85            labelEntity = new LabelEntity();
86            labelEntity.setText(label);
87            labelEJB.create(labelEntity);
88         }
89
90         categoryEntity.setLabel(labelEntity);
91         categoryEntity.setCategorySet(categorySetEntity);
92         categoryEntity.setCanOverlap(canOverlap);
93         categoryEntity.setDescription(description);
94
95         Map<Integer, CategoryEntity> categories = categorySetEntity.
               getCategoryEntitys();
96
97         if (categories == null) {
98            categories = new TreeMap<>();
99         }
100
101        Integer orderNumber = categories.size();
102
103        categories.put(orderNumber, categoryEntity);
104        categoryEntity.setOrderNumber(orderNumber);
105        categorySetEntity.setCategoryEntitys(categories);
106
107        categoryEJB.create(categoryEntity);
```

```
108
109         //controlManagedBean.addCategory(categoryEntity);
110     }
111 }
```

1.55 managedbeans/CategorySelectionManagedBean.java

```
1  package com.moveatis.managedbeans;
2
3  import com.moveatis.category.CategoryEntity;
4  import com.moveatis.category.CategorySetEntity;
5  import com.moveatis.event.EventEntity;
6  import com.moveatis.event.EventGroupEntity;
7  import com.moveatis.helpers.Validation;
8  import com.moveatis.interfaces.EventGroup;
9  import javax.inject.Named;
10 import java.io.Serializable;
11 import java.util.List;
12 import java.util.ResourceBundle;
13 import javax.annotation.PostConstruct;
14 import javax.faces.application.FacesMessage;
15 import javax.faces.context.FacesContext;
16 import javax.inject.Inject;
17 import org.slf4j.Logger;
18 import org.slf4j.LoggerFactory;
19 import com.moveatis.interfaces.MessageBundle;
20 import com.moveatis.interfaces.Session;
21 import com.moveatis.observation.ObservationCategory;
22 import com.moveatis.observation.ObservationCategorySet;
23 import com.moveatis.observation.ObservationCategorySetList;
24 import com.moveatis.user.IdentifiedUserEntity;
25 import java.util.HashSet;
26 import java.util.Map;
27 import java.util.Set;
28 import javax.faces.view.ViewScoped;
29
30 /**
31  * The bean that serves the category selection view.
32  * @author Sami Kallio <phinaliumz at outlook.com>
33  * @author Ilari Paananen <ilari.k.paananen at student.jyu.fi>
34  */
35 @Named(value = "categorySelectionBean")
36 @ViewScoped
37 public class CategorySelectionManagedBean implements Serializable {
38
39     private static final Logger LOGGER = LoggerFactory.getLogger(
        CategorySelectionManagedBean.class);
40     private static final long serialVersionUID = 1L;
41
42     private String newCategorySetName;
43
44     private Long selectedDefaultCategorySet;
45     private Long selectedPrivateCategorySet;
46
47     private ObservationCategorySetList defaultCategorySets; // From group key or
        event that was selected in control page.
```

```
48      private ObservationCategorySetList privateCategorySets;
49      private ObservationCategorySetList categorySetsInUse;
50
51      private EventGroupEntity eventGroup;
52
53      @Inject
54      private Session sessionBean;
55      @Inject
56      private EventGroup eventGroupEJB;
57      @Inject
58      private ObservationManagedBean observationManagedBean;
59
60      // TODO: Messages aren't updated to match language selection. Get
            ResourceBundle some other way?
61      @Inject @MessageBundle //created MessageBundle to allow resourcebundle
            injection to CDI beans
62      private transient ResourceBundle messages; //RequestBundle is not serializable
63
64      private Long addedCategorySetTag = 0L;
65
66      /**
67       * Creates a new instance of CategoryManagedBean.
68       */
69      public CategorySelectionManagedBean() {
70      }
71
72      /**
73       * Adds all category sets from given event group to given observation category
             set list.
74       */
75      private void addAllCategorySetsFromEventGroup(ObservationCategorySetList addTo,
             EventGroupEntity eventGroup) {
76
77          Set<CategorySetEntity> categorySets = eventGroup.getCategorySets();
78
79          if (categorySets == null) return;
80
81          for (CategorySetEntity categorySetEntity : categorySets) {
82              ObservationCategorySet categorySet = new ObservationCategorySet(
                    categorySetEntity.getId(), categorySetEntity.getLabel());
83
84              Map<Integer, CategoryEntity> categories = categorySetEntity.
                    getCategoryEntitys();
85              for (CategoryEntity category : categories.values()) {
86                  categorySet.add(category.getCategoryType(), observationManagedBean.
                        getNextTag(), category.getLabel().getText());
87              }
88
89              addTo.add(categorySet);
90          }
91      }
92
93      /**
94       * Adds all category sets from all the given event groups and puts them in a
             category set list.
95       */
96      private void addAllCategorySetsFromEventGroups(ObservationCategorySetList
            categorySets, List<EventGroupEntity> eventGroups) {
```

```
97          for (EventGroupEntity eventGroup_ : eventGroups) {
98             addAllCategorySetsFromEventGroup(categorySets, eventGroup_);
99          }
100      }
101
102      /**
103       * Initializes properly all the members needed for category selection.
104       */
105      @PostConstruct
106      public void init() {
107         eventGroup = null;
108         defaultCategorySets = new ObservationCategorySetList();
109         privateCategorySets = new ObservationCategorySetList();
110         categorySetsInUse = new ObservationCategorySetList();
111
112         if (observationManagedBean.getEventEntity() != null) {
113            EventEntity event = observationManagedBean.getEventEntity();
114            eventGroup = event.getEventGroup();
115            addAllCategorySetsFromEventGroup(defaultCategorySets, eventGroup);
116         }
117
118         if (sessionBean.isIdentifiedUser()) {
119            IdentifiedUserEntity user = sessionBean.getLoggedIdentifiedUser();
120            addAllCategorySetsFromEventGroups(privateCategorySets, eventGroupEJB.
                  findAllForOwner(user));
121         }
122
123         List<ObservationCategorySet> categorySets = sessionBean.getCategorySetsInUse
              ();
124         if (categorySets != null) {
125            categorySetsInUse.setCategorySets(categorySets);
126         } else {
127            for(ObservationCategorySet categorySet : defaultCategorySets.
                  getCategorySets()) {
128               categorySetsInUse.addClone(categorySet);
129            }
130         }
131      }
132
133      /**
134       * Gets the new category set name.
135       */
136      public String getNewCategorySetName() {
137         return newCategorySetName;
138      }
139
140      /**
141       * Sets the new category set name.
142       */
143      public void setNewCategorySetName(String newCategorySetName) {
144         this.newCategorySetName = newCategorySetName;
145      }
146
147      /**
148       * Gets the selected default category set.
149       */
150      public Long getSelectedDefaultCategorySet() {
151         return selectedDefaultCategorySet;
```

```
152     }
153
154     /**
155      * Sets the selected default category set.
156      */
157     public void setSelectedDefaultCategorySet(Long selectedDefaultCategorySet) {
158        this.selectedDefaultCategorySet = selectedDefaultCategorySet;
159     }
160
161     /**
162      * Gets the selected private category set.
163      */
164     public Long getSelectedPrivateCategorySet() {
165        return selectedPrivateCategorySet;
166     }
167
168
169     /**
170      * Sets the selected private category set.
171      */
172     public void setSelectedPrivateCategorySet(Long selectedPrivateCategorySet) {
173        this.selectedPrivateCategorySet = selectedPrivateCategorySet;
174     }
175
176     /**
177      * Gets the default category sets.
178      */
179     public List<ObservationCategorySet> getDefaultCategorySets() {
180        return defaultCategorySets.getCategorySets();
181     }
182
183     /**
184      * Gets the private category sets.
185      */
186     public List<ObservationCategorySet> getPrivateCategorySets() {
187        return privateCategorySets.getCategorySets();
188     }
189
190     /**
191      * Gets the category sets in use.
192      */
193     public List<ObservationCategorySet> getCategorySetsInUse() {
194        return categorySetsInUse.getCategorySets();
195     }
196
197     /**
198      * Gets the event group.
199      */
200     public EventGroupEntity getEventGroup() {
201        return eventGroup;
202     }
203
204     /**
205      * Adds a new category set for the observation if newCategorySetName isn't empty
           .
206      */
207     public void addNewCategorySet() {
208        String name = Validation.validateForJsAndHtml(newCategorySetName);
```

```
209
210         if (!name.isEmpty()) {
211             // TODO: Is this wanted? What about default category sets?
212             // Can they be added if there is already category with same name in use?
213             for (ObservationCategorySet set : categorySetsInUse.getCategorySets()) {
214                 if (name.equals(set.getName())) {
215                     showErrorMessage(messages.getString("cs_errorNotUniqueCategorySet")
                            );
216                     return;
217                 }
218             }
219
220             ObservationCategorySet categorySet = new ObservationCategorySet(
                    addedCategorySetTag++, name);
221             categorySetsInUse.add(categorySet);
222             newCategorySetName = "";
223         }
224     }
225
226     /**
227      * Adds the selected default category set for the observation.
228      */
229     public void addDefaultCategorySet() {
230         if (categorySetsInUse.find(selectedDefaultCategorySet) == null) {
231             ObservationCategorySet categorySet = defaultCategorySets.find(
                    selectedDefaultCategorySet);
232             if (categorySet != null) categorySetsInUse.addClone(categorySet);
233             else LOGGER.debug("Selected default category set not found!");
234         }
235     }
236
237     /**
238      * Adds the selected private category set for the observation.
239      */
240     public void addPrivateCategorySet() {
241         if (categorySetsInUse.find(selectedPrivateCategorySet) == null) {
242             ObservationCategorySet categorySet = privateCategorySets.find(
                    selectedPrivateCategorySet);
243             if (categorySet != null) categorySetsInUse.addClone(categorySet);
244             else LOGGER.debug("Selected private category set not found!");
245         }
246     }
247
248     /**
249      * Removes the given category set from the observation.
250      */
251     public void removeCategorySet(ObservationCategorySet categorySet) {
252         categorySetsInUse.remove(categorySet);
253     }
254
255     /**
256      * Checks if the continue button should be disabled.
257      * The button is disabled if no category sets have been added for the
            observation
258      * or if some of the added category sets are empty.
259      * @return True if the continue button should be disabled.
260      */
261     public boolean isContinueDisabled() {
```

```
262         for (ObservationCategorySet categorySet : categorySetsInUse.getCategorySets
                ()) {
263             if (categorySet.getCategories().isEmpty()) return true;
264         }
265         return categorySetsInUse.getCategorySets().isEmpty();
266     }
267
268     /**
269      * Shows given error message in primefaces message popup.
270      * @param message Error message to show.
271      */
272     private void showErrorMessage(String message) {
273         FacesContext context = FacesContext.getCurrentInstance();
274         context.addMessage(null, new FacesMessage(FacesMessage.SEVERITY_ERROR,
                messages.getString("dialogErrorTitle"), message));
275     }
276
277     /**
278      * Checks the categories in use before letting the user continue to observation.
279      * The categories in the same category set should have different names.
280      * The categories shouldn't have empty names.
281      * At least one category should be selected for the observation.
282      * It shows an error message if the categories aren't ok.
283      * @return "categoriesok" if the categories were ok, otherwise an empty string.
284      */
285     public String checkCategories() {
286         boolean atLeastOneCategorySelected = false;
287
288         for (ObservationCategorySet categorySet : categorySetsInUse.getCategorySets
                ()) {
289
290             List<ObservationCategory> categories = categorySet.getCategories();
291
292             if(hasDuplicate(categories)) {
293                 showErrorMessage(messages.getString("cs_errorNotUniqueCategories"));
294                 return "";
295             }
296
297             if (!categories.isEmpty()) {
298                 atLeastOneCategorySelected = true;
299             } else {
300                 showErrorMessage(messages.getString("cs_warningEmptyCategorySets"));
301                 return ""; // TODO: Show confirmation or something and let user
                        continue.
302             }
303
304             for (ObservationCategory category : categories) {
305
306                 if (category.getName().isEmpty()) {
307                     showErrorMessage(messages.getString("cs_warningEmptyCategories"));
308                     return ""; // TODO: Show confirmation or something and let user
                            continue.
309                 }
310             }
311         }
312
313         if (!atLeastOneCategorySelected) {
314             showErrorMessage(messages.getString("cs_errorNoneSelected"));
```

```
315             return "";
316         }
317 
318         observationManagedBean.setCategorySetsInUse(categorySetsInUse.
                getCategorySets());
319         return "categoriesok";
320     }
321 
322     /**
323      * Checks if given categories contain duplicate names.
324      * @param categories List of categories to check.
325      * @return True if categories contain duplicates, otherwise false.
326      */
327     private static boolean hasDuplicate(List<ObservationCategory> categories) {
328         Set<String> set = new HashSet<>();
329         for (ObservationCategory category : categories) {
330             String name = category.getName();
331             if (!name.isEmpty() && !set.add(category.getName())) {
332                 return true;
333             }
334         }
335         return false;
336     }
337 }
```

1.56 managedbeans/CategorySetManagedBean.java

```
1  package com.moveatis.managedbeans;
2  
3  import com.moveatis.category.CategoryEntity;
4  import com.moveatis.category.CategorySetEntity;
5  import com.moveatis.event.EventGroupEntity;
6  import com.moveatis.interfaces.Category;
7  import com.moveatis.interfaces.CategorySet;
8  import com.moveatis.interfaces.Label;
9  import com.moveatis.interfaces.Session;
10 import com.moveatis.label.LabelEntity;
11 import java.io.Serializable;
12 import java.util.ArrayList;
13 import java.util.List;
14 import java.util.Map;
15 import java.util.Set;
16 import java.util.TreeMap;
17 import java.util.HashSet;
18 import java.util.LinkedList;
19 import javax.inject.Named;
20 import javax.faces.view.ViewScoped;
21 import javax.inject.Inject;
22 import org.slf4j.Logger;
23 import org.slf4j.LoggerFactory;
24 
25 /**
26  * The bean for managing category sets in the appropriate views.
27  * @author Sami Kallio <phinaliumz at outlook.com>
28  */
```

```
@Named(value = "categorySetManagedBean")
@ViewScoped
public class CategorySetManagedBean implements Serializable {

    private static final Logger LOGGER = LoggerFactory.getLogger(
        CategorySetManagedBean.class);

    private final static long serialVersionUID = 1L;

    private String name;
    private String description;

    @Inject
    private ControlManagedBean controlManagedBean;
    @Inject
    private Session sessionBean;
    @Inject
    private CategorySet categorySetEJB;
    @Inject
    private Category categoryEJB;
    @Inject
    private Label labelEJB;

    private CategorySetEntity categorySetEntity;

    /**
     * Creates a new instance of CategorySetManagedBean.
     */
    public CategorySetManagedBean() {

    }

    public String getName() {
        return name;
    }

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

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    /**
     * Creates a new category set and adds it to the given event group.
     * @param eventGroupEntity The event group into which the new category set is
         added.
     */
    public void createNewCategorySet(EventGroupEntity eventGroupEntity) {
        categorySetEntity = new CategorySetEntity();
        categorySetEntity.setCreator(sessionBean.getLoggedIdentifiedUser());
        categorySetEntity.setEventGroupEntity(eventGroupEntity);
        categorySetEntity.setLabel(name);
```

```
85          categorySetEntity.setDescription(description);
86
87          Set<CategorySetEntity> categorySets = eventGroupEntity.getCategorySets();
88
89          if (categorySets == null) {
90             categorySets = new HashSet<>();
91          }
92
93          categorySets.add(categorySetEntity);
94          eventGroupEntity.setCategorySets(categorySets);
95
96          categorySetEJB.create(categorySetEntity);
97       }
98
99       /**
100       * The method is used for creating and editing the category set. As it's fairly
             complex,
101       * see the comments in the code to get better understanding what it does.
102       *
103       * @param eventGroupEntity The event group the category set belongs to.
104       * @param categorySetEntity The category set to be created or edited.
105       * @param newCategoryEntities The categories belonging to the category set.
106       */
107      public void createAndEditCategorySet(EventGroupEntity eventGroupEntity,
108            CategorySetEntity categorySetEntity, List<CategoryEntity>
                 newCategoryEntities) {
109
110         if (categorySetEntity.getId() == null) {
111            categorySetEJB.create(categorySetEntity);
112         }
113
114         Map<Integer, CategoryEntity> orderedCategories = new TreeMap<>();
115         List<CategoryEntity> unorderedCategories = new ArrayList<>();
116
117         // TODO: The logic of the for-loop can be done without this variable.
118         boolean doNotRemove = false;
119
120         /*
121         * Remove those categoryentities which were not part of newCategoryEntity
                list.
122         * Algorithm goes as follows:
123         * 1. add ids of categoryentities in newCategoryEntity list to a list,
124            so the list only has those categoryentities that are to be kept in
                 categoryset.
125         * 2. Compare previous list to list of all categoryentities of categoryset
126         * 3. If categoryentity is not in first list but is in second, it must be
                removed.
127         */
128         if (categorySetEntity.getCategoryEntitys() != null) {
129
130            List<Long> idForNewCategories = new ArrayList<>();
131            List<Integer> keysForCategoriesToBeRemoved = new ArrayList<>();
132
133            // newCategoryEntities containts the categoryentities user wants to keep
                 in categoryset
134            for(CategoryEntity newCategoryEntity : newCategoryEntities) {
135               if(newCategoryEntity.getId() != null) {
136                  idForNewCategories.add(newCategoryEntity.getId());
```

```
137             }
138          }
139
140          // get all categoryentities for categoryset
141          Map<Integer, CategoryEntity> categories = categorySetEntity.
                getCategoryEntitys();
142
143          // check if categoryentity is part of newcategoryentities list.
144          for(Integer key : categories.keySet()) {
145
146             doNotRemove = false;
147
148             CategoryEntity categoryEntity = categories.get(key);
149
150             if(idForNewCategories.contains(categoryEntity.getId())) {
151                doNotRemove = true;
152             }
153
154             // if its not, add it to be removed list.
155             if(!doNotRemove) {
156                keysForCategoriesToBeRemoved.add(key);
157             }
158          }
159
160          // remove those categoryentities which were part of toBeRemoved list.
161          for(Integer key : keysForCategoriesToBeRemoved) {
162             categoryEJB.removeFromCategorySet(categorySetEntity, categorySetEntity
                   .getCategoryEntitys().get(key));
163          }
164       }
165
166       for (CategoryEntity categoryEntity : newCategoryEntities) {
167          String label = categoryEntity.getLabel().getText();
168          LabelEntity labelEntity = labelEJB.findByLabel(label);
169
170          if (labelEntity == null) {
171             labelEntity = new LabelEntity();
172             labelEntity.setText(label);
173             // Create label entity before other categories in the loop
174             // to prevent creating non-unique labels. Is this required?
175             // labelEJB.create(labelEntity); //Sami: should not be required
                   because CategoryEntity has cascade=PERSIST and MERGE
176          }
177
178          categoryEntity.setLabel(labelEntity);
179          List<CategoryEntity> labelCategories = labelEntity.getCategoryEntities();
180
181          if (labelCategories == null) {
182             labelCategories = new LinkedList<>();
183          }
184          labelCategories.add(categoryEntity);
185          labelEntity.setCategoryEntities(labelCategories);
186
187          List<CategoryEntity> labelCategoryList = labelEntity.getCategoryEntities
                ();
188          if (labelCategoryList == null) {
189             labelCategoryList = new ArrayList<>();
190          }
```

```
191             labelCategoryList.add(categoryEntity);
192             labelEntity.setCategoryEntities(labelCategoryList);
193 
194             categoryEntity.setCategorySet(categorySetEntity);
195 
196             if (categoryEntity.getOrderNumber() == null) {
197                 unorderedCategories.add(categoryEntity);
198             } else {
199                 orderedCategories.put(categoryEntity.getOrderNumber(), categoryEntity)
                        ;
200             }
201         }
202 
203         for (CategoryEntity categoryEntity : unorderedCategories) {
204             categoryEntity.setOrderNumber(orderedCategories.size());
205             orderedCategories.put(orderedCategories.size(), categoryEntity);
206         }
207 
208         categorySetEntity.setCategoryEntitys(orderedCategories);
209         categorySetEntity.setCreator(sessionBean.getLoggedIdentifiedUser());
210         categorySetEntity.setEventGroupEntity(eventGroupEntity);
211 
212         Set<CategorySetEntity> categorySets = eventGroupEntity.getCategorySets();
213 
214         if (categorySets == null) {
215             categorySets = new HashSet<>();
216         }
217 
218         categorySets.add(categorySetEntity);
219         eventGroupEntity.setCategorySets(categorySets);
220 
221         categorySetEJB.edit(categorySetEntity);
222     }
223 }
```

1.57 managedbeans/ControlManagedBean.java

```
1  package com.moveatis.managedbeans;
2 
3  import com.moveatis.category.CategoryEntity;
4  import com.moveatis.category.CategorySetEntity;
5  import com.moveatis.category.CategoryType;
6  import com.moveatis.event.EventEntity;
7  import com.moveatis.event.EventGroupEntity;
8  import com.moveatis.interfaces.Category;
9  import com.moveatis.interfaces.CategorySet;
10 import com.moveatis.interfaces.Event;
11 import com.moveatis.interfaces.EventGroup;
12 import com.moveatis.interfaces.MessageBundle;
13 import com.moveatis.interfaces.Observation;
14 import com.moveatis.interfaces.Session;
15 import com.moveatis.label.LabelEntity;
16 import com.moveatis.observation.ObservationEntity;
17 import com.moveatis.user.AbstractUser;
18 import com.moveatis.user.IdentifiedUserEntity;
```

```
19 import java.io.Serializable;
20 import java.util.ArrayList;
21 import java.util.Collections;
22 import java.util.Comparator;
23 import java.util.HashSet;
24 import java.util.List;
25 import java.util.ResourceBundle;
26 import java.util.Set;
27 import java.util.TreeSet;
28 import java.util.concurrent.TimeUnit;
29 import javax.annotation.PostConstruct;
30 import javax.faces.application.FacesMessage;
31 import javax.faces.context.FacesContext;
32 import javax.faces.view.ViewScoped;
33 import javax.inject.Named;
34 import javax.inject.Inject;
35 import org.primefaces.event.ReorderEvent;
36 import org.primefaces.event.RowEditEvent;
37
38 /**
39  * The bean class for managing the control page view.
40  *
41  * @author Sami Kallio <phinaliumz at outlook.com>
42  * @author Juha Moisio <juha.pa.moisio at student.jyu.fi>
43  */
44 @Named(value = "controlBean")
45 @ViewScoped
46 public class ControlManagedBean implements Serializable {
47
48     private static final long serialVersionUID = 1L;
49
50     //private static final Logger LOGGER = LoggerFactory.getLogger(
        ControlManagedBean.class);
51     private List<EventGroupEntity> eventGroups;
52     private List<CategoryEntity> categories;
53     private List<ObservationEntity> otherObservations;
54
55     private EventGroupEntity selectedEventGroup;
56     private CategorySetEntity selectedCategorySet;
57     private CategoryEntity selectedCategory;
58     private ObservationEntity selectedObservation;
59
60     private boolean creatingNewEventGroup = false;
61
62     @Inject
63     private EventGroup eventGroupEJB;
64     @Inject
65     private Event eventEJB;
66     @Inject
67     private CategorySet categorySetEJB;
68     @Inject
69     private Category categoryEJB;
70     @Inject
71     private Observation observationEJB;
72     @Inject
73     private CategorySetManagedBean categorySetBean;
74     @Inject
75     private EventGroupManagedBean eventGroupBean;
```

```
76
77      @Inject
78      private Session sessionBean;
79      @Inject
80      private ObservationManagedBean observationBean;
81
82      @Inject
83      @MessageBundle
84      private transient ResourceBundle messages;
85
86      private AbstractUser user;
87
88      /**
89       * Initializes the bean appropriately.
90       */
91      @PostConstruct
92      public void init() {
93         user = sessionBean.getLoggedIdentifiedUser();
94         fetchEventGroups();
95         fetchOtherObservations();
96      }
97
98      /**
99       * Fetches the event groups of the user.
100      */
101     protected void fetchEventGroups() {
102        eventGroups = eventGroupEJB.findAllForOwner(user);
103        // Re order event groups by created date
104        Collections.sort(eventGroups, new Comparator<EventGroupEntity>() {
105           @Override
106           public int compare(EventGroupEntity one, EventGroupEntity other) {
107              return one.getCreated().compareTo(other.getCreated());
108           }
109        });
110     }
111
112     /**
113      * Fetches the other observations of the user.
114      */
115     private void fetchOtherObservations() {
116        otherObservations = observationEJB.findWithoutEvent(user);
117        otherObservations.addAll(observationEJB.findByEventsNotOwned(user));
118     }
119
120     /**
121      * Gets the observations of the event group.
122      */
123     public Set<ObservationEntity> getObservations(EventGroupEntity eventGroup) {
124        if (eventGroup != null && eventGroup.getEvent() != null) {
125           return eventGroup.getEvent().getObservations();
126        }
127        return new TreeSet<>();
128     }
129
130     /**
131      * Returns true if a new event group is being created.
132      */
133     public boolean isCreatingNewEventGroup() {
```

```
134         return creatingNewEventGroup;
135     }
136 
137     /**
138      * Sets whether a new event group is being created or not.
139      */
140     public void setCreatingNewEventGroup(boolean creatingNewEventGroup) {
141         this.creatingNewEventGroup = creatingNewEventGroup;
142     }
143 
144     /**
145      * Checks if the event group entity has a group key.
146      */
147     public boolean hasGroupKey(EventGroupEntity eventGroup) {
148         return eventGroup != null && eventGroup.getGroupKey() != null;
149     }
150 
151     /**
152      * Adds a new category set in the view.
153      */
154     public void addNewCategorySet() {
155         selectedCategorySet = new CategorySetEntity();
156         categories = new ArrayList<>();
157     }
158 
159     /**
160      * Adds a new category in the view.
161      */
162     public void addNewCategory() {
163         CategoryEntity category = new CategoryEntity();
164         LabelEntity label = new LabelEntity();
165         label.setText(messages.getString("con_newCategoryLabel"));
166         category.setOrderNumber(categories.size());
167         category.setLabel(label);
168 
169         List<CategoryEntity> labelCategoryEntities = label.getCategoryEntities();
170         if (labelCategoryEntities == null) {
171             labelCategoryEntities = new ArrayList<>();
172         }
173         labelCategoryEntities.add(category);
174         label.setCategoryEntities(labelCategoryEntities);
175 
176         category.setCategoryType(CategoryType.TIMED);
177         categories.add(category);
178         selectedCategory = category;
179     }
180 
181     /**
182      * Listener for event group row edit.
183      */
184     public void onEditEventGroup(RowEditEvent event) {
185         EventGroupEntity eventGroup = (EventGroupEntity) event.getObject();
186         eventGroupEJB.edit(eventGroup);
187     }
188 
189     /**
190      * Gets the event groups.
191      */
```

```
192     public List<EventGroupEntity> getEventGroups() {
193        return eventGroups;
194     }
195
196     /**
197      * Sets the event groups.
198      */
199     public void setEventGroups(List<EventGroupEntity> eventGroups) {
200        this.eventGroups = eventGroups;
201     }
202
203     /**
204      * Gets the selected event group.
205      */
206     public EventGroupEntity getSelectedEventGroup() {
207        return selectedEventGroup;
208     }
209
210     /**
211      * Sets the selected event group.
212      */
213     public void setSelectedEventGroup(EventGroupEntity selectedEventGroup) {
214        this.selectedEventGroup = selectedEventGroup;
215     }
216
217     /**
218      * Gets the categories.
219      */
220     public List<CategoryEntity> getCategories() {
221        return categories;
222     }
223
224     /**
225      * Sets the categories.
226      */
227     public void setCategories(List<CategoryEntity> categories) {
228        this.categories = categories;
229     }
230
231     /**
232      * Gets the selected category set.
233      */
234     public CategorySetEntity getSelectedCategorySet() {
235        return selectedCategorySet;
236     }
237
238     /**
239      * Sets the selected category set.
240      */
241     public void setSelectedCategorySet(CategorySetEntity selectedCategorySet) {
242        this.selectedCategorySet = selectedCategorySet;
243        this.selectedEventGroup = this.selectedCategorySet.getEventGroupEntity();
244        categories = new ArrayList<>(selectedCategorySet.getCategoryEntitys().values
               ());
245     }
246
247     /**
248      * Gets the selected category.
```

```
249      */
250     public CategoryEntity getSelectedCategory() {
251        return selectedCategory;
252     }
253
254     /**
255      * Sets the selected category.
256      */
257     public void setSelectedCategory(CategoryEntity selectedCategory) {
258        this.selectedCategory = selectedCategory;
259     }
260
261     /**
262      * Gets the selected observation.
263      */
264     public ObservationEntity getSelectedObservation() {
265        return selectedObservation;
266     }
267
268     /**
269      * Sets the selected observation.
270      */
271     public void setSelectedObservation(ObservationEntity selectedObservation) {
272        this.selectedObservation = selectedObservation;
273     }
274
275     /**
276      * Gets the other observations.
277      */
278     public List<ObservationEntity> getOtherObservations() {
279        return otherObservations;
280     }
281
282     /**
283      * Sets the other observations.
284      */
285     public void setOtherObservations(List<ObservationEntity> otherObservations) {
286        this.otherObservations = otherObservations;
287     }
288
289     /**
290      * Gets the category types.
291      */
292     public CategoryType[] getCategoryTypes() {
293        return CategoryType.values();
294     }
295
296     /**
297      * Gets the name of the observer of the selected observation entity.
298      */
299     public String getObserverName() {
300        if (selectedObservation == null) {
301           return "";
302        } else if (selectedObservation.getObserver() instanceof IdentifiedUserEntity
               ) {
303           return ((IdentifiedUserEntity) selectedObservation.getObserver()).
                  getGivenName();
304        } else {
```

```
305             return messages.getString("con_publicUser");
306          }
307       }
308
309       /**
310        * Initializes a new observation and redirects to the category selection view.
311        *
312        * @return The navigation rule string that redirects to the category selection
               view.
313        */
314       public String newObservation() {
315          observationBean.setEventEntity(selectedEventGroup.getEvent());
316          // Make sure we don't modify earlier categories.
317          observationBean.resetCategorySetsInUse();
318          return "newobservation";
319       }
320
321       /**
322        * Removes the event group from the database.
323        */
324       public void removeEventGroup(EventGroupEntity eventGroup) {
325          if (eventGroup != null) {
326             // remove group key first
327             if (eventGroup.getGroupKey() != null) {
328                eventGroupBean.removeGroupKey(eventGroup);
329             }
330             eventGroupEJB.remove(eventGroup);
331             eventGroups.remove(eventGroup);
332          }
333       }
334
335       /**
336        * Removes the selected category set from the database.
337        */
338       public void removeCategorySet() {
339          if (selectedCategorySet != null) {
340             categorySetEJB.remove(selectedCategorySet);
341             selectedCategorySet = null;
342             selectedCategory = null;
343             // refetch eventgroups, maybe other way to update it?
344             fetchEventGroups();
345          }
346       }
347
348       /**
349        * Removes the selected category from the view, reorders the categories and
350        * selects a new category.
351        */
352       public void removeCategory() {
353          if (selectedCategory != null) {
354             int index = selectedCategory.getOrderNumber();
355             categories.remove(index);
356             int i = 0;
357             for (CategoryEntity category : categories) {
358                category.setOrderNumber(i);
359                i++;
360             }
361             if (categories.isEmpty()) {
```

```
362             selectedCategory = null;
363          } else if (index > 0) {
364             selectedCategory = categories.get(index - 1);
365          } else {
366             selectedCategory = categories.get(0);
367          }
368       }
369    }
370
371    /**
372     * Removes the selected observation from the database.
373     */
374    public void removeObservation() {
375       if (selectedObservation != null) {
376          observationEJB.remove(selectedObservation);
377          selectedObservation = null;
378          fetchEventGroups();
379       }
380    }
381
382    /**
383     * ReorderEvent listener for categories reorder.
384     */
385    public void onCategoryReorder(ReorderEvent event) {
386       int i = 0;
387       for (CategoryEntity category : categories) {
388          category.setOrderNumber(i);
389          i++;
390       }
391    }
392
393    /**
394     * Listener for observation editing.
395     */
396    public void onEditObservation() {
397       if (selectedObservation != null) {
398          observationEJB.edit(selectedObservation);
399       }
400    }
401
402    /**
403     * Adds a new event group to the event groups.
404     */
405    public void addEventGroup(EventGroupEntity eventGroup) {
406       eventGroups.add(eventGroup);
407    }
408
409    /**
410     * Saves the selected category set.
411     */
412    public void saveCategorySet() {
413       if (selectedEventGroup != null && selectedCategorySet != null) {
414          if (!hasDuplicate()) {
415             categorySetBean.createAndEditCategorySet(selectedEventGroup,
                   selectedCategorySet, categories);
416             fetchEventGroups();
417          } else {
418             FacesContext.getCurrentInstance().validationFailed();
```

```
419             FacesContext.getCurrentInstance().addMessage(null,
420                   new FacesMessage(FacesMessage.SEVERITY_ERROR,
421                         messages.getString("dialogErrorTitle"),
422                         messages.getString("cs_errorNotUniqueCategories")));
423          }
424       }
425    }
426
427    /**
428     * Shows the selected observation in the summary page.
429     *
430     * @return The navigation rule string that redirects to the summary page.
431     */
432    public String showObservationInSummaryPage() {
433       observationBean.setObservationEntity(selectedObservation);
434       observationBean.setCategorySetsInUse(new ArrayList<>(selectedObservation.
             getObservationCategorySets()));
435       return "summary";
436    }
437
438    /**
439     * Checks if the categories have duplicates.
440     */
441    private boolean hasDuplicate() {
442       Set<String> duplicates = new HashSet<>();
443       for (CategoryEntity categoryEntity : categories) {
444          String categoryText = categoryEntity.getLabel().getText();
445          if (!categoryText.isEmpty() && !duplicates.add(categoryText)) {
446             selectedCategory = categoryEntity;
447             return true;
448          }
449       }
450       return false;
451    }
452
453    /**
454     * Converts milliseconds to string with time units h, m, s.
455     *
456     * @param ms The time to be converted in milliseconds.
457     * @return String of the converted time units.
458     */
459    public String msToUnits(long ms) {
460       if (ms <= 0) {
461          return "0 s";
462       }
463       if (ms < 1000) {
464          return "~1 s";
465       }
466       String hms = String.format("%d h %d m %d s", TimeUnit.MILLISECONDS.toHours(
             ms),
467             TimeUnit.MILLISECONDS.toMinutes(ms) % TimeUnit.HOURS.toMinutes(1),
468             TimeUnit.MILLISECONDS.toSeconds(ms) % TimeUnit.MINUTES.toSeconds(1));
469       hms = hms.replaceFirst("0 h ", "");
470       hms = hms.replaceFirst("0 m ", "");
471       return hms;
472    }
473
474    /**
```

```
475     * Gets the name of the event group of the observation.
476     */
477    public String getObservationEventGroupName(ObservationEntity observationEntity)
           {
478       EventEntity eventEntity = observationEntity.getEvent();
479       if (eventEntity != null && eventEntity.getEventGroup() != null) {
480          return eventEntity.getEventGroup().getLabel();
481       }
482       return "";
483    }
484 }
```

1.58 managedbeans/EventGroupManagedBean.java

```
1  package com.moveatis.managedbeans;
2
3  import com.moveatis.event.EventEntity;
4  import com.moveatis.event.EventGroupEntity;
5  import com.moveatis.groupkey.GroupKeyEntity;
6  import com.moveatis.interfaces.AnonUser;
7  import com.moveatis.interfaces.EventGroup;
8  import com.moveatis.interfaces.GroupKey;
9  import com.moveatis.interfaces.Session;
10 import com.moveatis.user.AbstractUser;
11 import com.moveatis.user.TagUserEntity;
12 import java.util.HashSet;
13 import java.util.Set;
14 import javax.annotation.PostConstruct;
15 import javax.enterprise.context.RequestScoped;
16 import javax.inject.Named;
17 import javax.inject.Inject;
18 import org.slf4j.Logger;
19 import org.slf4j.LoggerFactory;
20
21 /**
22  * The bean is used to manage event groups in the appropriate views.
23  * @author Sami Kallio <phinaliumz at outlook.com>
24  */
25 @Named(value = "eventGroupManagedBean")
26 @RequestScoped
27 public class EventGroupManagedBean {
28
29    private static final Logger LOGGER = LoggerFactory.getLogger(
          EventGroupManagedBean.class);
30
31    private String eventGroupName;
32    private String eventGroupKey;
33    private String eventGroupDescription;
34    private String visibility;
35
36    private boolean groupKeySelected;
37
38    private EventGroupEntity eventGroupEntity;
39
40    @Inject
```

```
41      private EventGroup eventGroupEJB;
42
43      @Inject
44      private GroupKey groupKeyEJB;
45
46      @Inject
47      private Session sessionBean;
48
49      @Inject
50      private AnonUser anonUserEJB;
51
52      @Inject
53      private ControlManagedBean controlManagedBean;
54
55      public EventGroupManagedBean() {
56
57      }
58
59      @PostConstruct
60      public void init() {
61         this.visibility = "own";
62      }
63
64      public String getEventGroupName() {
65         return eventGroupName;
66      }
67
68      public void setEventGroupName(String eventGroupName) {
69         this.eventGroupName = eventGroupName;
70      }
71
72      public String getVisibility() {
73         return visibility;
74      }
75
76      public void setVisibility(String visibility) {
77         groupKeySelected = visibility.equalsIgnoreCase("groupKey");
78         this.visibility = visibility;
79      }
80
81      public String getEventGroupDescription() {
82         return eventGroupDescription;
83      }
84
85      public void setEventGroupDescription(String eventGroupDescription) {
86         this.eventGroupDescription = eventGroupDescription;
87      }
88
89      public String getEventGroupKey() {
90         return eventGroupKey;
91      }
92
93      public void setEventGroupKey(String eventGroupKey) {
94         this.eventGroupKey = eventGroupKey;
95      }
96
97      public boolean isGroupKeySelected() {
98         return groupKeySelected;
```

```
99      }
100
101     public void setGroupKeySelected(boolean groupKeySelected) {
102        this.groupKeySelected = groupKeySelected;
103     }
104
105     public void addGroupKey() {
106        groupKeySelected = this.visibility.equalsIgnoreCase("groupKey");
107     }
108
109     /**
110      * Adds the group key to the specified event group. The group key is used
111      * to identify the event group.
112      * @param eventGroup The event group into which the the group key is added.
113      */
114     public void addGroupKey(EventGroupEntity eventGroup) {
115
116        if (eventGroupKey != null) {
117
118           GroupKeyEntity groupKey = new GroupKeyEntity();
119           groupKey.setCreator(sessionBean.getLoggedIdentifiedUser());
120           groupKey.setGroupKey(eventGroupKey);
121           groupKey.setEventGroup(eventGroup);
122           eventGroup.setGroupKey(groupKey);
123
124           TagUserEntity tagUserEntity = new TagUserEntity();
125           tagUserEntity.setCreator(sessionBean.getLoggedIdentifiedUser());
126           tagUserEntity.setLabel(eventGroupKey);
127           tagUserEntity.setGroupKey(groupKey);
128           groupKey.setTagUser(tagUserEntity);
129
130           groupKeyEJB.create(groupKey);
131
132           eventGroupEJB.edit(eventGroup);
133
134           controlManagedBean.fetchEventGroups();
135        }
136     }
137
138     /**
139      * Sets a new group key for the event group.
140      *
141      * @param eventGroup The event group to edit.
142      * @param newGroupKey The new group key for the event group.
143      */
144     public void editGroupKey(EventGroupEntity eventGroup, String newGroupKey) {
145        if (newGroupKey != null) {
146           GroupKeyEntity groupKeyEntity = eventGroup.getGroupKey();
147           groupKeyEntity.setCreator(sessionBean.getLoggedIdentifiedUser());
148           groupKeyEntity.setGroupKey(newGroupKey);
149           groupKeyEntity.setEventGroup(eventGroup);
150           eventGroup.setGroupKey(groupKeyEntity);
151
152           TagUserEntity tagUserEntity = groupKeyEntity.getTagUser();
153           tagUserEntity.setGroupKey(groupKeyEntity);
154
155           groupKeyEJB.edit(groupKeyEntity);
156           controlManagedBean.fetchEventGroups();
```

```
157         }
158     }
159 
160     /**
161      * Removes the group key from the given event group.
162      */
163     public void removeGroupKey(EventGroupEntity eventGroup) {
164         GroupKeyEntity groupKey = eventGroup.getGroupKey();
165         eventGroup.setGroupKey(null);
166         eventGroupEJB.edit(eventGroup);
167         groupKeyEJB.removePermanently(groupKey);
168         controlManagedBean.fetchEventGroups();
169     }
170 
171     /**
172      * Creates a new event group.
173      */
174     public void createNewEventGroup() {
175 
176         eventGroupEntity = new EventGroupEntity();
177         eventGroupEntity.setLabel(eventGroupName);
178         eventGroupEntity.setDescription(eventGroupDescription);
179         eventGroupEntity.setOwner(sessionBean.getLoggedIdentifiedUser());
180 
181         if(groupKeySelected) {
182             GroupKeyEntity groupKey = new GroupKeyEntity();
183             groupKey.setCreator(sessionBean.getLoggedIdentifiedUser());
184             groupKey.setGroupKey(eventGroupKey);
185             groupKey.setEventGroup(eventGroupEntity);
186 
187             TagUserEntity tagUser = new TagUserEntity();
188             tagUser.setCreator(sessionBean.getLoggedIdentifiedUser());
189             tagUser.setGroupKey(groupKey);
190 
191             groupKey.setTagUser(tagUser);
192 
193             eventGroupEntity.setGroupKey(groupKey);
194 
195         } else if(visibility.equalsIgnoreCase("public")) {
196             Set<AbstractUser> users = new HashSet<>();
197             users.add(anonUserEJB.find());
198             eventGroupEntity.setUsers(users);
199         }
200         /*
201         * As agreed on meeting 5.5.2016, eventGroup will be renamed in the UI as "
                event",
202         * while eventgroups and events in the code stay the same.
203          */
204         EventEntity eventEntity = new EventEntity();
205         eventEntity.setCreator(sessionBean.getLoggedIdentifiedUser());
206         eventEntity.setLabel("DEFAULT");
207         eventEntity.setEventGroup(eventGroupEntity);
208 
209         eventGroupEntity.setEvent(eventEntity);
210         eventGroupEJB.create(eventGroupEntity);
211 
212         controlManagedBean.addEventGroup(eventGroupEntity);
213         controlManagedBean.setCreatingNewEventGroup(false);
```

```
214     }
215 }
```

1.59 managedbeans/EventManagedBean.java

```
1
2  package com.moveatis.managedbeans;
3
4  import com.moveatis.event.EventEntity;
5  import com.moveatis.event.EventGroupEntity;
6  import com.moveatis.interfaces.Event;
7  import com.moveatis.interfaces.Session;
8  import javax.inject.Named;
9  import javax.enterprise.context.RequestScoped;
10 import javax.inject.Inject;
11 import org.slf4j.Logger;
12 import org.slf4j.LoggerFactory;
13
14 /**
15  * The bean is used to manage events in the appropriate views.
16  * @author Sami Kallio <phinaliumz at outlook.com>
17  */
18 @Named(value="eventManagedBean")
19 @RequestScoped
20 public class EventManagedBean {
21
22    private static final Logger LOGGER = LoggerFactory.getLogger(EventManagedBean.
          class);
23
24    private String label;
25    private String description;
26    private EventEntity eventEntity;
27
28    @Inject
29    private Event eventEJB;
30
31    @Inject
32    private Session sessionBean;
33
34    /** Creates a new instance of EventManagedBean. */
35    public EventManagedBean() {
36
37    }
38
39    public String getLabel() {
40       return label;
41    }
42
43    public void setLabel(String label) {
44       this.label = label;
45    }
46
47    public String getDescription() {
48       return description;
49    }
```

```
50
51      public void setDescription(String description) {
52          this.description = description;
53      }
54
55      /**
56       * Creates a new event for the event group.
57       */
58      public void createNewEvent(EventGroupEntity eventGroupEntity) {
59
60          eventEntity = new EventEntity();
61          eventEntity.setCreator(sessionBean.getLoggedIdentifiedUser());
62          eventEntity.setDescription(description);
63          eventEntity.setLabel(label);
64          eventEntity.setEventGroup(eventGroupEntity);
65
66          eventGroupEntity.setEvent(eventEntity);
67
68          eventEJB.create(eventEntity);
69
70      }
71
72  }
```

1.60 managedbeans/LoginManagedBean.java

```
1   package com.moveatis.managedbeans;
2
3   import com.moveatis.application.RedirectURLs;
4   import com.moveatis.groupkey.GroupKeyEntity;
5   import com.moveatis.interfaces.GroupKey;
6   import com.moveatis.interfaces.Session;
7   import com.moveatis.interfaces.TagUser;
8   import com.moveatis.user.TagUserEntity;
9   import java.io.IOException;
10  import java.util.Map;
11  import java.util.ResourceBundle;
12  import javax.enterprise.context.RequestScoped;
13  import javax.faces.application.FacesMessage;
14  import javax.faces.bean.ManagedBean;
15  import javax.faces.bean.ManagedProperty;
16  import javax.faces.context.FacesContext;
17  import javax.faces.event.ActionEvent;
18  import javax.inject.Inject;
19  import javax.servlet.http.HttpServletRequest;
20  import org.primefaces.component.menuitem.UIMenuItem;
21  import org.primefaces.context.RequestContext;
22  import org.slf4j.Logger;
23  import org.slf4j.LoggerFactory;
24
25  /**
26   * The bean that manages the login for three types of users: the public user,
27   * the tag user and the identified user.
28   * @author Sami Kallio <phinaliumz at outlook.com>
29   */
```

```
30 @ManagedBean(name="loginBean")
31 @RequestScoped
32 public class LoginManagedBean {
33
34    private static final Logger LOGGER = LoggerFactory.getLogger(LoginManagedBean.
          class);
35
36    @ManagedProperty("#{msg}")
37    private ResourceBundle messages;
38
39    @Inject
40    private Session sessionBean;
41
42    @Inject
43    private GroupKey groupKeyEJB;
44
45    @Inject
46    private TagUser tagUserEJB;
47
48    private String loginOutcome;
49    private String tag;
50
51    /**
52     * Creates a new instance of LoginManagedBean.
53     */
54    public LoginManagedBean() {
55
56    }
57
58    public String getTag() {
59       return tag;
60    }
61
62    public void setTag(String tag) {
63       this.tag = tag;
64    }
65
66    /**
67     * Sets the session for a particular group key.
68     * @return The navigation rule string that redirects to the category selection
            view.
69     */
70    public String doTagLogin() {
71       GroupKeyEntity groupKeyEntity = groupKeyEJB.findByKey(tag);
72
73       if(groupKeyEntity == null) {
74          RequestContext.getCurrentInstance().showMessageInDialog(
75                new FacesMessage(FacesMessage.SEVERITY_ERROR, messages.getString("
                      dialogErrorTitle"),
76                messages.getString("tagNotFound")));
77       }
78
79       TagUserEntity tagUserEntity = tagUserEJB.findByKey(groupKeyEntity);
80
81       if(tagUserEntity == null) {
82          RequestContext.getCurrentInstance().showMessageInDialog(
83                new FacesMessage(FacesMessage.SEVERITY_ERROR, messages.getString("
                      dialogErrorTitle"),
```

```
84                messages.getString("tagNotFound")));
85          } else {
86             sessionBean.setTagUser(tagUserEntity);
87             this.loginOutcome = "tagUser";
88             return "taguser";
89          }
90
91          return "";
92       }
93
94       /**
95        * Sets the session for a public user.
96        * @return The navigation rule string that redirects to the category selection
              view.
97        */
98       public String doAnonymityLogin() {
99          sessionBean.setAnonymityUser();
100         return "anonymityuser";
101      }
102
103      /**
104       * Allows users to login from different views. On May 2016, it's not
105       * working as supposed, since Shibboleth nulls the session on redirect.
106       * @param actionEvent The action event that activated the login button.
107       */
108      public void doIdentityLogin(ActionEvent actionEvent) {
109
110         String secureRedirectUri, defaultUri;
111
112         if(((HttpServletRequest) FacesContext.getCurrentInstance().
                getExternalContext().getRequest())
113                .getRequestURL().toString().contains("localhost")) {
114            secureRedirectUri = RedirectURLs.LOCALHOST_REDIRECT_SECURE_URI;
115            defaultUri = RedirectURLs.LOCALHOST_HOME_URI;
116         } else {
117            secureRedirectUri = RedirectURLs.SHIBBOLETH_REDIRECT_SECURE_URI;
118            defaultUri = RedirectURLs.HOME_URI;
119         }
120         try {
121            Object source = actionEvent.getSource();
122            if(source instanceof UIMenuItem) {
123               // Clicked in menu so we need to redirect to page where user clicked
                     login
124               Map<String, String> params = FacesContext.getCurrentInstance().
                     getExternalContext().getRequestParameterMap();
125               sessionBean.setReturnUri(defaultUri + params.get("view"));
126            }
127
128            FacesContext.getCurrentInstance().getExternalContext().redirect(
                  secureRedirectUri);
129         } catch (IOException ex) {
130            LOGGER.error("An error happened in identitylogin" + ex.toString());
131         }
132      }
133
134      public ResourceBundle getMessages() {
135         return messages;
136      }
```

```
137
138     public void setMessages(ResourceBundle messages) {
139         this.messages = messages;
140     }
141
142     public String getLoginOutcome() {
143         return loginOutcome;
144     }
145
146     public void setLoginOutcome(String loginOutcome) {
147         this.loginOutcome = loginOutcome;
148     }
149 }
```

1.61 managedbeans/ObservationManagedBean.java

```
1  package com.moveatis.managedbeans;
2  import com.moveatis.event.EventEntity;
3  import com.moveatis.interfaces.Observation;
4  import com.moveatis.interfaces.Session;
5  import com.moveatis.observation.ObservationCategory;
6  import com.moveatis.observation.ObservationCategorySet;
7  import com.moveatis.observation.ObservationEntity;
8  import com.moveatis.records.RecordEntity;
9  import java.io.Serializable;
10 import java.util.ArrayList;
11 import java.util.HashSet;
12 import java.util.List;
13 import javax.annotation.PostConstruct;
14 import javax.annotation.PreDestroy;
15 import javax.ejb.EJB;
16 import javax.enterprise.context.SessionScoped;
17 import javax.inject.Inject;
18 import javax.inject.Named;
19 import org.slf4j.Logger;
20 import org.slf4j.LoggerFactory;
21
22 /**
23  * The bean is used to manage observations in the appropriate views.
24  * @author Sami Kallio <phinaliumz at outlook.com>
25  */
26 @Named(value = "observationBean")
27 @SessionScoped
28 public class ObservationManagedBean implements Serializable {
29
30     private static final Logger LOGGER = LoggerFactory.getLogger(
          ObservationManagedBean.class);
31
32     private static final long serialVersionUID = 1L;
33
34     private ObservationEntity observationEntity;
35     private List<ObservationCategorySet> categorySetsInUse;
36     private EventEntity eventEntity;
37
38     @EJB
```

```
39      private Observation observationEJB;
40      @Inject
41      private Session sessionBean;
42
43      // Tag is used to identify the observationcategories within a observation
44      private Long nextTag;
45
46      public ObservationManagedBean() {
47
48      }
49
50      @PostConstruct
51      public void init() {
52         nextTag = 0L;
53      }
54
55      /**
56       * Removes the observations the user doesn't want to save to database
57       * when the session timeout happens and the bean is destroyed.
58       */
59      @PreDestroy
60      public void destroy() {
61         if(observationEntity != null) {
62            if(!observationEntity.getUserWantsToSaveToDatabase()) {
63               observationEJB.removeUnsavedObservation(observationEntity);
64            }
65         }
66      }
67
68      public void resetCategorySetsInUse() {
69         this.categorySetsInUse = null;
70      }
71
72      public void setEventEntity(EventEntity eventEntity) {
73         this.eventEntity = eventEntity;
74      }
75
76      public EventEntity getEventEntity() {
77         return this.eventEntity;
78      }
79
80      /**
81       * Creates a new observation entity and initializes it to be used in a new
82       * observation.
83       */
84      public void startObservation() {
85         this.observationEntity = new ObservationEntity();
86         // Can we use created time for observation start time?
87         this.observationEntity.setCreated();
88         this.observationEntity.setEvent(eventEntity);
89         // Summary view doesn't break if no records are added.
90         // TODO: Should observer not let user continue, if there are no records?
91         if(observationEntity.getRecords() == null) {
92            observationEntity.setRecords(new ArrayList<RecordEntity>());
93         }
94      }
95
96      /**
```

```
97      * Returns the current observation entity.
98      */
99     public ObservationEntity getObservationEntity() {
100        return observationEntity;
101    }
102
103    /**
104     * Sets the current observation entity.
105     */
106    public void setObservationEntity(ObservationEntity observationEntity) {
107       this.observationEntity = observationEntity;
108    }
109
110    /**
111     * Gets the observation categories to be used in the observation.
112     */
113    public List<ObservationCategorySet> getCategorySetsInUse() {
114       return categorySetsInUse;
115    }
116
117    /**
118     * Sets the observation categories to be used in the observation.
119     */
120    public void setCategorySetsInUse(List<ObservationCategorySet> categorySetsInUse
       ) {
121
122       for(ObservationCategorySet observationCategorySet : categorySetsInUse) {
123          for(ObservationCategory observationCategory : observationCategorySet.
              getCategories()) {
124             if(observationCategory.getTag().equals(-1L)) {
125                observationCategory.setTag(getNextTag());
126             }
127          }
128       }
129       this.categorySetsInUse = categorySetsInUse;
130    }
131
132    public Long getNextTag() {
133       return nextTag++;
134    }
135
136    public void setObservationName(String name) {
137       this.observationEntity.setName(name);
138    }
139
140    public void setObservationDuration(long duration) {
141       this.observationEntity.setDuration(duration);
142    }
143
144    /**
145     * Adds a record to the observation.
146     * @param record The record to be added to the observation.
147     */
148    public void addRecord(RecordEntity record) {
149       List<RecordEntity> records = observationEntity.getRecords();
150       record.setObservation(observationEntity);
151       record.setVoiceComment(null);
152
```

```
153         if(records == null) {
154             records = new ArrayList<>();
155         }
156 
157         records.add(record);
158         observationEntity.setRecords(records);
159     }
160 
161     /**
162      * The method is called from REST API to save the records to the observation.
163      */
164     public void saveObservation() {
165         if (sessionBean.isIdentifiedUser()) {
166             observationEntity.setObserver(sessionBean.getLoggedIdentifiedUser());
167         }
168         /*
169         NOTE: GroupKey couldn't be removed if there were observations whose
170               observer was the TagUser corresponding to the GroupKey.
171               Solution: don't set observer if not identified user.
172 
173         else { TODO: Can we leave observer unset? Should we ensure it is null or...?
174             observationEntity.setObserver(sessionBean.getLoggedInUser());
175         }*/
176 
177         /*
178         * Client wanted that user has the ability to persist observation into
179         * database when he/she wants to. Since it was easier to build business logic
180         * by creating the observationEntity when observation is done, the boolean
              flag
181         * "userWantsToSaveToDatabase was added. This flag is true if user wants to
              save
182         * the observation to database. If its false, we need remove this observation
               later.
183         */
184         observationEntity.setUserWantsToSaveToDatabase(false);
185         observationEJB.create(observationEntity);
186     }
187 
188     /**
189      * The method saves the observation to the database.
190      */
191     public void saveObservationToDatabase() {
192         observationEntity.setUserWantsToSaveToDatabase(true);
193         observationEntity.setObservationCategorySets(new HashSet<>(
              getCategorySetsInUse()));
194         observationEJB.edit(observationEntity);
195     }
196 }
```

1.62 managedbeans/SummaryManagedBean.java

```
1 package com.moveatis.managedbeans;
2 
3 import com.moveatis.export.CSVFileBuilder;
4 import com.moveatis.interfaces.Mailer;
```

```
5  import com.moveatis.interfaces.MessageBundle;
6  import com.moveatis.interfaces.Observation;
7  import com.moveatis.interfaces.Session;
8  import com.moveatis.observation.ObservationCategory;
9  import com.moveatis.observation.ObservationCategorySet;
10 import com.moveatis.observation.ObservationEntity;
11 import com.moveatis.records.RecordEntity;
12 import java.io.File;
13 import java.io.FileOutputStream;
14 import java.io.IOException;
15 import java.io.OutputStream;
16 import java.io.Serializable;
17 import java.text.MessageFormat;
18 import java.util.ArrayList;
19 import java.util.Date;
20 import java.util.List;
21 import java.util.ResourceBundle;
22 import javax.annotation.PostConstruct;
23 import javax.faces.application.FacesMessage;
24 import javax.faces.context.ExternalContext;
25 import javax.faces.context.FacesContext;
26 import javax.inject.Inject;
27 import javax.inject.Named;
28 import org.primefaces.extensions.model.timeline.TimelineEvent;
29 import org.primefaces.extensions.model.timeline.TimelineModel;
30 import org.slf4j.Logger;
31 import org.slf4j.LoggerFactory;
32 import javax.faces.view.ViewScoped;
33 import org.apache.commons.lang3.StringEscapeUtils;
34 import org.apache.commons.lang3.StringUtils;
35 import org.primefaces.extensions.model.timeline.TimelineGroup;
36
37 /**
38  * The bean that serves the summary page. It is responsible for creating the
39  * timeline model and for getting the attributes of an observation for the summary.
40  *
41  * @author Juha Moisio <juha.pa.moisio at student.jyu.fi>
42  */
43 @Named(value = "summaryBean")
44 @ViewScoped
45 public class SummaryManagedBean implements Serializable {
46
47    private TimelineModel timeline;
48    private final Date min;
49    private final Date start;
50    private final long zoomMin;
51    private final long zoomMax;
52    private Date max;
53    private Date duration;
54    private String recipientEmail;
55
56    private ObservationEntity observation;
57
58    private List<String> selectedSaveOptions;
59
60    private static final String MAIL_OPTION = "mail";
61    private static final String SAVE_OPTION = "save";
62    private static final String DOWNLOAD_OPTION = "download";
```

```
63
64      private boolean observationSaved = false;
65
66      @Inject
67      private Observation observationEJB;
68
69      @Inject
70      private Session sessionBean;
71      @Inject
72      private ObservationManagedBean observationManagedBean;
73      @Inject
74      private Mailer mailerEJB;
75
76      @Inject
77      private ValidationManagedBean validationBean;
78
79      @Inject
80      @MessageBundle
81      private transient ResourceBundle messages;
82
83      private static final Logger LOGGER = LoggerFactory.getLogger(SummaryManagedBean
            .class);
84
85      /**
86       * The default constructor initializes the timeline options.
87       */
88      public SummaryManagedBean() {
89         this.start = new Date(0);
90         this.min = new Date(0);
91         this.max = new Date(0);
92         this.zoomMin = 10 * 1000;
93         this.zoomMax = 24 * 60 * 60 * 1000;
94         this.selectedSaveOptions = new ArrayList<>();
95      }
96
97      /**
98       * The post constructor creates the timeline on request.
99       */
100     @PostConstruct
101     protected void initialize() {
102        createTimeline();
103     }
104
105     /**
106      * Create timeline model. Add category groups as timeline event groups and
107      * records as timeline events.
108      */
109     private void createTimeline() {
110        timeline = new TimelineModel();
111
112        observation = observationManagedBean.getObservationEntity();
113        List<RecordEntity> records = observation.getRecords();
114
115        duration = new Date(observation.getDuration());
116        max = new Date(Math.round(this.observation.getDuration() * 1.1)); //
               timeline max 110% of obs. duration
117
118        // Add categories to timeline as timeline groups
```

```
119         int categoryNumber = 1;
120         for (ObservationCategorySet categorySet : observationManagedBean.
                getCategorySetsInUse()) {
121             for (ObservationCategory category : categorySet.getCategories()) {
122                 // Add category name inside element with class name
123                 // use css style to hide them in timeline
124                 // class name is intentionally without quotes, timeline expectional
                        case
125                 String numberedLabel
126                       = "<span class=categoryNumber>" + categoryNumber + ". </span>"
127                       + "<span class=categoryLabel>" + StringEscapeUtils.escapeHtml4(
                              category.getName()) + "</span>"
128                       + "<span class=categorySet>" + StringEscapeUtils.escapeHtml4(
                              categorySet.getName()) + "</span>";
129                 String groupID = Long.toString(category.getTag());
130                 TimelineGroup timelineGroup = new TimelineGroup(groupID, numberedLabel
                        );
131                 timeline.addGroup(timelineGroup);
132                 // Add dummy records to show empty categories in timeline
133                 TimelineEvent timelineEvent = new TimelineEvent("", new Date(0), false
                        , groupID, "dummyRecord");
134                 timeline.add(timelineEvent);
135                 categoryNumber++;
136             }
137         }
138 
139         // Add records to timeline as timeline-events
140         for (RecordEntity record : records) {
141             ObservationCategory category = record.getCategory();
142             long startTime = record.getStartTime();
143             long endTime = record.getEndTime();
144             TimelineEvent timelineEvent = new TimelineEvent("", new Date(startTime),
145                     new Date(endTime), false, Long.toString(category.getTag()));
146             timeline.add(timelineEvent);
147         }
148     }
149 
150     /**
151      * Shows a message of the saved observation.
152      */
153     public void showObservationSavedMessage() {
154         if (observationSaved) {
155             FacesContext.getCurrentInstance().addMessage(null,
156                     new FacesMessage(FacesMessage.SEVERITY_INFO, messages.getString("
                            sum_observationSaved"), ""));
157             observationSaved = false;
158         }
159     }
160 
161     /**
162      * Saves the current observation to the database.
163      */
164     public void saveCurrentObservation() {
165         observationManagedBean.saveObservationToDatabase();
166         observationSaved = true;
167     }
168 
169     /**
```

```
170      *
171      */
172     public void mailCurrentObservation() {
173        CSVFileBuilder csv = new CSVFileBuilder();
174        FacesContext context = FacesContext.getCurrentInstance();
175        ResourceBundle bundle = context.getApplication().getResourceBundle(context,
               "msg");
176        try {
177           String fileName = this.observation.getName();
178           // replace non-word ![a-ZA-Z_0-9] chars with underscope
179           fileName = fileName.replaceAll("\\W", "_");
180           File f = File.createTempFile(fileName, ".csv");
181
182           StringBuilder msgBuilder = new StringBuilder();
183           String description = StringUtils.defaultIfEmpty(observation.
                  getDescription(),
184                    bundle.getString("sum_descriptionNotSet"));
185           String target = StringUtils.defaultIfEmpty(observation.getTarget(),
186                 bundle.getString("sum_targetNotSet"));
187           String descriptionPartOfMessage = MessageFormat.format(bundle.getString("
                  sum_descriptionWas"), description);
188           String targetPartOfMessage = MessageFormat.format(bundle.getString("
                  sum_targetWas"), target);
189           String messageWithSender = MessageFormat.format(bundle.getString("
                  sum_message"),
190                    sessionBean.getLoggedIdentifiedUser().getGivenName());
191
192           msgBuilder
193                 .append(messageWithSender)
194                 .append("\n\n")
195                 .append(descriptionPartOfMessage)
196                 .append("\n\n")
197                 .append(targetPartOfMessage)
198                 .append("\n\n")
199                 .append(bundle.getString("emailSignature"));
200
201           FileOutputStream fos = new FileOutputStream(f);
202           csv.buildCSV(fos, observation, ",");
203           fos.flush();
204           String[] recipients = {recipientEmail};
205           File[] files = {f};
206           mailerEJB.sendEmailWithAttachment(recipients, bundle.getString("
                  sum_subject"),
207                    msgBuilder.toString(), files);
208           //remove the temp file after sending it
209           f.delete(); // TODO: Check the return value.
210        } catch (IOException ex) {
211           LOGGER.error("Failed to create temporary file for sending observeraion by
                   email.", ex);
212        }
213        observationSaved = true;
214     }
215
216     /**
217      * File name converter.
218      */
219     private static String convertToFilename(String s) {
220        if (s == null || s.isEmpty()) {
```

```
221             return "unnamed";
222         }
223         return s.replaceAll("[^a-zA-Z0-9_]", "_");
224     }
225 
226     /**
227      * Downloads the current observation.
228      *
229      * @throws IOException
230      */
231     public void downloadCurrentObservation() throws IOException {
232         String fileName = convertToFilename(observation.getName()) + ".csv";
233 
234         FacesContext facesCtx = FacesContext.getCurrentInstance();
235         ExternalContext externalCtx = facesCtx.getExternalContext();
236 
237         externalCtx.responseReset();
238         externalCtx.setResponseContentType("text/plain");
239         externalCtx.setResponseHeader("Content-Disposition", "attachment; filename
                =\"" + fileName + "\"");
240 
241         OutputStream outputStream = externalCtx.getResponseOutputStream();
242 
243         CSVFileBuilder csv = new CSVFileBuilder();
244         csv.buildCSV(outputStream, observation, ",");
245         outputStream.flush();
246 
247         facesCtx.responseComplete();
248 
249         observationSaved = true;
250     }
251 
252     /**
253      * Do all the save operations selected by the user.
254      */
255     public void doSelectedSaveOperation() {
256         if (selectedSaveOptions.contains(DOWNLOAD_OPTION)) {
257             try {
258                 downloadCurrentObservation();
259             } catch (IOException e) {
260                 //TODO: show error message
261                 LOGGER.error("Failed to download the observation.", e);
262             }
263         }
264         if (selectedSaveOptions.contains(MAIL_OPTION)) {
265             mailCurrentObservation();
266         }
267         if (selectedSaveOptions.contains(SAVE_OPTION)) {
268             saveCurrentObservation();
269         }
270     }
271 
272     /**
273      * Gets the timeline model.
274      */
275     public TimelineModel getTimeline() {
276         return timeline;
277     }
```

```
278
279     /**
280      * Gets the minimum date of the timeline. The user cannot move the timeline
281      * before that date.
282      */
283     public Date getMin() {
284        return min;
285     }
286
287     /**
288      * Gets the maximum date of the timeline. The user cannot move the timeline
289      * after that date.
290      */
291     public Date getMax() {
292        return max;
293     }
294
295     /**
296      *
297      */
298     public Date getDuration() {
299        return duration;
300     }
301
302     /**
303      * Gets the start date of the timeline.
304      */
305     public Date getStart() {
306        return start;
307     }
308
309     /**
310      * Gets the minimum zoom interval for the timeline in milliseconds.
311      */
312     public long getZoomMin() {
313        return zoomMin;
314     }
315
316     /**
317      * Gets the maximum zoom interval for the timeline in milliseconds.
318      */
319     public long getZoomMax() {
320        return zoomMax;
321     }
322
323     /**
324      *
325      */
326     public String getRecipientEmail() {
327        return recipientEmail;
328     }
329
330     /**
331      *
332      */
333     public void setRecipientEmail(String recipientEmail) {
334        this.recipientEmail = recipientEmail;
335     }
```

```
336
337     /**
338      *
339      */
340     public ObservationEntity getObservation() {
341        return observation;
342     }
343
344     /**
345      *
346      */
347     public void setObservation(ObservationEntity observation) {
348        this.observation = observation;
349     }
350
351     /**
352      *
353      */
354     public List<String> getSelectedSaveOptions() {
355        return selectedSaveOptions;
356     }
357
358     /**
359      *
360      */
361     public void setSelectedSaveOptions(List<String> selectedSaveOptions) {
362        this.selectedSaveOptions = selectedSaveOptions;
363     }
364
365     /**
366      *
367      */
368     public boolean getMailOptionChecked() {
369        return selectedSaveOptions.contains(MAIL_OPTION);
370     }
371
372     /**
373      *
374      */
375     public boolean isObservationSaved() {
376        return observationSaved;
377     }
378
379     /**
380      *
381      */
382     public void setObservationSaved(boolean observationSaved) {
383        this.observationSaved = observationSaved;
384     }
385 }
```

1.63 managedbeans/SuperUserManagedBean.java

```
1 package com.moveatis.managedbeans;
2
```

```
3  import com.moveatis.interfaces.Role;
4  import com.moveatis.roles.SuperUserRoleEntity;
5  import com.moveatis.user.IdentifiedUserEntity;
6  import javax.inject.Named;
7  import java.util.Date;
8  import java.util.List;
9  import javax.enterprise.context.RequestScoped;
10 import javax.inject.Inject;
11
12 /**
13  * The bean sets the superuser role for the identified users. It is not used
14  * in the current version of Moveatis.
15  * @author Sami Kallio <phinaliumz at outlook.com
16  */
17 @Named(value = "superUserBean")
18 @RequestScoped
19 public class SuperUserManagedBean {
20
21    @Inject
22    private Role roleBean;
23
24    public SuperUserManagedBean() {
25
26    }
27
28    /**
29     * Adds the superuser role to the given identified user.
30     */
31    public void addSuperUserRights(IdentifiedUserEntity userEntity) {
32       roleBean.addSuperuserRoleToUser(userEntity);
33    }
34
35    /**
36     * Adds the superuser role for the given time period to the given identified
          user.
37     */
38    public void addSuperUserRights(IdentifiedUserEntity userEntity, Date startDate,
          Date endDate) {
39       roleBean.addSuperuserRoleToUser(userEntity, startDate, endDate);
40    }
41
42    /**
43     * Removes the superuser role from the given identified user.
44     */
45    public void removeSuperUserRights(IdentifiedUserEntity userEntity) {
46       roleBean.removeSuperuserRoleFromUser(userEntity);
47    }
48
49    public List<SuperUserRoleEntity> listSuperUsers() {
50       return roleBean.listSuperusers();
51    }
52
53 }
```

1.64 managedbeans/TimeManagedBean.java

```java
package com.moveatis.managedbeans;

import com.moveatis.interfaces.Session;
import com.moveatis.timezone.TimeZoneInformation;
import java.util.TimeZone;
import javax.enterprise.context.RequestScoped;
import javax.inject.Inject;
import javax.inject.Named;

/**
 * Provides time related attributes like time zone.
 *
 * @author Juha Moisio
 */
@Named(value = "timeBean")
@RequestScoped
public class TimeManagedBean {

   @Inject
   private Session sessionBean;

   public TimeZone getUserTimeZone() {
      return sessionBean.getSessionTimeZone();
   }

   public TimeZone getServerTimeZone() {
      return TimeZoneInformation.getTimeZone();
   }
}
```

1.65 managedbeans/UserManagedBean.java

```java
package com.moveatis.managedbeans;

import javax.inject.Named;
import javax.enterprise.context.SessionScoped;
import java.io.Serializable;
import java.util.Locale;
import java.util.TimeZone;
import javax.faces.context.FacesContext;
import javax.faces.event.ActionEvent;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * The bean manages user-related information in a session.
 * @author Sami Kallio <phinaliumz at outlook.com>
 */
@Named(value = "userManagedBean")
@SessionScoped
public class UserManagedBean implements Serializable {

   private static final long serialVersionUID = 1L;
   private static final Logger LOGGER = LoggerFactory.getLogger(UserManagedBean.
      class);
```

```
23
24      private Locale locale;
25      private TimeZone timeZone;
26      private String languageString;
27      private String optionLanguageString;
28
29      /**
30       * Creates a new instance of UserManagedBean.
31       */
32      public UserManagedBean() {
33
34      }
35
36      /**
37       * Gets the locale to use in the user interface of the application.
38       */
39      public Locale getLocale() {
40
41         if(this.locale == null) {
42            FacesContext context = FacesContext.getCurrentInstance();
43            this.locale = context.getViewRoot().getLocale();
44            this.languageString = this.locale.getLanguage();
45         }
46         return this.locale;
47      }
48
49      public void setLocale(Locale locale) {
50         this.locale = locale;
51         FacesContext context = FacesContext.getCurrentInstance();
52         context.getViewRoot().setLocale(this.locale);
53
54         this.languageString = this.locale.getLanguage();
55      }
56
57      public void setLocale(String language) {
58
59         Locale newLocale = null;
60
61         if(language.equalsIgnoreCase("fi")) {
62            newLocale = new Locale("fi", "FI");
63         } else if(language.equalsIgnoreCase("en")) {
64            newLocale = new Locale("en");
65         }
66
67         if(newLocale != null) {
68            this.setLocale(newLocale);
69         }
70      }
71
72      public void changeLocale(ActionEvent event) {
73         Locale finnishLocale = new Locale("fi", "FI");
74         Locale defaultLocale = new Locale("en");
75
76         if(this.locale.getLanguage().equals(finnishLocale.getLanguage())) {
77            this.setLocale(defaultLocale);
78            this.setLanguageString(defaultLocale.getLanguage());
79            this.setOptionLanguageString(finnishLocale.getLanguage());
80         } else {
```

```
81            this.setLocale(finnishLocale);
82            this.setLanguageString(finnishLocale.getLanguage());
83            this.setOptionLanguageString(defaultLocale.getLanguage());
84         }
85
86      }
87
88      public String getLanguageString() {
89         return languageString;
90      }
91
92      public void setLanguageString(String languageString) {
93         this.languageString = languageString;
94      }
95
96      public TimeZone getTimeZone() {
97         return timeZone;
98      }
99
100     public void setTimeZone(TimeZone timeZone) {
101        this.timeZone = timeZone;
102     }
103
104     public String getOptionLanguageString() {
105        Locale finnishLocale = new Locale("fi", "FI");
106
107        if(this.languageString == null) {
108           this.getLocale();
109        }
110
111        if(this.languageString.equals(finnishLocale.getLanguage())) {
112           this.optionLanguageString = "English";
113        } else {
114           this.optionLanguageString = "Suomi";
115        }
116
117        return optionLanguageString;
118     }
119
120     public void setOptionLanguageString(String optionLanguageString) {
121        this.optionLanguageString = optionLanguageString;
122     }
123  }
```

1.66 managedbeans/ValidationManagedBean.java

```
1  package com.moveatis.managedbeans;
2
3  import com.moveatis.helpers.Validation;
4  import com.moveatis.interfaces.GroupKey;
5  import com.moveatis.interfaces.MessageBundle;
6  import java.text.MessageFormat;
7  import java.util.ResourceBundle;
8  import javax.inject.Named;
9  import javax.enterprise.context.RequestScoped;
```

```
10 import javax.faces.application.FacesMessage;
11 import javax.faces.component.UIComponent;
12 import javax.faces.context.FacesContext;
13 import javax.faces.validator.ValidatorException;
14 import javax.inject.Inject;
15
16 /**
17  * The bean implements commonly used methods to validate user input.
18  * @author Ilari Paananen
19  */
20 @Named(value = "validationBean")
21 @RequestScoped
22 public class ValidationManagedBean {
23
24    @Inject @MessageBundle
25    private transient ResourceBundle messages;
26
27    @Inject
28    private GroupKey groupKeyEJB;
29
30    private void throwError(String message) {
31       throw new ValidatorException(new FacesMessage(FacesMessage.SEVERITY_ERROR,
             messages.getString("dialogErrorTitle"), message));
32    }
33
34    public void validateStringForJsAndHtml(FacesContext context, UIComponent
          component, Object value) {
35       String s = (String)value;
36       String valid = Validation.validateForJsAndHtml(s);
37       if (!s.equals(valid)) {
38 // String error = MessageFormat.format(messages.getString("validate_invalidChars")
     , invalidChars);
39          String error = messages.getString("validate_invalidChars");
40          throwError(error);
41       }
42    }
43
44    public void validateGroupKey(FacesContext context, UIComponent component,
          Object value) {
45       validateStringForJsAndHtml(context, component, value);
46       validateStringMinLength((String) value, 4);
47       validateStringMaxLength((String) value, 64);
48       if(groupKeyEJB.findByKey((String) value) != null) {
49          String error = messages.getString("validate_groupKeyReserved");
50          throwError(error);
51       }
52    }
53
54    public void validateShortString(FacesContext context, UIComponent component,
          Object value) {
55       validateStringForJsAndHtml(context, component, value);
56       validateStringMaxLength((String) value, 64);
57    }
58
59    public void validateLongString(FacesContext context, UIComponent component,
          Object value) {
60       validateStringForJsAndHtml(context, component, value);
61       validateStringMaxLength((String) value, 256);
```

```
62     }
63
64     private void validateStringMaxLength(String str, int maxLength) {
65        if (str.length() > maxLength) {
66           String error = MessageFormat.format(messages.getString("
                 validate_maxLength"), maxLength);
67           throwError(error);
68        }
69     }
70
71     private void validateStringMinLength(String str, int minLength) {
72        if (str.length() < minLength) {
73           String error = MessageFormat.format(messages.getString("
                 validate_minLength"), minLength);
74           throwError(error);
75        }
76     }
77
78  }
```

1.67 observation/ObservationBean.java

```
1  package com.moveatis.observation;
2
3  import com.moveatis.abstracts.AbstractBean;
4  import com.moveatis.interfaces.Event;
5  import javax.ejb.Stateful;
6  import javax.persistence.EntityManager;
7  import javax.persistence.PersistenceContext;
8  import com.moveatis.interfaces.Observation;
9  import com.moveatis.records.RecordEntity;
10 import java.io.Serializable;
11 import javax.ejb.EJB;
12 import javax.inject.Inject;
13 import org.slf4j.Logger;
14 import org.slf4j.LoggerFactory;
15 import com.moveatis.session.SessionBean;
16 import com.moveatis.user.AbstractUser;
17 import java.util.ArrayList;
18 import java.util.List;
19 import javax.persistence.TypedQuery;
20
21 /**
22  * The EJB manages observations..
23  *
24  * @author Sami Kallio <phinaliumz at outlook.com>
25  */
26 @Stateful
27 public class ObservationBean extends AbstractBean<ObservationEntity> implements
       Observation, Serializable {
28
29    private static final Logger LOGGER = LoggerFactory.getLogger(ObservationBean.
          class);
30
31    private static final long serialVersionUID = 1L;
```

```
32
33      @Inject
34      private SessionBean sessionBean;
35
36      @EJB
37      private Event eventEJB;
38
39      private ObservationEntity observationEntity;
40
41      @PersistenceContext(unitName = "MOVEATIS_PERSISTENCE")
42      private EntityManager em;
43
44      @Override
45      protected EntityManager getEntityManager() {
46          return em;
47      }
48
49      public ObservationBean() {
50          super(ObservationEntity.class);
51      }
52
53      /**
54       * Finds and returns all observations for the specific user.
55       * @param observer The user, whose observations are to be searched.
56       * @return A list of the observations for the user.
57       */
58      @Override
59      public List<ObservationEntity> findAllByObserver(AbstractUser observer) {
60          TypedQuery<ObservationEntity> query = em.createNamedQuery("findByObserver",
                ObservationEntity.class);
61          query.setParameter("observer", observer);
62          return query.getResultList();
63      }
64
65      /**
66       * Finds the observations for the user, which have no event attached to them.
67       * @param observer The user, whose observations are to be searched.
68       * @return A list of the observations.
69       */
70      @Override
71      public List<ObservationEntity> findWithoutEvent(AbstractUser observer) {
72          TypedQuery<ObservationEntity> query = em.createNamedQuery("findWithoutEvent"
                , ObservationEntity.class);
73          query.setParameter("observer", observer);
74          return query.getResultList();
75      }
76
77      /**
78       * Finds the observations that are made for events that the specified
79       * user does not own.
80       * @param observer The user, whose observations are to be searched.
81       * @return A list of the observations.
82       */
83      @Override
84      public List<ObservationEntity> findByEventsNotOwned(AbstractUser observer) {
85          TypedQuery<ObservationEntity> query = em.createNamedQuery("
                findByEventsNotOwned", ObservationEntity.class);
86          query.setParameter("observer", observer);
```

```
87          return query.getResultList();
88      }
89
90      /**
91       * Persists the observations to the database.
92       * @param observationEntity The observatio entity to be persisted.
93       */
94      @Override
95      public void create(ObservationEntity observationEntity) {
96          super.create(observationEntity);
97      }
98
99      /**
100      * Finds a list of the records for the observation with the given id.
101      * @param id The id of the observation.
102      * @return A list of the records.
103      */
104     @Override
105     public List<RecordEntity> findRecords(Object id) {
106         observationEntity = em.find(ObservationEntity.class, id);
107         if(observationEntity != null) {
108             return observationEntity.getRecords();
109         }
110         return new ArrayList<>(); //return empty list
111     }
112
113     /**
114      * Removes the observation and also removes the observation from the event
115      * it was associated with.
116      * @param observationEntity The observation to be removed.
117      */
118     @Override
119     public void remove(ObservationEntity observationEntity) {
120         super.remove(observationEntity);
121         eventEJB.removeObservation(observationEntity);
122         observationEntity.setEvent(null);
123         super.edit(observationEntity);
124     }
125
126     /**
127      * Permanently removes the observation, which the user did not set to be
128      * saved into the database.
129      * @param observationEntity The observation to be removed.
130      */
131     @Override
132     public void removeUnsavedObservation(ObservationEntity observationEntity) {
133         em.remove(em.merge(observationEntity));
134     }
135 }
```

1.68 observation/ObservationCategory.java

```
1
2 package com.moveatis.observation;
3
```

```
4  import com.moveatis.category.CategoryType;
5  import com.moveatis.helpers.Validation;
6  import java.io.Serializable;
7
8  /**
9   * The observation has its own categories, so renaming the original category
10  * does not affect on old observations.
11  *
12  * @author Sami Kallio <phinaliumz at outlook.com>
13  * @author Ilari Paananen <ilari.k.paananen at student.jyu.fi>
14  */
15 public class ObservationCategory implements Serializable {
16
17    private static final long serialVersionUID = 1L;
18
19    private CategoryType type;
20    private Long tag;
21    private String name;
22
23    public ObservationCategory() {
24       this.type = CategoryType.TIMED;
25       this.name = "";
26    }
27
28    public ObservationCategory(ObservationCategory other) {
29       this.type = other.type;
30       this.name = other.name;
31       this.tag = other.tag;
32    }
33
34    public CategoryType getType() {
35       return type;
36    }
37
38    public void setType(CategoryType type) {
39       this.type = type;
40    }
41
42    public int getTypeAsInt() {
43       return type.ordinal();
44    }
45
46    /**
47     * Returns true if the type of the category is COUNTED.
48     * The method is used with the PrimeFaces boolean button.
49     */
50    public boolean getTypeAsBoolean() {
51       return (type == CategoryType.COUNTED);
52    }
53
54    /**
55     * Sets the type of the category based on the given boolean value.
56     * If the value is true, the type will be COUNTED. Otherwise the type will be
          TIMED.
57     * The method is used with the PrimeFaces boolean button.
58     */
59    public void setTypeAsBoolean(boolean value) {
60       if (value) {
```

```
61            type = CategoryType.COUNTED;
62         }
63         else {
64            type = CategoryType.TIMED;
65         }
66      }
67
68      public Long getTag() {
69         return tag;
70      }
71
72      public void setTag(Long tag) {
73         this.tag = tag;
74      }
75
76      public String getName() {
77          return name;
78      }
79
80      public final void setName(String name) {
81         String validName = Validation.validateForJsAndHtml(name).trim();
82         if (!this.name.equals(validName)) {
83            this.name = validName;
84         }
85      }
86   }
```

1.69 observation/ObservationCategorySet.java

```
1
2  package com.moveatis.observation;
3
4  import com.moveatis.category.CategoryType;
5  import java.io.Serializable;
6  import java.util.ArrayList;
7  import java.util.List;
8
9  /**
10  * The observation has its own category sets, so renaming or removing
11  * original category sets does not alter old observations.
12  *
13  * @author Sami Kallio <phinaliumz at outlook.com>
14  * @author Ilari Paananen <ilari.k.paananen at student.jyu.fi>
15  */
16 public class ObservationCategorySet implements Serializable {
17
18    private static final long serialVersionUID = 1L;
19
20    private final Long id;
21    private final String name;
22    private final List<ObservationCategory> categories;
23
24    /**
25     * Creates a new instance of ObservationCategorySet with the given id and name.
26     */
```

```
27      public ObservationCategorySet(Long id, String name) {
28          this.id = id;
29          this.name = name;
30          this.categories = new ArrayList<>();
31      }
32
33      public Long getId() {
34          return id;
35      }
36
37      public String getName() {
38          return name;
39      }
40
41      /**
42       * Returns the list of the categories in the category set.
43       */
44      public List<ObservationCategory> getCategories() {
45          return categories;
46      }
47
48      /**
49       * Adds a new category to the list of the categories.
50       * @param type The type of the new category.
51       * @param tag The tag of the new category.
52       * @param name The name of the new category.
53       */
54      public void add(CategoryType type, Long tag, String name) {
55          ObservationCategory category = new ObservationCategory();
56          category.setType(type);
57          category.setName(name);
58          category.setTag(tag);
59          categories.add(category);
60      }
61
62      /**
63       * Adds the given category to the list of the categories.
64       * @param category The category to be added.
65       */
66      public void add(ObservationCategory category) {
67          categories.add(category);
68      }
69
70      /**
71       * Adds an empty category to the list of the categories.
72       */
73      public void addEmpty() {
74          ObservationCategory category = new ObservationCategory();
75          category.setTag(-1L);
76          categories.add(category);
77      }
78
79      /**
80       * Removes the given category from the list of the categories.
81       * @param category The category to be removed.
82       */
83      public void remove(ObservationCategory category) {
84          categories.remove(category);
```

```
85      }
86  }
```

1.70 observation/ObservationCategorySetList.java

```
1
2   package com.moveatis.observation;
3
4   import java.util.ArrayList;
5   import java.util.List;
6
7   /**
8    * The observation has its own category sets, and this class has a list, which
9    * holds those category sets.
10   *
11   * @author Sami Kallio <phinaliumz at outlook.com>
12   * @author Ilari Paananen <ilari.k.paananen at student.jyu.fi>
13   */
14  public class ObservationCategorySetList {
15
16      private List<ObservationCategorySet> categorySets = new ArrayList<>();
17
18      public ObservationCategorySetList() {
19
20      }
21
22      public List<ObservationCategorySet> getCategorySets() {
23          return categorySets;
24      }
25
26      public void setCategorySets(List<ObservationCategorySet> categorySets) {
27          this.categorySets = categorySets;
28      }
29
30      /**
31       * Adds the given category set to the category set list.
32       */
33      public void add(ObservationCategorySet categorySet) {
34          categorySets.add(categorySet);
35      }
36
37      /**
38       * Creates a clone of the given category set and adds it to the category set
             list.
39       */
40      public void addClone(ObservationCategorySet categorySet) {
41          ObservationCategorySet cloned = new ObservationCategorySet(categorySet.getId
                 (), categorySet.getName());
42          for (ObservationCategory category : categorySet.getCategories()) {
43              cloned.add(new ObservationCategory(category));
44          }
45          categorySets.add(cloned);
46      }
47
48      /**
```

```
49      * Searches for the category set with the given id from the category set list.
50      * Returns the category if it was found and null otherwise.
51      */
52     public ObservationCategorySet find(Long id) {
53        for (ObservationCategorySet categorySet : categorySets) {
54           if (categorySet.getId().equals(id)) {
55              return categorySet;
56           }
57        }
58        return null;
59     }
60
61     /**
62      * Removes the given category set from the category set list.
63      */
64     public void remove(ObservationCategorySet categorySet) {
65        categorySets.remove(categorySet);
66     }
67
68  }
```

1.71 observation/ObservationEntity.java

```
1  package com.moveatis.observation;
2
3  import com.moveatis.abstracts.BaseEntity;
4  import com.moveatis.records.RecordEntity;
5  import com.moveatis.event.EventEntity;
6  import com.moveatis.user.AbstractUser;
7  import java.io.Serializable;
8  import java.util.ArrayList;
9  import java.util.List;
10 import java.util.Set;
11 import static javax.persistence.CascadeType.ALL;
12 import javax.persistence.ElementCollection;
13 import javax.persistence.Entity;
14 import javax.persistence.FetchType;
15 import javax.persistence.ManyToOne;
16 import javax.persistence.NamedQueries;
17 import javax.persistence.NamedQuery;
18 import javax.persistence.OneToMany;
19 import javax.persistence.Table;
20
21 /**
22  * The entity represents the observation data, that is saved to the database.
23  * @author Sami Kallio <phinaliumz at outlook.com>
24  */
25 @Table(name = "OBSERVATION")
26 @NamedQueries({
27    @NamedQuery(
28          name = "findByObserver",
29          query = "SELECT observation FROM ObservationEntity observation WHERE
                observation.observer=:observer"
30    ),
31    @NamedQuery(
```

```
32          name = "findWithoutEvent",
33          query = "SELECT observation FROM ObservationEntity observation WHERE
                observation.observer=:observer AND observation.event is null"
34      ),
35      @NamedQuery(
36          name = "findByEventsNotOwned",
37          query = "SELECT observation FROM ObservationEntity observation WHERE
                observation.observer=:observer AND observation.event.creator!=:
                observer"
38      )
39  })
40  @Entity
41  public class ObservationEntity extends BaseEntity implements Serializable {
42
43      private static final long serialVersionUID = 1L;
44
45      @ManyToOne
46      private AbstractUser observer;
47
48      @ManyToOne
49      private EventEntity event;
50
51      @ElementCollection(fetch = FetchType.LAZY)
52      private Set<ObservationCategorySet> observationCategorySets;
53
54      @OneToMany(mappedBy = "observation", fetch = FetchType.LAZY, cascade = ALL)
55      private List<RecordEntity> records;
56
57      private long duration;
58
59      private String description;
60      private String name;
61      private String target;
62      private Boolean userWantsToSaveToDatabase;
63
64      @Override
65      public Long getId() {
66          return id;
67      }
68
69      @Override
70      public void setId(Long id) {
71          this.id = id;
72      }
73
74      public List<RecordEntity> getRecords() {
75          return records;
76      }
77
78      public AbstractUser getObserver() {
79          return observer;
80      }
81
82      public void setObserver(AbstractUser observer) {
83          this.observer = observer;
84      }
85
86      public EventEntity getEvent() {
```

```
87          return event;
88      }
89
90      public void setEvent(EventEntity event) {
91          this.event = event;
92      }
93
94      public void setRecords(List<RecordEntity> records) {
95          this.records = records;
96      }
97
98      public void addRecord(RecordEntity record) {
99          if (this.getRecords() == null) {
100             this.records = new ArrayList<>();
101         }
102         getRecords().add(record);
103         record.setObservation(this);
104     }
105
106     public long getDuration() {
107         return duration;
108     }
109
110     public void setDuration(long duration) {
111         this.duration = duration;
112     }
113
114     public String getDescription() {
115         return description;
116     }
117
118     public void setDescription(String description) {
119         this.description = description;
120     }
121
122     public String getName() {
123         return name;
124     }
125
126     public void setName(String name) {
127         this.name = name;
128     }
129
130     public String getTarget() {
131         return target;
132     }
133
134     public void setTarget(String target) {
135         this.target = target;
136     }
137
138     public Set<ObservationCategorySet> getObservationCategorySets() {
139         return observationCategorySets;
140     }
141
142     public void setObservationCategorySets(Set<ObservationCategorySet>
            observationCategorySets) {
143         this.observationCategorySets = observationCategorySets;
```

```
144     }
145
146     public Boolean getUserWantsToSaveToDatabase() {
147         return userWantsToSaveToDatabase;
148     }
149
150     public void setUserWantsToSaveToDatabase(Boolean userWantsToSaveToDatabase) {
151         this.userWantsToSaveToDatabase = userWantsToSaveToDatabase;
152     }
153
154     @Override
155     public int hashCode() {
156         int hash = 0;
157         hash += (id != null ? id.hashCode() : 0);
158         return hash;
159     }
160
161     @Override
162     public boolean equals(Object object) {
163         if (!(object instanceof ObservationEntity)) {
164             return false;
165         }
166         ObservationEntity other = (ObservationEntity) object;
167         return !((this.id == null && other.id != null) || (this.id != null && !this.
                id.equals(other.id)));
168     }
169
170     @Override
171     public String toString() {
172         return "com.moveatis.observation.ObservationEntity[ id=" + id + " ]";
173     }
174
175 }
```

1.72 providers/MessageProvider.java

```
1
2  package com.moveatis.providers;
3
4  import com.moveatis.interfaces.MessageBundle;
5  import java.util.ResourceBundle;
6  import javax.enterprise.context.RequestScoped;
7  import javax.enterprise.inject.Produces;
8  import javax.faces.context.FacesContext;
9
10 /**
11  * The CDI bean produces a resource bundle for the transient CDI beans.
12  * @author Sami Kallio <phinaliumz at outlook.>
13  */
14 @RequestScoped
15 public class MessageProvider {
16
17     private ResourceBundle resourceBundle;
18
19     @Produces @MessageBundle
```

```
20     public ResourceBundle getBundle() {
21
22        if(resourceBundle == null) {
23           FacesContext context = FacesContext.getCurrentInstance();
24           resourceBundle = context.getApplication().getResourceBundle(context, "msg
                 ");
25        }
26
27        return resourceBundle;
28     }
29  }
```

1.73 records/RecordBean.java

```
1  package com.moveatis.records;
2
3  import com.moveatis.abstracts.AbstractBean;
4  import com.moveatis.interfaces.Record;
5  import javax.ejb.Stateless;
6  import javax.persistence.EntityManager;
7  import javax.persistence.PersistenceContext;
8
9  /**
10  * The EJB for managing records. Holds the information for the analysis of an
       observation.
11  * @author Sami Kallio <phinaliumz at outlook.com>
12  */
13 @Stateless
14 public class RecordBean extends AbstractBean<RecordEntity> implements Record {
15
16    @PersistenceContext(unitName = "MOVEATIS_PERSISTENCE")
17    private EntityManager em;
18
19    @Override
20    protected EntityManager getEntityManager() {
21       return em;
22    }
23
24    public RecordBean() {
25       super(RecordEntity.class);
26    }
27
28 }
```

1.74 records/RecordEntity.java

```
1  package com.moveatis.records;
2
3  import com.moveatis.abstracts.BaseEntity;
4  import com.moveatis.observation.ObservationCategory;
5  import com.moveatis.observation.ObservationEntity;
6  import java.io.File;
7  import java.io.Serializable;
```

```
8  import javax.persistence.Entity;
9  import javax.persistence.ManyToOne;
10 import javax.persistence.Table;
11
12 /**
13  * The entity represent the data of a record, which will be persisted to the
        database.
14  *
15  * @author Sami Kallio <phinaliumz at outlook.com>
16  */
17 @Entity
18 @Table(name="RECORD")
19 public class RecordEntity extends BaseEntity implements Serializable {
20
21    private static final long serialVersionUID = 1L;
22
23    private ObservationCategory category;
24
25    private Long startTime;
26    private Long endTime;
27
28    @ManyToOne
29    private ObservationEntity observation;
30
31    private String comment;
32    // Not used in version 1.0
33    private File voiceComment;
34
35    public ObservationEntity getObservation() {
36       return observation;
37    }
38
39    public void setObservation(ObservationEntity observation) {
40       this.observation = observation;
41    }
42
43    public ObservationCategory getCategory() {
44       return category;
45    }
46
47    public void setObservationCategory(ObservationCategory category) {
48       this.category = category;
49    }
50
51    public Long getStartTime() {
52       return startTime;
53    }
54
55    public void setStartTime(Long startTime) {
56       this.startTime = startTime;
57    }
58
59    public Long getEndTime() {
60       return endTime;
61    }
62
63    public void setEndTime(Long endTime) {
64       this.endTime = endTime;
```

```
65      }
66
67      public String getComment() {
68          return comment;
69      }
70
71      public void setComment(String comment) {
72          this.comment = comment;
73      }
74
75      /**
76       * Not used in version 1.0.
77       * @return The file for the voice comment.
78       */
79      public File getVoiceComment() {
80          return voiceComment;
81      }
82
83      /**
84       * Not used in version 1.0.
85       * @param voiceComment The file that holds the voice comment.
86       */
87      public void setVoiceComment(File voiceComment) {
88          this.voiceComment = voiceComment;
89      }
90
91      @Override
92      public int hashCode() {
93          int hash = 0;
94          hash += (id != null ? id.hashCode() : 0);
95          return hash;
96      }
97
98      @Override
99      public boolean equals(Object object) {
100         if (!(object instanceof RecordEntity)) {
101             return false;
102         }
103         RecordEntity other = (RecordEntity) object;
104         return !((this.id == null && other.id != null) || (this.id != null && !this.
                id.equals(other.id)));
105     }
106
107     @Override
108     public String toString() {
109         return "com.moveatis.records.RecordEntity[ id=" + id + " ]";
110     }
111
112 }
```

1.75 restful/MoveatisRestApplication.java

```
1  package com.moveatis.restful;
2
3  import java.util.HashSet;
```

```
4  import java.util.Set;
5  import javax.ws.rs.ApplicationPath;
6  import javax.ws.rs.core.Application;
7
8  /**
9   * RestApplication for RESTfull actions in Moveatis. Includes actions for
10  * adding records to an observation as well as starting and stopping an observation
        .
11  * @author Sami Kallio <phinaliumz at outlook.com>
12  */
13 @ApplicationPath("/webapi")
14 public class MoveatisRestApplication extends Application {
15
16    public MoveatisRestApplication() {
17
18    }
19
20    @Override
21    public Set<Class<?>> getClasses() {
22       final Set<Class<?>> classes = new HashSet<>();
23       classes.add(RecordListenerBean.class);
24       classes.add(NotFoundExceptionMapper.class);
25
26       return classes;
27    }
28 }
```

1.76 restful/NotFoundExceptionMapper.java

```
1  package com.moveatis.restful;
2
3  import javax.ws.rs.NotFoundException;
4  import javax.ws.rs.core.MediaType;
5  import javax.ws.rs.core.Response;
6  import javax.ws.rs.core.Response.Status;
7  import javax.ws.rs.ext.ExceptionMapper;
8  import javax.ws.rs.ext.Provider;
9
10 /**
11  * Custom NotFound exception for those REST API calls, that are not to
12  * /webapi or RecordListenerBean.
13  * @author Sami Kallio <phinaliumz at outlook.com>
14  */
15 @Provider
16 public class NotFoundExceptionMapper implements ExceptionMapper<NotFoundException>
        {
17
18    public NotFoundExceptionMapper() {
19
20    }
21
22    @Override
23    public Response toResponse(NotFoundException exception) {
24       return Response
25             .status(Status.NOT_FOUND)
```

```
            .entity("HTTP 404 - Not found")
            .type(MediaType.TEXT_PLAIN)
            .build();
    }
}
```

1.77 restful/RecordListenerBean.java

```
package com.moveatis.restful;

import com.moveatis.interfaces.Category;
import com.moveatis.interfaces.Label;
import com.moveatis.interfaces.Observation;
import com.moveatis.interfaces.Record;
import com.moveatis.interfaces.Session;
import com.moveatis.managedbeans.ObservationManagedBean;
import com.moveatis.managedbeans.UserManagedBean;
import com.moveatis.observation.ObservationCategory;
import com.moveatis.observation.ObservationCategorySet;
import com.moveatis.records.RecordEntity;
import com.moveatis.timezone.TimeZoneInformation;
import java.io.Serializable;
import java.io.StringReader;
import java.text.DateFormat;
import java.util.Date;
import java.util.HashMap;
import java.util.Locale;
import java.util.Map;
import java.util.ResourceBundle;
import java.util.TimeZone;
import javax.ejb.Stateful;
import javax.inject.Inject;
import javax.inject.Named;
import javax.json.Json;
import javax.json.JsonArray;
import javax.json.JsonNumber;
import javax.json.JsonObject;
import javax.json.JsonReader;
import javax.servlet.http.HttpServletRequest;
import javax.ws.rs.Consumes;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.MediaType;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * The bean manages REST API for adding records to an observation as well as
 * starting and stopping an observation.
 * @author Sami Kallio <phinaliumz at outlook.com>
 */
@Path("/records")
@Named(value = "recordBean")
```

```
48 @Stateful
49 public class RecordListenerBean implements Serializable {
50
51     private static final Logger LOGGER = LoggerFactory.getLogger(RecordListenerBean
           .class);
52     private static final long serialVersionUID = 1L;
53
54     private JsonReader jsonReader;
55
56     @Context
57     private HttpServletRequest httpRequest;
58
59     @Inject
60     private Session sessionBean;
61     @Inject
62     private ObservationManagedBean observationManagedBean;
63     @Inject
64     private UserManagedBean userManagedBean;
65
66     private ResourceBundle messages;
67
68     @Inject
69     private Observation observationEJB;
70     @Inject
71     private Record recordEJB;
72     @Inject
73     private Category categoryEJB;
74     @Inject
75     private Label labelEJB;
76
77     public RecordListenerBean() {
78
79     }
80
81     /**
82      * Marks the observation as started.
83      */
84     @POST
85     @Path("startobservation")
86     @Consumes(MediaType.TEXT_PLAIN)
87     @Produces(MediaType.TEXT_PLAIN)
88     public String startObservation(String data) {
89        observationManagedBean.startObservation();
90        return "success";
91     }
92
93     /**
94      * Keeps the session alive during the observation.
95      */
96     /*
97     * TODO: Needs work - what to do when keep-alive request is commenced?
98     */
99     // NOTE: Used by observer view.
100    // TODO: What about other views? What happens if session expires? Redirect to
           front page?
101    @POST
102    @Path("keepalive")
103    @Produces(MediaType.TEXT_PLAIN)
```

```
104     public String keepAlive() {
105        return "keep-alive";
106     }
107
108     /**
109      * Adds the observation data coming from the observation view.
110      * It could be used to take data from other clients, but not yet
111      * implemented or even planned.
112      * @param data The JSON data containing the records of the observation.
113      * @return "success" if the action succeeded and "failed" if it failed.
114      */
115     @POST
116     @Path("addobservationdata")
117     @Consumes(MediaType.APPLICATION_JSON)
118     @Produces(MediaType.TEXT_PLAIN)
119     public String addObservationData(String data) {
120        Locale locale = userManagedBean.getLocale();
121        messages = ResourceBundle.getBundle("com.moveatis.messages.Messages", locale
               );
122        StringReader stringReader = new StringReader(data);
123
124        jsonReader = Json.createReader(stringReader);
125
126        JsonObject jObject = jsonReader.readObject();
127        JsonNumber duration = jObject.getJsonNumber("duration");
128        JsonNumber timeZoneOffset = jObject.getJsonNumber("timeZoneOffsetInMs");
129        JsonNumber DSTOffset = jObject.getJsonNumber("daylightSavingInMs");
130        JsonArray array = jObject.getJsonArray("data");
131
132        jsonReader.close();
133
134        Date createdTime = observationManagedBean.getObservationEntity().getCreated
               ();
135
136        TimeZone timeZone = TimeZoneInformation.getTimeZoneFromOffset(
137              timeZoneOffset.intValue(), DSTOffset.intValue());
138
139        sessionBean.setSessionTimeZone(timeZone);
140
141        DateFormat dateFormat = DateFormat.getDateTimeInstance(DateFormat.SHORT,
142              DateFormat.SHORT, locale);
143        dateFormat.setTimeZone(timeZone);
144
145        observationManagedBean.setObservationName(messages.getString("obs_title")
146              + " - " + dateFormat.format(createdTime));
147        observationManagedBean.setObservationDuration(duration.longValue());
148
149        Map<Long, ObservationCategory> categoriesById = new HashMap<>();
150
151        for (ObservationCategorySet categorySet : observationManagedBean.
               getCategorySetsInUse()) {
152           for (ObservationCategory category : categorySet.getCategories()) {
153              categoriesById.put(category.getTag(), category);
154           }
155        }
156
157        try {
158           for (int i = 0; i < array.size(); i++) {
```

```
159             JsonObject object = array.getJsonObject(i);
160             RecordEntity record = new RecordEntity();
161 
162             Long id = object.getJsonNumber("id").longValue();
163             ObservationCategory category = categoriesById.get(id);
164 
165             record.setObservationCategory(category);
166 
167             record.setStartTime(object.getJsonNumber("startTime").longValue());
168             record.setEndTime(object.getJsonNumber("endTime").longValue());
169 
170             observationManagedBean.addRecord(record);
171          }
172       } catch(Exception e) {
173          LOGGER.error("Parsing JSON to records failed", e);
174          return "failed";
175       }
176 
177       observationManagedBean.saveObservation();
178 
179       return "success";
180    }
181 }
```

1.78 roles/AbstractRole.java

```
1 
2  package com.moveatis.roles;
3 
4  import com.moveatis.abstracts.BaseEntity;
5  import com.moveatis.user.IdentifiedUserEntity;
6  import java.io.Serializable;
7  import java.util.Date;
8  import javax.persistence.DiscriminatorColumn;
9  import javax.persistence.Entity;
10 import javax.persistence.Inheritance;
11 import javax.persistence.InheritanceType;
12 import javax.persistence.Temporal;
13 
14 /**
15  * The entity is the base for the roles in Moveatis. The roles can be used to make
16  * the access rights system more finegrained. The entity can be extended to
17  * add more roles.
18  * @author Sami Kallio <phinaliumz at outlook.com>
19  */
20 @Entity
21 @Inheritance(strategy=InheritanceType.JOINED)
22 @DiscriminatorColumn(name="ROLE_TYPE")
23 public abstract class AbstractRole extends BaseEntity implements Serializable {
24 
25    @Temporal(javax.persistence.TemporalType.TIMESTAMP)
26    protected Date startDate;
27    @Temporal(javax.persistence.TemporalType.TIMESTAMP)
28    protected Date endDate;
29 
```

```
30     public Date getStartDate() {
31        return startDate;
32     }
33
34     public void setStartDate(Date startDate) {
35        this.startDate = startDate;
36     }
37
38     public Date getEndDate() {
39        return endDate;
40     }
41
42     public void setEndDate(Date endDate) {
43        this.endDate = endDate;
44     }
45
46     public abstract void setUserEntity(IdentifiedUserEntity user);
47     public abstract IdentifiedUserEntity getUserEntity();
48
49     @Override
50     public int hashCode() {
51        int hash = 0;
52        hash += (id != null ? id.hashCode() : 0);
53        return hash;
54     }
55
56     @Override
57     public boolean equals(Object object) {
58        if (!(object instanceof AbstractRole)) {
59           return false;
60        }
61        AbstractRole other = (AbstractRole) object;
62        return !((this.id == null && other.id != null) || (this.id != null && !this.
              id.equals(other.id)));
63     }
64
65     @Override
66     public String toString() {
67        return "com.moveatis.roles.AbstractRole[ id=" + id + " ]";
68     }
69
70  }
```

1.79 roles/RoleBean.java

```
1  package com.moveatis.roles;
2
3  import com.moveatis.abstracts.AbstractBean;
4  import com.moveatis.interfaces.Role;
5  import com.moveatis.user.IdentifiedUserEntity;
6  import java.util.Collections;
7  import java.util.Date;
8  import java.util.List;
9  import javax.ejb.Stateless;
10 import javax.persistence.EntityManager;
```

```
11 import javax.persistence.NoResultException;
12 import javax.persistence.PersistenceContext;
13 import javax.persistence.TypedQuery;
14 import org.slf4j.Logger;
15 import org.slf4j.LoggerFactory;
16
17 /**
18  * The EJB manages the roles, which can be added to the users to allow
19  * the access system to be more finegrained.
20  * @author Sami Kallio <phinaliumz at outlook.com>
21  */
22 @Stateless
23 public class RoleBean extends AbstractBean<AbstractRole> implements Role {
24
25    private static final Logger LOGGER = LoggerFactory.getLogger(RoleBean.class);
26
27    @PersistenceContext(unitName = "MOVEATIS_PERSISTENCE")
28    private EntityManager em;
29
30    public RoleBean() {
31       super(AbstractRole.class);
32    }
33
34    @Override
35    protected EntityManager getEntityManager() {
36       return em;
37    }
38
39    /**
40     * Adds the superuser rights to the user.
41     * @param user The user to whom the superuser rights are granted.
42     */
43    @Override
44    public void addSuperuserRoleToUser(IdentifiedUserEntity user) {
45       SuperUserRoleEntity role = new SuperUserRoleEntity();
46       addRoleToUser(role, user);
47    }
48
49    /**
50     * Adds the superuser rights to the user with the start and end date.
51     * @param user The user to whom the superuser rights are granted.
52     * @param startDate The date when te role is actived.
53     * @param endDate The date when the role is deactived.
54     */
55    @Override
56    public void addSuperuserRoleToUser(IdentifiedUserEntity user, Date startDate,
          Date endDate) {
57       SuperUserRoleEntity role = new SuperUserRoleEntity();
58       addRoleToUser(role, user, startDate, endDate);
59    }
60
61    /**
62     * Removes the superuser rights from the user.
63     * @param user The user whose superuser rights should be removed.
64     */
65    @Override
66    public void removeSuperuserRoleFromUser(IdentifiedUserEntity user) {
```

```
67          TypedQuery typedQuery = em.createNamedQuery("findSuperUserRoleByUser",
                SuperUserRoleEntity.class);
68          typedQuery.setParameter("userEntity", user);
69          SuperUserRoleEntity role = (SuperUserRoleEntity)typedQuery.getSingleResult()
                ;
70          if(role != null) {
71             removeRoleFromUser(role);
72          }
73       }
74
75       /**
76        * Finds and returns a list of the users with the superuser rights.
77        * @return A list of the users with the superuser rights.
78        */
79       @Override
80       public List<SuperUserRoleEntity> listSuperusers() {
81          SuperUserRoleEntity role = new SuperUserRoleEntity();
82          return (List<SuperUserRoleEntity>)listRoleUsers(role);
83       }
84
85       private void addRoleToUser(AbstractRole role, IdentifiedUserEntity user) {
86          role.setUserEntity(user);
87          super.create(role);
88       }
89
90       private void addRoleToUser(AbstractRole role, IdentifiedUserEntity user, Date
             startDate, Date endDate) {
91          role.setUserEntity(user);
92          role.setStartDate(startDate);
93          role.setEndDate(endDate);
94          super.create(role);
95       }
96
97       private void removeRoleFromUser(AbstractRole role) {
98          super.remove(role);
99       }
100
101      /**
102       * Finds and returns the users with the given role.
103       * Not implemented in version 1.0.
104       * @param role The role to be searched for.
105       * @return A list of the users with the role.
106       */
107      public List<? extends AbstractRole> listRoleUsers(AbstractRole role) {
108         return Collections.emptyList();
109      }
110
111      /**
112       * Checks if the user has the superuser rights.
113       * @param user The user to be checked for the superuser rights.
114       * @return true if the user had the superuser rights, false otherwise.
115       */
116      @Override
117      public boolean checkIfUserIsSuperUser(IdentifiedUserEntity user) {
118
119         TypedQuery typedQuery = em.createNamedQuery("findSuperUserRoleByUser",
                SuperUserRoleEntity.class);
120         typedQuery.setParameter("userEntity", user);
```

```
121
122         try {
123             SuperUserRoleEntity role = (SuperUserRoleEntity)typedQuery.
                    getSingleResult();
124             return role != null;
125         } catch (NoResultException nre) {
126             return false;
127         }
128     }
129 }
```

1.80 roles/SuperUserRoleEntity.java

```
1
2  package com.moveatis.roles;
3
4  import com.moveatis.user.IdentifiedUserEntity;
5  import java.io.Serializable;
6  import javax.persistence.Entity;
7  import javax.persistence.NamedQueries;
8  import javax.persistence.NamedQuery;
9  import javax.persistence.OneToOne;
10 import javax.persistence.Table;
11
12 /**
13  * The entity represents the superuser role, which can be added to the users,
14  * who should have the superuser rights to Moveatis. In version 1.0 doesn't
15  * include superuser tasks.
16  *
17  * @author Sami Kallio <phinaliumz at outlook.com>
18  */
19 @Entity
20 @Table(name="SUPERUSER_ROLE")
21 @NamedQueries({
22     @NamedQuery(name="findSuperUserRoleByUser", query="SELECT super FROM
          SuperUserRoleEntity super WHERE "
23             + "super.userEntity=:userEntity")
24 })
25 public class SuperUserRoleEntity extends AbstractRole implements Serializable {
26
27     private static final long serialVersionUID = 1L;
28
29     @OneToOne
30     private IdentifiedUserEntity userEntity;
31
32     @Override
33     public IdentifiedUserEntity getUserEntity() {
34         return userEntity;
35     }
36
37     @Override
38     public void setUserEntity(IdentifiedUserEntity userEntity) {
39         this.userEntity = userEntity;
40     }
41
```

```
42     @Override
43     public int hashCode() {
44        int hash = 0;
45        hash += (id != null ? id.hashCode() : 0);
46        return hash;
47     }
48
49     @Override
50     public boolean equals(Object object) {
51        if (!(object instanceof SuperUserRoleEntity)) {
52           return false;
53        }
54        SuperUserRoleEntity other = (SuperUserRoleEntity) object;
55        return !((this.id == null && other.id != null) || (this.id != null && !this.
              id.equals(other.id)));
56     }
57
58     @Override
59     public String toString() {
60        return "com.moveatis.roles.SuperUserRoleEntity[ id=" + id + " ]";
61     }
62
63  }
```

1.81 servlet/JyuIdentityServlet.java

```
1  package com.moveatis.servlet;
2
3  import com.moveatis.application.InstallationBean;
4  import com.moveatis.application.RedirectURLs;
5  import com.moveatis.enums.ApplicationStatusCode;
6  import com.moveatis.identityprovider.IdentityProviderBean;
7  import com.moveatis.identityprovider.IdentityProviderInformationEntity;
8  import com.moveatis.interfaces.Application;
9  import com.moveatis.interfaces.Role;
10 import com.moveatis.interfaces.Session;
11 import com.moveatis.interfaces.User;
12 import com.moveatis.user.IdentifiedUserEntity;
13 import java.io.IOException;
14 import javax.inject.Inject;
15 import javax.servlet.ServletException;
16 import javax.servlet.annotation.WebServlet;
17 import javax.servlet.http.HttpServlet;
18 import javax.servlet.http.HttpServletRequest;
19 import javax.servlet.http.HttpServletResponse;
20 import org.slf4j.Logger;
21 import org.slf4j.LoggerFactory;
22
23 /**
24  * The servlet handles the identification of a user using the Shibboleth service
25  * of Jyväskylä University.
26  *
27  * If you are modifying Moveatis to your own organization, you need to
28  * implement your own identity provider service with the classes in
29  * the identity provider package.
```

```
30  *
31  * @see IdentityProviderInformationEntity
32  * @see IdentityProvider
33  * @see IdentityProviderBean
34  * @see IdentityProviderRegistrationBean
35  * @author Sami Kallio <phinaliumz at outlook.com>
36  */
37 @WebServlet(name = "JyuIdentityServlet", urlPatterns = {"/moveatis/secure"})
38 public class JyuIdentityServlet extends HttpServlet {
39
40     private static final Logger LOGGER = LoggerFactory.getLogger(JyuIdentityServlet
        .class);
41
42     private IdentifiedUserEntity userEntity;
43
44     @Inject
45     private Session sessionBean;
46     @Inject
47     private IdentityProviderBean ipBean;
48     @Inject
49     private User userEJB;
50     @Inject
51     private Role roleEJB;
52     @Inject
53     private Application applicationEJB;
54     @Inject
55     private InstallationBean installationEJB;
56
57     /**
58      * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
59      * methods.
60      *
61      * @throws ServletException if a servlet-specific error occurs.
62      * @throws IOException if an I/O error occurs.
63      */
64     protected void processRequest(HttpServletRequest request, HttpServletResponse
        response)
65         throws ServletException, IOException {
66
67         String userName = (String) request.getAttribute("eppn");
68         String affiliation = (String) request.getAttribute("unscoped-affiliation");
69         String displayName = (String) request.getAttribute("displayName");
70
71         if(userName != null && affiliation != null && displayName != null) {
72             IdentityProviderInformationEntity ipInformationEntity = ipBean.
                findIpEntityByUsername(userName);
73
74             if(ipInformationEntity != null) {
75                 userEntity = ipInformationEntity.getIdentifiedUserEntity();
76                 sessionBean.setIdentityProviderUser(userEntity);
77                 response.sendRedirect(RedirectURLs.CONTROL_PAGE_URI);
78
79             } else {
80                 /*
81                 * IdentityProviderInformationEntity was not found, but as our service
                    is open to all
82                 * students and affiliates of Jyväskylä University, we shall create a
                    new entity for this user
```

```
83              */
84              userEntity = new IdentifiedUserEntity();
85
86              IdentityProviderInformationEntity identityProviderInformationEntity =
                    new IdentityProviderInformationEntity();
87              identityProviderInformationEntity.setUsername(userName);
88              identityProviderInformationEntity.setAffiliation(affiliation);
89
90              userEntity.setIdentityProviderInformationEntity(
                    identityProviderInformationEntity);
91              userEntity.setGivenName(displayName);
92
93              identityProviderInformationEntity.setUserEntity(userEntity);
94
95              userEJB.create(userEntity);
96              sessionBean.setIdentityProviderUser(userEntity);
97
98              if(!applicationEJB.checkInstalled()) {
99                  // Application itself has not been installed yet, so that
100                 // needs to be done
101                 // First user is the admin user
102                 roleEJB.addSuperuserRoleToUser(userEntity);
103
104                 if(installationEJB.createApplication() == ApplicationStatusCode.
                        INSTALLATION_OK) {
105                     response.sendRedirect(RedirectURLs.CONTROL_PAGE_URI);
106                 } else {
107                     response.sendError(HttpServletResponse.SC_SERVICE_UNAVAILABLE);
108                 }
109             } else {
110                 response.sendRedirect(RedirectURLs.CONTROL_PAGE_URI);
111             }
112         }
113     } else {
114         response.sendRedirect(RedirectURLs.HOME_URI);
115     }
116 }
117
118 /**
119  * Handles the HTTP <code>POST</code> method.
120  *
121  * @throws ServletException if a servlet-specific error occurs.
122  * @throws IOException if an I/O error occurs.
123  */
124 @Override
125 protected void doPost(HttpServletRequest request, HttpServletResponse response)
126         throws ServletException, IOException {
127     processRequest(request, response);
128 }
129
130 @Override
131 protected void doGet(HttpServletRequest request, HttpServletResponse response)
132         throws ServletException, IOException {
133     processRequest(request, response);
134 }
135
136 /**
137  * Returns a short description of the servlet.
```

```
138      *
139      * @return a String containing servlet description.
140      */
141     @Override
142     public String getServletInfo() {
143        return "This servlet is the endpoint to Shibboleth-identityprovider service"
               ;
144     }
145
146 }
```

1.82 session/LogoutBean.java

```
1
2  package com.moveatis.session;
3
4  import com.moveatis.application.RedirectURLs;
5  import java.io.IOException;
6  import javax.inject.Named;
7  import javax.enterprise.context.RequestScoped;
8  import javax.faces.context.ExternalContext;
9  import javax.faces.context.FacesContext;
10 import javax.faces.event.ActionEvent;
11 import org.slf4j.Logger;
12 import org.slf4j.LoggerFactory;
13
14 /**
15  * The bean manages logging out of Moveatis, which includes invalidating the
       session
16  * and redirecting the user to the Shibboleth logout URI.
17  * @author Sami Kallio <phinaliumz at outlook.com>
18  */
19 @Named(value="logoutBean")
20 @RequestScoped
21 public class LogoutBean {
22
23    private static final Logger LOGGER = LoggerFactory.getLogger(LogoutBean.class);
24
25    public LogoutBean() {
26
27    }
28
29    /**
30     * Logs the user out by invalidating the session and redirecting the user to
31     * the Shibboleth logout URI.
32     */
33    public void logOut(ActionEvent event) throws IOException {
34       ExternalContext context = FacesContext.getCurrentInstance().
             getExternalContext();
35       context.invalidateSession();
36
37       context.redirect(RedirectURLs.SHIBBOLET_LOGOUT_URL);
38    }
39
40 }
```

1.83 session/SessionBean.java

```
1  package com.moveatis.session;
2
3  import com.moveatis.groupkey.GroupKeyEntity;
4  import com.moveatis.interfaces.Session;
5  import com.moveatis.managedbeans.ObservationManagedBean;
6  import com.moveatis.observation.ObservationCategorySet;
7  import com.moveatis.timezone.TimeZoneInformation;
8  import com.moveatis.user.AbstractUser;
9  import com.moveatis.user.IdentifiedUserEntity;
10 import com.moveatis.user.TagUserEntity;
11 import java.io.Serializable;
12 import java.util.List;
13 import java.util.Locale;
14 import java.util.SortedSet;
15 import java.util.TimeZone;
16 import java.util.TreeSet;
17 import javax.enterprise.context.SessionScoped;
18 import javax.faces.context.FacesContext;
19 import javax.inject.Inject;
20 import javax.inject.Named;
21 import javax.servlet.http.HttpServletRequest;
22 import org.slf4j.Logger;
23 import org.slf4j.LoggerFactory;
24
25 /**
26  * The bean manages actions the user needs in the usage of Moveatis.
27  * @author Sami Kallio <phinaliumz at outlook.com>
28  */
29 @SessionScoped
30 @Named
31 public class SessionBean implements Serializable, Session {
32
33    private static final long serialVersionUID = 1L;
34    private static final Logger LOGGER = LoggerFactory.getLogger(SessionBean.class)
          ;
35
36    @Inject
37    private ObservationManagedBean observationManagedBean;
38
39    private boolean loggedIn = false;
40    private IdentifiedUserEntity userEntity;
41    private TagUserEntity tagEntity;
42
43    private SortedSet<Long> sessionObservations;
44
45    private String returnUri;
46
47    private TimeZone sessionTimeZone = TimeZoneInformation.getTimeZone();
48    private Locale locale; // Locale switching based on BalusC's example: http://
          stackoverflow.com/a/4830669
49
50
51    public SessionBean() {
52
53    }
54
```

```
55      @Override
56      public void setIdentityProviderUser(IdentifiedUserEntity user) {
57          this.userEntity = user;
58          commonSettingsForLoggedInUsers();
59      }
60
61      @Override
62      public void setAnonymityUser() {
63          // TODO: Doesn't set abstractUser. Is this ok?
64          tagEntity = null;
65          commonSettingsForLoggedInUsers();
66          // If user wants to observe without selecting existing event group
67          // (in control view or with a group key), we should reset the event.
68          observationManagedBean.setEventEntity(null);
69      }
70
71      @Override
72      public void setTagUser(TagUserEntity tagUser) {
73          if(tagUser == null) {
74              return;
75          }
76          this.tagEntity = tagUser;
77          commonSettingsForLoggedInUsers();
78          observationManagedBean.setEventEntity(tagUser.getGroupKey().getEventGroup().
                getEvent());
79      }
80
81      private void commonSettingsForLoggedInUsers() {
82          this.loggedIn = true;
83          // Make sure we don't modify earlier categories.
84          observationManagedBean.resetCategorySetsInUse();
85      }
86
87      @Override
88      public boolean isLoggedIn() {
89          return loggedIn;
90      }
91
92      @Override
93      public String toString() {
94          String userType = this.tagEntity != null ? "tag" : "anonymous";
95          userType = this.userEntity != null ? "identified" : userType;
96          return "SessionBean: userType -> " + userType + ", loggedIn -> " +
                isLoggedIn();
97      }
98
99      @Override
100     public SortedSet<Long> getSessionObservationsIds() {
101         if(this.sessionObservations == null) {
102             return new TreeSet<>();
103         }
104         return this.sessionObservations;
105     }
106
107     /**
108      * Checks if the observation is in saveable state. It is used in checking if
109      * the Save button can be displayed.
110      * @return true if the observation could be saved.
```

```
111      */
112     @Override
113     public boolean isSaveable() {
114        return observationManagedBean.getObservationEntity() != null;
115     }
116
117     @Override
118     public void setSessionObservations(SortedSet<Long> observationsIds) {
119        this.sessionObservations = observationsIds;
120     }
121
122     @Override
123     public AbstractUser getLoggedInUser() {
124        return this.tagEntity;
125     }
126
127     @Override
128     public TimeZone getSessionTimeZone() {
129        return sessionTimeZone;
130     }
131
132     @Override
133     public void setSessionTimeZone(TimeZone timeZone) {
134        this.sessionTimeZone = timeZone;
135     }
136
137     @Override
138     public IdentifiedUserEntity getLoggedIdentifiedUser() {
139        return this.userEntity;
140     }
141
142     @Override
143     public GroupKeyEntity getGroupKey() {
144        return this.tagEntity.getGroupKey();
145     }
146
147     /**
148      * Returns true if the button that resets the current observation
149      * should be available to the user.
150      */
151     public boolean isResetObsAvailable() {
152        String viewId = FacesContext.getCurrentInstance().getViewRoot().getViewId();
153        boolean result = (viewId.equals("/app/observer/index.xhtml") || viewId.
               equals("/app/summary/index.xhtml"));
154        return result;
155     }
156
157     /**
158      * Returns true if the button that redirects the user to the category
159      * selection view should be available to the user.
160      */
161     public boolean isBackToCatEdAvailable() {
162        String viewId = FacesContext.getCurrentInstance().getViewRoot().getViewId();
163        boolean result = (viewId.equals("/app/observer/index.xhtml"));
164        return result;
165     }
166
167     /**
```

```
168      * Returns true if the button that redirects the user to the front page
169      * should be available to the user.
170      */
171     public boolean isToFrontPageAvailable() {
172        String viewId = FacesContext.getCurrentInstance().getViewRoot().getViewId();
173        boolean result = !(viewId.equals("/index.xhtml"));
174        return result;
175     }
176
177     /**
178      * Used in development of Moveatis to detect whether the application is running
179      * on the actual production server or on localhost.
180      * @return true if Moveatis is running in localhost, false otherwise.
181      */
182     @Override
183     public boolean getIsLocalhost() {
184        boolean isLocalhost = ((HttpServletRequest) FacesContext.getCurrentInstance
             ().getExternalContext().getRequest())
185              .getRequestURL().toString().contains("localhost");
186        return isLocalhost;
187     }
188
189     /**
190      * Returns the URI in which the user was before he or she clicked the
191      * login button when not in the front page. Not implemented in version 1.0.
192      * @return the URI to return the user to.
193      */
194     @Override
195     public String getReturnUri() {
196        return returnUri;
197     }
198
199     /**
200      * Sets the URI into which the user will be returned when he or she
201      * clicks the login button outside of the front page.
202      * Not implemented in version 1.0.
203      * @param returnUri The URI that is set as return URI.
204      */
205     @Override
206     public void setReturnUri(String returnUri) {
207        this.returnUri = returnUri;
208     }
209
210     @Override
211     public boolean isIdentifiedUser() {
212        return userEntity != null;
213     }
214
215     @Override
216     public void setCategorySetsInUse(List<ObservationCategorySet> categorySets) {
217        observationManagedBean.setCategorySetsInUse(categorySets);
218     }
219
220     @Override
221     public List<ObservationCategorySet> getCategorySetsInUse() {
222        return observationManagedBean.getCategorySetsInUse();
223     }
224 }
```

1.84 timezone/TimeZoneInformation.java

```
1  package com.moveatis.timezone;
2
3  import java.util.Calendar;
4  import java.util.SimpleTimeZone;
5  import java.util.TimeZone;
6
7  /**
8   * The class to keep the default time zone of Moveatis.
9   * @author Sami Kallio <phinaliumz at outlook.com>
10  */
11 public class TimeZoneInformation {
12
13    /**
14     * We use stardard UTC timezone for saving information to
15     * the server - the client can then convert this time
16     * his/her timezone
17     */
18    private static final TimeZone TIMEZONE = TimeZone.getTimeZone("UTC");
19
20    public TimeZoneInformation() {
21
22    }
23
24    public static TimeZone getTimeZone() {
25       return TIMEZONE;
26    }
27
28    /**
29     * Gets a TimeZone from the time zone offset and the daylight saving time.
30     *
31     * @param offset the time zone offset in milliseconds.
32     * @param DSTSaving the daylight saving time in milliseconds.
33     */
34    public static TimeZone getTimeZoneFromOffset(int offset, int DSTSaving) {
35       if (DSTSaving > 0) {
36          return new SimpleTimeZone(
37                offset - DSTSaving, "GMT/" + offset,
38                Calendar.JANUARY, -1, Calendar.SUNDAY,
39                2,
40                Calendar.DECEMBER, -1, Calendar.SUNDAY,
41                2,
42                DSTSaving
43          );
44       } else {
45          return new SimpleTimeZone(offset, "GMT/" + offset);
46       }
47    }
48 }
```

1.85 user/AbstractUser.java

```
1
2  package com.moveatis.user;
```

```
3
4  import com.moveatis.abstracts.BaseEntity;
5  import java.io.Serializable;
6  import javax.persistence.DiscriminatorColumn;
7  import javax.persistence.Entity;
8  import javax.persistence.Inheritance;
9  import javax.persistence.InheritanceType;
10
11 /**
12  * The base user entity for TagUser and IdentifiendUser. TagUser is the user
13  * which provides access for group keys and the identified user is an individual
14  * user, who is identified using the Shibboleth identity system of Jyväskylä
        University.
15  * @author Sami Kallio <phinaliumz at outlook.com>
16  */
17 @Entity
18 @Inheritance(strategy=InheritanceType.JOINED)
19 @DiscriminatorColumn(name="USER_TYPE")
20 public abstract class AbstractUser extends BaseEntity implements Serializable {
21
22    @Override
23    public Long getId() {
24       return id;
25    }
26
27    @Override
28    public void setId(Long id) {
29       this.id = id;
30    }
31
32    @Override
33    public int hashCode() {
34       int hash = 0;
35       hash += (id != null ? id.hashCode() : 0);
36       return hash;
37    }
38
39    @Override
40    public boolean equals(Object object) {
41       if (!(object instanceof AbstractUser)) {
42          return false;
43       }
44       AbstractUser other = (AbstractUser) object;
45       return !((this.id == null && other.id != null) || (this.id != null && !this.
             id.equals(other.id)));
46    }
47
48    @Override
49    public String toString() {
50       return "com.moveatis.user.AbstractUser[ id=" + id + " ]";
51    }
52 }
```

1.86 user/AnonUserBean.java

```java
package com.moveatis.user;

import com.moveatis.abstracts.AbstractBean;
import com.moveatis.interfaces.AnonUser;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.TypedQuery;
import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;

/**
 * The EJB manages the anonymity user, which represents the public user
 * of Moveatis. The class is not used in version 1.0
 * @author Sami Kallio <phinaliumz at outlook.com>
 */
@Stateless
public class AnonUserBean extends AbstractBean<AnonUserEntity> implements AnonUser
    {

   @PersistenceContext(unitName = "MOVEATIS_PERSISTENCE")
   private EntityManager em;

   public AnonUserBean() {
      super(AnonUserEntity.class);
   }

   @Override
   protected EntityManager getEntityManager() {
      return em;
   }

   @Override
   public AnonUserEntity find() {
      CriteriaBuilder cb = getEntityManager().getCriteriaBuilder();
      CriteriaQuery<AnonUserEntity> cq = cb.createQuery(AnonUserEntity.class);

      Root<AnonUserEntity> rt = cq.from(AnonUserEntity.class);
      CriteriaQuery<AnonUserEntity> all = cq.select(rt);

      TypedQuery<AnonUserEntity> allQuery = getEntityManager().createQuery(all);
      return allQuery.getSingleResult();
   }
}
```

1.87 user/AnonUserEntity.java

```java
package com.moveatis.user;

import java.io.Serializable;
import javax.persistence.Entity;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

```

```
8  /**
9   * The entity represents one public user account in Moveatis. It's used
10  * as bookkeeping purposes, as each session must have a user.
11  * @author Sami Kallio <phinaliumz at outlook.com>
12  */
13 @Entity
14 @Table(name="PUBLIC_USER")
15 public class AnonUserEntity extends AbstractUser implements Serializable {
16
17    private static final long serialVersionUID = 1L;
18
19    @ManyToOne
20    private IdentifiedUserEntity creator;
21
22    private String label;
23
24    public IdentifiedUserEntity getCreator() {
25       return creator;
26    }
27
28    public void setCreator(IdentifiedUserEntity creator) {
29       this.creator = creator;
30    }
31
32    public String getLabel() {
33       return label;
34    }
35
36    public void setLabel(String label) {
37       this.label = label;
38    }
39
40    @Override
41    public int hashCode() {
42       int hash = 0;
43       hash += (id != null ? id.hashCode() : 0);
44       return hash;
45    }
46
47    @Override
48    public boolean equals(Object object) {
49       if (!(object instanceof AnonUserEntity)) {
50          return false;
51       }
52       AnonUserEntity other = (AnonUserEntity) object;
53       return !((this.id == null && other.id != null) || (this.id != null && !this.
             id.equals(other.id)));
54    }
55
56    @Override
57    public String toString() {
58       return "com.moveatis.user.AnonUserEntity[ id=" + id + " ]";
59    }
60 }
```

1.88 user/IdentifiedUserEntity.java

```
1  package com.moveatis.user;
2
3  import com.moveatis.identityprovider.IdentityProviderInformationEntity;
4  import java.io.Serializable;
5  import static javax.persistence.CascadeType.PERSIST;
6  import javax.persistence.Entity;
7  import javax.persistence.NamedQueries;
8  import javax.persistence.NamedQuery;
9  import javax.persistence.OneToOne;
10 import javax.persistence.Table;
11
12 /**
13  * The entity represents the individual user, which is identified
14  * using the Shibboleth identity system of Jyväskylä University.
15  * @author Sami Kallio <phinaliumz at outlook.com>
16  */
17 @Entity
18 @NamedQueries({
19       @NamedQuery(
20          name="findUserByName",
21          query="SELECT user FROM IdentifiedUserEntity user WHERE user.givenName=:
                 givenName"
22       )
23 })
24 @Table(name="IDENTIFIED_USER")
25 public class IdentifiedUserEntity extends AbstractUser implements Serializable {
26
27    private static final long serialVersionUID = 1L;
28
29    @OneToOne(mappedBy = "userEntity", cascade=PERSIST)
30    private IdentityProviderInformationEntity identityProviderInformation;
31
32    private String givenName;
33    private String email;
34
35    public IdentityProviderInformationEntity getIdentityProviderInformation() {
36       return identityProviderInformation;
37    }
38
39    public void setIdentityProviderInformation(IdentityProviderInformationEntity
          identityProviderInformation) {
40       this.identityProviderInformation = identityProviderInformation;
41    }
42
43    public String getGivenName() {
44       return givenName;
45    }
46
47    public void setGivenName(String givenName) {
48       this.givenName = givenName;
49    }
50
51    public IdentityProviderInformationEntity getIdentityProviderInformationEntity()
          {
52       return identityProviderInformation;
53    }
```

```
54
55      public void setIdentityProviderInformationEntity(
            IdentityProviderInformationEntity identityProviderInformation) {
56          this.identityProviderInformation = identityProviderInformation;
57      }
58
59      public String getEmail() {
60          return email;
61      }
62
63      public void setEmail(String email) {
64          this.email = email;
65      }
66
67      @Override
68      public int hashCode() {
69          int hash = 0;
70          hash += (id != null ? id.hashCode() : 0);
71          return hash;
72      }
73
74      @Override
75      public boolean equals(Object object) {
76          if (!(object instanceof IdentifiedUserEntity)) {
77              return false;
78          }
79          IdentifiedUserEntity other = (IdentifiedUserEntity) object;
80          return !((this.id == null && other.id != null) || (this.id != null && !this.
                id.equals(other.id)));
81      }
82
83      @Override
84      public String toString() {
85          return "com.moveatis.user.IdentifiedUserEntity[ id=" + id + " ]";
86      }
87  }
```

1.89 user/TagUserBean.java

```
1   package com.moveatis.user;
2
3   import com.moveatis.groupkey.GroupKeyEntity;
4   import com.moveatis.abstracts.AbstractBean;
5   import com.moveatis.interfaces.TagUser;
6   import javax.ejb.Stateless;
7   import javax.persistence.EntityManager;
8   import javax.persistence.NoResultException;
9   import javax.persistence.PersistenceContext;
10  import javax.persistence.TypedQuery;
11
12  /**
13   * The EJB is used in managing of TagUser, which represents the user account for
14   * accessing with a group key into Moveatis.
15   * @author Sami Kallio <phinaliumz at outlook.com>
16   */
```

```
@Stateless
public class TagUserBean extends AbstractBean<TagUserEntity> implements TagUser {

   @PersistenceContext(unitName = "MOVEATIS_PERSISTENCE")
   private EntityManager em;

   public TagUserBean() {
      super(TagUserEntity.class);
   }

   @Override
   protected EntityManager getEntityManager() {
      return em;
   }

   @Override
   public TagUserEntity findByKey(GroupKeyEntity groupkey) {
      TypedQuery<TagUserEntity> query = em.createNamedQuery("findTagUserByTag",
          TagUserEntity.class);
      query.setParameter("groupKey", groupkey);
      try {
         TagUserEntity tagUserEntity = query.getSingleResult();
         return tagUserEntity;
      } catch(NoResultException nre) {
         return null;
      }
   }
}
```

1.90 user/TagUserEntity.java

```
package com.moveatis.user;

import com.moveatis.groupkey.GroupKeyEntity;
import java.io.Serializable;
import static javax.persistence.CascadeType.PERSIST;
import javax.persistence.Entity;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToOne;
import javax.persistence.Table;

/**
 * The entity represent the data for accessing with a group key into Moveatis.
 * @author Sami Kallio <phinaliumz at outlook.com>
 */
@Entity
@NamedQueries({
      @NamedQuery(
         name="findTagUserByTag",
         query="SELECT user FROM TagUserEntity user WHERE user.groupKey=:groupKey"
      )
})
```

```
25 @Table(name="TAG_USER")
26 public class TagUserEntity extends AbstractUser implements Serializable {
27
28    private static final long serialVersionUID = 1L;
29
30    @OneToOne(cascade=PERSIST)
31    private GroupKeyEntity groupKey;
32
33    @ManyToOne
34    @JoinColumn(name="CREATOR_ID")
35    private IdentifiedUserEntity creator;
36
37    private String label;
38
39    public GroupKeyEntity getGroupKey() {
40       return groupKey;
41    }
42
43    public void setGroupKey(GroupKeyEntity groupKey) {
44       this.groupKey = groupKey;
45    }
46
47    public IdentifiedUserEntity getCreator() {
48       return creator;
49    }
50
51    public void setCreator(IdentifiedUserEntity creator) {
52       this.creator = creator;
53    }
54
55    public String getLabel() {
56       return label;
57    }
58
59    public void setLabel(String label) {
60       this.label = label;
61    }
62
63    @Override
64    public int hashCode() {
65       int hash = 0;
66       hash += (id != null ? id.hashCode() : 0);
67       return hash;
68    }
69
70    @Override
71    public boolean equals(Object object) {
72       if (!(object instanceof TagUserEntity)) {
73          return false;
74       }
75       TagUserEntity other = (TagUserEntity) object;
76       return !((this.id == null && other.id != null) || (this.id != null && !this.
             id.equals(other.id)));
77    }
78
79    @Override
80    public String toString() {
81       return "com.moveatis.user.TagUserEntity[ id=" + id + " ]";
```

```
82      }
83  }
```

1.91 user/UserBean.java

```
1   package com.moveatis.user;
2
3   import com.moveatis.abstracts.AbstractBean;
4   import javax.ejb.Stateless;
5   import javax.persistence.EntityManager;
6   import javax.persistence.PersistenceContext;
7   import com.moveatis.interfaces.User;
8   import org.slf4j.Logger;
9   import org.slf4j.LoggerFactory;
10
11  /**
12   * The EJB manages an identified user. It is not used much in version 1.0
13   * @author Sami Kallio <phinalium at outlook.com>
14   */
15  @Stateless
16  public class UserBean extends AbstractBean<IdentifiedUserEntity> implements User {
17
18      private static final Logger LOGGER = LoggerFactory.getLogger(UserBean.class);
19
20      @PersistenceContext(unitName = "MOVEATIS_PERSISTENCE")
21      private EntityManager em;
22
23      @Override
24      protected EntityManager getEntityManager() {
25          return em;
26      }
27
28      public UserBean() {
29          super(IdentifiedUserEntity.class);
30      }
31
32  }
```

1.92 validation/EmailValidatorBean.java

```
1
2   package com.moveatis.validation;
3
4   import java.util.Map;
5   import javax.faces.application.FacesMessage;
6   import javax.faces.component.UIComponent;
7   import javax.faces.context.FacesContext;
8   import javax.faces.validator.FacesValidator;
9   import javax.faces.validator.Validator;
10  import javax.faces.validator.ValidatorException;
11  import org.apache.commons.validator.routines.EmailValidator;
12  import org.primefaces.validate.ClientValidator;
13
```

```
/**
 * The validator for user submitted email addresses.
 *
 * @author Sami Kallio <phinaliumz at outlook.com>
 */
@FacesValidator("emailValidator")
public class EmailValidatorBean implements Validator, ClientValidator{

   public EmailValidatorBean() {

   }

   /**
    * Validates the email address.
    * @param context The FacesContext for the session.
    * @param component The component to be validated.
    * @param value The value for the component.
    * @throws ValidatorException if the email address was not valid.
    */
   @Override
   public void validate(FacesContext context, UIComponent component, Object value)
        throws ValidatorException {
      if(value == null) {
         return;
      }

      String email = (String)value;
      EmailValidator emailValidator = EmailValidator.getInstance();
      if(emailValidator.isValid(email)) {

      } else {
         throw new ValidatorException(new FacesMessage(FacesMessage.SEVERITY_ERROR
             , "Validation error",
            email + " is not a valid email address"));
      }
   }

   @Override
   public Map<String, Object> getMetadata() {
      return null;
   }

   @Override
   public String getValidatorId() {
      return "emailValidator";
   }

}
```

Chapter 2

JavaScript source code

2.1 resources/js/control.js

```
1
2  /**
3   * @fileOverview JavaScript methods for control view.
4   * @module control
5   * @author Juha Moisio <juha.pa.moisio at student.jyu.fi>
6   */
7
8  /* global PrimeFaces, PF */
9
10 /**
11  * Hides the dialog on succesfull submit and displays the given message.
12  * @param {object} args - Containing the validationFailed attribute.
13  * @param {string} dialogWidgetVar - The widget variable of the dialog.
14  * @param {string} message - The given message.
15  */
16 function onDialogSuccess(args, dialogWidgetVar, message) {
17    if (args && !args.validationFailed) {
18       PF(dialogWidgetVar).hide();
19       if (message) {
20          PF('growlWdgt').renderMessage({summary: message, severity: 'info'});
21       }
22    }
23 }
24
25 /**
26  * Sets the last editable data table row into edit mode and sets the focus to it's
        first editable input.
27  * @param {string} table The identifier, id or class name of the data table.
28  */
29 function focusDataTableEditInput(table) {
30    $(table + ' .ui-datatable-data tr').last().find('.ui-icon-pencil').each(
          function(){
31       $(this).click();
32    });
33    $(table + ' .ui-datatable-data tr').last().find('.ui-editable-column:first-
          child input').focus();
34    $(table + ' .ui-datatable-data tr').last().find('.ui-editable-column:first-
          child input').select();
35 }
36
```

```
/**
 * Submits the data table rows in edit mode.
 * @param {string} table - The identifier, id or class name of the data table.
 */
function submitDataTableEditInputs(table) {
   $(table + ' .ui-datatable-data tr').find('.ui-icon-check').each(function(){
      $(this).click();
   });
}
```

2.2 resources/js/locales.js

```

/**
 * @fileOverview PrimeFaces localization file.
 * @module locales
 * @author Juha Moisio <juha.pa.moisio at student.jyu.fi>
 * @property {object} PrimeFaces.locales.fi Finnish localization.
 */
PrimeFaces.locales['fi'] = {
   closeText: 'Sulje',
   prevText: 'Edellinen',
   nextText: 'Seuraava',
   monthNames: ['tammikuu', 'helmikuu', 'maaliskuu', 'huhtikuu', 'toukokuu', 'kesäkuu', 'heinäkuu', 'elokuu', 'syyskuu', 'lokakuu', 'marraskuu', 'joulukuu'],
   monthNamesShort: ['tammi', 'helmi', 'maalis', 'huhti', 'touko', 'kesä', 'heinä', 'elo', 'syys', 'loka', 'marras', 'joulu'],
   dayNames: ['sunnuntai', 'maanantai', 'tiistai', 'keskiviikko', 'torstai', 'perjantai', 'lauantai'],
   dayNamesShort: ['su', 'ma', 'ti', 'ke', 'to', 'pe', 'la'],
   dayNamesMin: ['su', 'ma', 'ti', 'ke', 'to', 'pe', 'la'],
   weekHeader: 'Viikko',
   firstDay: 1,
   isRTL: false,
   showMonthAfterYear: false,
   yearSuffix: '',
   timeOnlyTitle: 'Valitse aika',
   timeText: 'Aika',
   hourText: 'Tunnit',
   minuteText: 'Minuutit',
   secondText: 'Sekunnit',
   currentText: 'Tämä päivä',
   ampm: false,
   month: 'kuukausi',
   week: 'viikko',
   day: 'päivä',
   allDayText: 'Koko päivä'
};
```

2.3 resources/js/observer.js

```

/**
```

```
3  * @fileOverview JavaScript logic for observation view.
4  * @module observer
5  * @author Ilari Paananen <ilari.k.paananen at student.jyu.fi>
6  */
7
8  //
9  // TODO:
10 // - Remove/comment out console.log calls in release?
11 //
12
13 // NOTE: Functions in observer/index.xhtml.
14 var CategoryType = getCategoryTypes();
15 var msg = getMessages();
16
17
18 /**
19  * The observation clock that can be paused and resumed.
20  * The individual categories get their time from the observation clock.
21  * @constructor
22  */
23 function Clock() {
24    this.total_time = 0;
25    this.resume_time = 0;
26    this.running = false;
27
28    /**
29     * Resumes the observation clock.
30     * @param {number} now The time in milliseconds when the observation clock was
           resumed.
31     */
32    this.resume = function(now) {
33       if (!this.running) {
34          this.resume_time = now;
35          this.running = true;
36       } else {
37          console.log("Clock.resume(): Clock is already running!");
38       }
39    };
40
41    /**
42     * Pauses the observation clock.
43     * @param {number} now The time in milliseconds when the observation clock was
           paused.
44     */
45    this.pause = function(now) {
46       if (this.running) {
47          var delta_time = now - this.resume_time;
48          this.total_time += delta_time;
49          this.running = false;
50       } else {
51          console.log("Clock.pause(): Clock is already paused!");
52       }
53    };
54
55    /**
56     * Returns the total time the observation clock has been running in milliseconds
           .
```

```
57      * @param {number} now The time in milliseconds when the elapsed time was wanted
            .
58      */
59     this.getElapsedTime = function(now) {
60        if (this.running) {
61           return this.total_time + (now - this.resume_time);
62        } else {
63           return this.total_time;
64        }
65     };
66
67     /**
68      * Returns true if the observation clock is paused, otherwise false.
69      */
70     this.isPaused = function() {
71        return !this.running;
72     };
73  }
74
75
76  /**
77   * Converts milliseconds to a string representing time.
78   * The time format is hh:mm:ss if the given time is at
79   * least one hour and mm:ss otherwise.
80   * @param {number} ms The time in milliseconds.
81   */
82  function timeToString(ms) {
83     var t = Math.floor(ms / 1000);
84     var s = t % 60;
85     var m = Math.floor(t / 60) % 60;
86     var h = Math.floor(t / 60 / 60) % 60;
87     var str = (m < 10 ? "0" + m : m) + ":" + (s < 10 ? "0" + s : s);
88     if (h > 0) {
89        str = (h < 10 ? "0" + h : h) + ":" + str;
90     }
91     return str;
92  }
93
94
95  /**
96   * Returns the given count as a string with abbreviation, e.g. "13 ct.".
97   * @param {number} count The count to make the string from.
98   */
99  function countToString(count) {
100    return count + " " + msg.countAbbreviation;
101 }
102
103
104 /**
105  * The class acts as a category button. It creates the HTML
106  * elements it needs and responds to the click events it gets.
107  * @constructor
108  * @param {String} name The name to be displayed on the button.
109  * @param {number} type The type of the category (TIMED or COUNTED).
110  * @param {number} index The index of the category button.
111  * @returns {CategoryItem}
112  */
113 function CategoryItem(name, type, id, index) {
```

```
114     this.li = $(document.createElement("li"));
115     this.li.addClass("category-item");
116     this.li.attr("id", "category-item_" + index);
117     this.value_div = $(document.createElement("div"));
118     this.value_div.addClass("category-value");
119     this.name_div = $(document.createElement("div"));
120     this.name_div.addClass("category-name");
121     this.name_div.append(document.createTextNode(name));
122     this.li.append(this.value_div);
123     this.li.append(this.name_div);
124
125     this.type = type;
126     this.id = id;
127
128     // Used if type is COUNTED.
129     this.count = 0;
130
131     // Used if type is TIMED.
132     this.time = 0;
133     this.start_time = 0;
134     this.down = false;
135
136
137     if (this.type === CategoryType.TIMED)
138        initTimedCategory(this);
139     else
140        initCountedCategory(this);
141
142
143     /**
144      * The private method replaces the contents of the HTML element
145      * that displays the value of the category button.
146      * @param {CategoryItem} this_ The category button.
147      * @param {String} text The text to replace the contents of the element with.
148      */
149     function updateValueDiv(this_, text) {
150        this_.value_div.empty();
151        this_.value_div.append(document.createTextNode(text));
152     }
153
154     /**
155      * The private method initializes the category button to behave
156      * as a time interval category.
157      * @param {CategoryItem} this_ The category button.
158      */
159     function initTimedCategory(this_) {
160        updateValueDiv(this_, timeToString(0));
161
162        /*
163         * Click handler for timed category item.
164         */
165        this_.click = function(master_time) {
166           var record;
167
168           if (this.down) {
169              this.li.removeClass("down");
170              if (master_time > this.start_time) {
171                 this.time += master_time - this.start_time;
```

```
172                 record = {id: this.id, startTime: this.start_time, endTime:
                        master_time};
173             }
174             this.down = false;
175          } else {
176             this.li.addClass("down");
177             this.start_time = master_time;
178             this.down = true;
179          }
180
181          return record;
182       };
183
184       /*
185        * Updates category item's timer div if the category type is timed.
186        */
187       this_.updateTimer = function(master_time) {
188          var time = this.time;
189          if (this.down) {
190             time += master_time - this.start_time;
191          }
192          updateValueDiv(this, timeToString(time));
193       };
194    }
195
196    /**
197     * The private method initializes the category button to behave
198     * as a category that counts the click events it gets.
199     * @param {CategoryItem} this_ The category button.
200     */
201    function initCountedCategory(this_) {
202       updateValueDiv(this_, countToString(0));
203
204       /*
205        * Click handler for counted category item.
206        */
207       this_.click = function(master_time, paused) {
208          if (paused) return;
209
210          this.count += 1;
211          updateValueDiv(this, countToString(this.count));
212
213          this.li.addClass("down");
214          var item = this.li;
215          setTimeout(function() { item.removeClass("down"); }, 50);
216
217          return {id: this.id, startTime: master_time, endTime: master_time};
218       };
219
220       /*
221        * Does nothing because the category type is counted.
222        */
223       this_.updateTimer = function() { };
224    }
225 }
226
227
228 /**
```

```
229  * The class does the actual observation. It keeps the records
230  * made during the observation and sends them to the backend after
231  * the observation is stopped.
232  * @constructor
233  * @param category_sets The array of the category sets to be used in the
         observation.
234  * @returns {Observer} Constructed observer.
235  */
236 function Observer(category_sets) {
237    this.master_clock = new Clock();
238    this.categories = [];
239    this.records = [];
240    this.started = false;
241    this.waiting = false;
242
243    initialize(this);
244
245    /**
246     * The private method initializes the observer. It creates
247     * the category buttons and adds them to the HTML element tree.
248     * @param {Observer} this_ The observer to be initialized.
249     */
250    function initialize(this_) {
251       $("#continue").hide();
252       $("#pause").hide();
253       $("#stop").hide();
254       $("#total-time").append(document.createTextNode(timeToString(0)));
255
256       var category_list = $("#category-list");
257
258       var index = 0;
259
260       for (var i = 0; i < category_sets.length; i++) {
261
262          var set = category_sets[i];
263
264          if (set.categories.length > 0) {
265             var category_set = $(document.createElement("ul"));
266             category_set.attr("id", set.name);
267             category_set.addClass("category-set");
268
269             for (var j = 0; j < set.categories.length; j++) {
270                var cat = set.categories[j];
271                var category = new CategoryItem(cat.name, cat.type, cat.id, index);
272                this_.categories.push(category);
273                category_set.append(category.li);
274
275                index += 1;
276             }
277
278             category_list.append(category_set);
279          }
280       }
281
282       $(".category-item").addClass("disabled");
283    }
284
285    /**
```

```
286     * The private method adds the record to the records list if it's not undefined.
287     * The method is used by categoryClick() and stopClick().
288     * @param record The record or undefined if there is nothing to be added.
289     */
290    function addRecord(this_, record) {
291       if (record !== undefined) {
292          this_.records.push(record);
293       }
294    }
295
296    /**
297     * The event handler starts the observation. It sends an AJAX notification
298     * to the backend when the observation is started.
299     */
300    this.startClick = function() {
301       if (this.waiting) return;
302       this.waiting = true;
303
304       var this_ = this;
305
306       $.ajax({
307          url: "../../webapi/records/startobservation",
308          type: "POST",
309          dataType: "text",
310          contentType: "text/plain",
311          cache: false,
312          data: "start observation",
313          success: function(data) {
314             this_.master_clock.resume(Date.now());
315             this_.started = true;
316             this_.waiting = false;
317             var start_button = $("#start");
318             start_button.off("click");
319             start_button.hide();
320             $("#pause").show();
321             $("#stop-disabled").hide();
322             $("#stop").show();
323             $(".category-item").removeClass("disabled");
324          },
325          error: function(xhr, status, error) {
326             showError(msg.obs_errorCouldntSendStart + " " + error);
327             this_.waiting = false;
328          }
329       });
330    };
331
332    /**
333     * The event handler continues the observation.
334     */
335    this.continueClick = function () {
336       if (this.master_clock.isPaused()) {
337          this.master_clock.resume(Date.now());
338          $("#continue").hide();
339          $("#pause").show();
340       }
341    };
342
343    /**
```

```
344      * The event handler pauses the observation.
345      */
346     this.pauseClick = function() {
347        if (!this.master_clock.isPaused()) {
348           this.master_clock.pause(Date.now());
349           $("#pause").hide();
350           $("#continue").show();
351        }
352     };
353
354     /**
355      * The event handler stops the observation.
356      * It disables the continue, pause and category buttons.
357      * If some categories were still on, it stops them
358      * and creates records accordingly. It sends the recorded
359      * information to the backend with AJAX and
360      * redirects the user to the summary page (on success).
361      */
362     this.stopClick = function() {
363        if (!this.started || this.waiting) return;
364        this.waiting = true;
365
366        var now = Date.now();
367
368        if (!this.master_clock.isPaused()) {
369           this.master_clock.pause(now);
370        }
371
372        var continue_button = $("#continue");
373        var pause_button = $("#pause");
374        continue_button.off("click");
375        continue_button.addClass("disabled");
376        pause_button.off("click");
377        pause_button.addClass("disabled");
378
379        var time = this.master_clock.getElapsedTime(now);
380
381        for (var i = 0; i < this.categories.length; i++) {
382           var category = this.categories[i];
383           category.li.off("click");
384           category.li.addClass("disabled");
385           if (category.down) {
386              addRecord(this, category.click(time));
387           }
388        }
389
390        var this_ = this;
391
392        $.ajax({
393           url: "../../webapi/records/addobservationdata",
394           type: "POST",
395           dataType: "text",
396           contentType: "application/json",
397           cache: false,
398           data: JSON.stringify({
399              duration: time,
400              timeZoneOffsetInMs: getTimeZoneOffset(),
401              daylightSavingInMs: getDaylightSaving(),
```

```
402             data: this.records
403          }),
404          success: function(data) {
405             this_.waiting = false;
406             // TODO: Redirect properly.
407             window.location = "../summary/";
408          },
409          error: function(xhr, status, error) {
410             showError(msg.obs_errorCouldntSendData + ": " + error);
411             this_.waiting = false;
412          }
413       });
414    };
415
416    /**
417     * Delegates the click of a category button to the correct category.
418     * It adds the (possible) record returned by the category to the
419     * list of all the records made during the observation.
420     * @param {number} index The index of the category button that was clicked.
421     */
422    this.categoryClick = function(index) {
423       var category = this.categories[index];
424       var time = this.master_clock.getElapsedTime(Date.now());
425       addRecord(this, category.click(time, this.master_clock.isPaused()));
426    };
427
428    /**
429     * Updates the observation clock and all the categories based on it.
430     */
431    this.tick = function() {
432       var time = this.master_clock.getElapsedTime(Date.now());
433
434       var time_str = timeToString(time);
435       var total_time = $("#total-time");
436       total_time.empty();
437       total_time.append(document.createTextNode(time_str));
438
439       for (var i = 0; i < this.categories.length; i++) {
440          this.categories[i].updateTimer(time);
441       }
442    };
443 }
444
445
446 /**
447  * Sends an AJAX keep-alive signal to the backend.
448  */
449 function keepAlive() {
450    $.ajax({
451       url: "../../webapi/records/keepalive",
452       type: "POST",
453       dataType: "text",
454       contentType: "text/plain",
455       cache: false,
456       data: "keep-alive",
457       success: function(data) {
458          //console.log("Success: " + data);
459       },
```

```
460         error: function(xhr, status, error) {
461             showError(msg.obs_errorKeepAliveFailed + ": " + error);
462         }
463     });
464 }
465
466
467 /**
468  * Shows an error message in a PrimeFaces growl.
469  * @param {String} error_msg The error message to be shown.
470  */
471 function showError(error_msg) {
472     var growl = PF("growlWdgt");
473     growl.removeAll();
474     growl.renderMessage({
475         summary: msg.dialogErrorTitle,
476         detail: error_msg,
477         severity: "error"
478     });
479 }
480
481
482 /**
483  * The function will call observer.stop().
484  * It is needed if the stopping of the observation has to be confirmed.
485  */
486 var stopObservation = function () {};
487
488
489 /**
490  * This function is ran when the document is ready.
491  * Creates observer, binds event handlers, and sets two intervals:
492  * one that updates the observer and one that sends keep alive to backend.
493  */
494 $(document).ready(function() {
495     var category_sets = getCategorySets(); // NOTE: Function in observer/index.
        xhtml.
496     var observer = new Observer(category_sets);
497
498     $("#start").click(function() {
499         observer.startClick();
500         $(".category-item").click(function() {
501             var id = $(this).attr("id");
502             var index = parseInt(id.split("_")[1]);
503             observer.categoryClick(index);
504         });
505     });
506     $("#continue").click(function() { observer.continueClick(); });
507     $("#pause").click(function() { observer.pauseClick(); });
508     stopObservation = function() { observer.stopClick(); };
509
510     setInterval(function() { observer.tick(); }, 200);
511     setInterval(keepAlive, 5*60000); // Send keep-alive every 5 minutes.
512 });
513
514 /**
515  * Gets the offset of the time zone in milliseconds (in JAVA format).
516  */
```

```
517 function getTimeZoneOffset(){
518    return -1 * 60 * 1000 * new Date().getTimezoneOffset();
519 }
520
521 /**
522  * Gets the daylight saving time offset in milliseconds.
523  */
524 function getDaylightSaving() {
525    var now = new Date();
526    var jan = new Date(now.getFullYear(), 0, 1);
527    return (jan.getTimezoneOffset() - now.getTimezoneOffset()) * 60 * 1000;
528 }
```

2.4 resources/js/summary.js

```
1
2  /* global PF, links, SummaryIndex */
3
4  /**
5   * @fileOverview Javascript methods for the summary page.
6   * @module summary
7   * @author Juha Moisio <juha.pa.moisio at student.jyu.fi>
8   */
9  var TIMELINE_BEGIN = getLocalZeroDate();
10 var OBSERVATION_DURATION = SummaryIndex.getObservationDuration(); // function in
       summary/index.xhtml
11 var msg = SummaryIndex.getMessages(); // function in summary/index.xhtml
12 var ESCAPE_KEY = 27;
13
14 /**
15  * On document ready:
16  * - Calculate recordings summary details.
17  * - Update the details on time frame change.
18  * - Add zoom button click events for timeline zooming.
19  * - Show growl message on timeline event selection.
20  */
21 $(function () {
22    var timeline = PF("timelineWdgt").getInstance();
23    var growl = PF("growlWdgt");
24    var startTimeWdgt = PF("startTimeWdgt");
25    var endTimeWdgt = PF("endTimeWdgt");
26    var timeframe = timeline.getVisibleChartRange();
27    var startTimePicker = $("#startTime_input");
28    var endTimePicker = $("#endTime_input");
29
30    timeline.options.showCurrentTime = false; // NOTE: setting this did not work
          from Summary Bean.
31
32    updateRecordsTable(timeline, timeframe);
33
34    // Set time select listeners and restore original dates that get reseted on
          event bind.
35    var startDate = startTimeWdgt.getDate();
36    var endDate = endTimeWdgt.getDate();
37    startTimePicker.timepicker("option", "onSelect", function (startTime) {
```

```
38        var error = updateTimelineTimeframe(timeline, startTime, endTimePicker.val()
              );
39        startTimePicker.toggleClass("ui-state-error", error);
40        if (error && convertStrToMs(startTime) > OBSERVATION_DURATION) {
41           startTimeWdgt.setDate(endDate);
42        }
43     });
44     startTimePicker.keyup(function () {
45        var error = updateTimelineTimeframe(timeline, startTimePicker.val(),
              endTimePicker.val());
46        $(this).toggleClass("ui-state-error", error);
47        if (error && convertStrToMs(startTimePicker.val()) > OBSERVATION_DURATION) {
48           startTimeWdgt.setDate(endDate);
49        }
50     });
51     endTimePicker.timepicker("option", "onSelect", function (endTime) {
52        var error = updateTimelineTimeframe(timeline, startTimePicker.val(), endTime
              );
53        endTimePicker.toggleClass("ui-state-error", error);
54        if (error && convertStrToMs(endTime) > OBSERVATION_DURATION) {
55           endTimeWdgt.setDate(endDate);
56        }
57     });
58     endTimePicker.keyup(function () {
59        var error = updateTimelineTimeframe(timeline, startTimePicker.val(),
              endTimePicker.val());
60        $(this).toggleClass("ui-state-error", error);
61        if (error && convertStrToMs(endTimePicker.val()) > OBSERVATION_DURATION) {
62           endTimeWdgt.setDate(endDate);
63        }
64     });
65     startTimeWdgt.setDate(startDate);
66     endTimeWdgt.setDate(endDate);
67
68     links.events.addListener(timeline, "select", function () {
69        showRecordDetails(timeline, growl);
70     });
71
72     $(document).click(function (e) {
73        if (!$(e.target).hasClass("timeline-event-content")) {
74           hideMessages(timeline, growl);
75        }
76     });
77
78     $(document).keyup(function (e) {
79        if (e.keyCode === ESCAPE_KEY) {
80           hideMessages(timeline, growl);
81        }
82     });
83
84     /* Disabled */
85     /*
86      $("#total-records").text(getRecordsInTimeframe(timeline.items, timeframe).
            length);
87      $("#total-duration").text(convertMsToUnits(OBSERVATION_DURATION));
88      $("#button-zoom-in").click(function () {
89      timeline.zoom(0.2, TIMELINE_BEGIN);
90      });
```

```
91      $("#button-zoom-out").click(function () {
92      timeline.zoom(-0.2);
93      });
94      $(window).on('scroll resize', function () {
95      $("#timelineControls").toggleClass("bottom",
96      isBottomOfDocument($("#Footer").height()));
97      });
98      $("#timelineControls").toggleClass("bottom",
99      isBottomOfDocument($("#Footer").height()));
100     */
101
102    /* Ask confirmation before leaving unsaved observation data */
103    /*
104     window.onbeforeunload = function () {
105     return msg.dlg_confirmLeave;
106     };
107     */
108 });
109
110 /**
111  * Updates the records table information according to the given time frame.
112  * @param {object} timeline - The timeline component.
113  * @param {object} timeframe - The selected start and end time.
114  */
115 function updateRecordsTable(timeline, timeframe) {
116    var recordsTable = $("#records");
117    var categories = timeline.getItemsByGroup(timeline.items);
118    var timeframeDuration = getTimeframeDuration(timeframe);
119    var recordsTotalCount = getRecordsInTimeframe(timeline.items, timeframe).length
          ;
120    recordsTable.empty();
121
122    var oldCategorySet;
123    $.each(categories, function (category, categoryRecords) {
124       var records = getRecordsInTimeframe(categoryRecords, timeframe);
125       var duration = getDurationOfRecords(records, timeframe);
126       var newCategorySet = category.match("<span class=categorySet>(.*)</span>")
             [1];
127       var recordRow = createRecordRow({
128          name: category,
129          count: records.length,
130          duration: duration,
131          addGap: oldCategorySet !== newCategorySet,
132          countPercent: spanPercentOf(records.length, recordsTotalCount),
133          durationPercent: spanPercentOf(duration, timeframeDuration)
134       });
135       recordsTable.append(recordRow);
136       oldCategorySet = newCategorySet;
137    });
138    var summaryRow = createRecordRow({
139       name: msg.sum_total,
140       count: recordsTotalCount,
141       duration: timeframeDuration,
142       countPercent: " ",
143       durationPercent: " "
144    });
145    summaryRow.addClass("summary-row");
146    recordsTable.append(summaryRow);
```

```
147 }
148
149 /**
150  * Creates a HTML element containing the data of a record.
151  * @param {object} record - The object contains record data
152  * in the form: {name, count, countPercentage, duration, durationPercentage}
153  * @returns {object} - The jquery object containing the record row element.
154  */
155 function createRecordRow(record, colcount) {
156    // TODO: escape XSS; Is it required? Values are from backing bean and are
157    // already escaped and user cannot change them later.
158    var row = $('<div class="ui-grid-row">');
159    var count = $('<div class="ui-grid-col-3">');
160    var duration = $('<div class="ui-grid-col-3">');
161    count.append('<span>' + record.count + "</span>");
162    count.append('<span>' + record.countPercent + "</span>");
163    duration.append('<span>' + convertMsToUnits(record.duration) + "</span>");
164    duration.append('<span>' + record.durationPercent + "</span>");
165    row.append('<div class="ui-grid-col-5">' + record.name + "</div>");
166    row.append(count);
167    row.append(duration);
168    if (record.addGap) {
169       row.addClass("gapBefore");
170    }
171    return row;
172 }
173
174 /**
175  * Updates the time frame of the timeline to the given start and end times.
176  * @param {object} timeline - The timeline component.
177  * @param {string} strStart - The time frame starting time in hh:mm:ss format.
178  * @param {string} strEnd - The time frame ending time in hh:mm:ss format.
179  * @returns {boolean} - returns true on errors, false if updated successfully.
180  */
181 function updateTimelineTimeframe(timeline, strStart, strEnd) {
182    var msStart = convertStrToMs(strStart);
183    var msEnd = convertStrToMs(strEnd);
184
185    // check the validity of time frame
186    if (msStart >= msEnd
187          || msStart > OBSERVATION_DURATION
188          || msEnd > OBSERVATION_DURATION) {
189       return true;
190    }
191
192    if (msStart) {
193       timeline.options.min = new Date(TIMELINE_BEGIN.getTime() + msStart);
194    } else {
195       timeline.options.min = TIMELINE_BEGIN;
196    }
197
198    if (msEnd) {
199       timeline.options.max = new Date(TIMELINE_BEGIN.getTime() + msEnd);
200    } else {
201       timeline.options.max = new Date(TIMELINE_BEGIN.getTime() +
          OBSERVATION_DURATION * 1.1);
202    }
203
```

```
204     timeline.setVisibleChartRangeAuto();
205     updateRecordsTable(timeline, timeline.getVisibleChartRange());
206     return false;
207 }
208 
209 /**
210  * Shows a PrimeFaces growl message with details of the selected record.
211  * @param {object} timeline - The timeline component.
212  * @param {object} growl - The growl component.
213  */
214 function showRecordDetails(timeline, growl) {
215     var selection = timeline.getSelection();
216     if (selection.length) {
217         if (selection[0].row !== undefined) {
218             var record = timeline.getItem(selection[0].row);
219             growl.removeAll();
220             growl.renderMessage({
221                 summary: record.group,
222                 detail: getRecordDetails(record),
223                 severity: "info"
224             });
225         }
226     }
227 }
228 
229 /**
230  * Hides all growl messages and removes timeline selection.
231  * @param {object} timeline - The timeline component.
232  * @param {object} growl - The growl component.
233  */
234 function hideMessages(timeline, growl) {
235     growl.removeAll();
236     timeline.setSelection(null);
237 }
238 
239 /**
240  * Gets all the records that are fully or partially in the given time frame.
241  * @param {object} records - The object containing the records.
242  * @param {object} timeframe - The selected start and end time.
243  * @returns {object} - returns a list of matched records.
244  */
245 function getRecordsInTimeframe(records, timeframe) {
246     var recordsIn = [];
247     $.each(records, function (i, record) {
248         if (record.className === "dummyRecord") {
249             return true;
250         } else if (record.start >= timeframe.start && record.start < timeframe.end)
                {
251             recordsIn.push(record);
252         } else if (record.end <= timeframe.end && record.end > timeframe.start) {
253             recordsIn.push(record);
254         } else if (record.start < timeframe.start && record.end > timeframe.end) {
255             recordsIn.push(record);
256         }
257     });
258     return recordsIn;
259 }
260 
```

```
261 /**
262  * Gets the record details as a string.
263  * @param {object} record - The record object from the timeline component.
264  * @returns {string} - The details as a string value.
265  */
266 function getRecordDetails(record) {
267    var details = "";
268    var start = toTimelineTime(record.start);
269    var end = toTimelineTime(record.end);
270    details += msg.sum_begin + ": " + convertMsToStr(start);
271    details += "<br/>";
272    details += msg.sum_end + ": " + convertMsToStr(end);
273    details += "<br/>";
274    details += msg.sum_duration + ": " + convertMsToUnits(end - start);
275    return details;
276 }
277
278 /**
279  * Get total duration of records of all categories in given time frame.
280  * @param {object} records - object containing the records.
281  * @param {object} timeframe - The selected start and end time.
282  * @returns {number} - duration of the records.
283  */
284 function getDurationOfCategories(categories, timeframe) {
285    var duration = 0;
286    $.each(categories, function (category, records) {
287       duration += getDurationOfRecords(records, timeframe);
288    });
289    return duration;
290 }
291
292 /**
293  * Gets the duration of the given time frame.
294  * @param {object} timeframe - The selected start and end time.
295  * @returns {number} - duration of the observation's time frame.
296  */
297 function getTimeframeDuration(timeframe) {
298    var rStartMs = toTimelineTime(timeframe.start);
299    var rEndMs = toTimelineTime(timeframe.end);
300    var start = (rStartMs > 0) ? rStartMs : 0;
301    var end = (rEndMs < OBSERVATION_DURATION) ? rEndMs : OBSERVATION_DURATION;
302    return end - start;
303 }
304
305 /**
306  * Gets the total duration of the records in the given time frame.
307  * @param {object} records - The object containing the records.
308  * @returns {number} - The duration of the records.
309  */
310 function getDurationOfRecords(records, timeframe) {
311    var duration = 0;
312    $.each(records, function () {
313       var start = this.start;
314       var end = this.end;
315       if (this.className === "dummyRecord") {
316          return true;
317       }
318       if (start < timeframe.start) {
```

```
319            start = timeframe.start;
320         }
321         if (end > timeframe.end) {
322            end = timeframe.end;
323         }
324         if (end > start) {
325            duration += end - start;
326         }
327      });
328      return duration;
329   }
330
331   /**
332    * Converts the time in milliseconds to a string hh:mm:ss.
333    * @param {number} ms - The time in milliseconds.
334    * @returns {string} - The time in string as hh:mm:ss.
335    */
336   function convertMsToStr(ms) {
337      var d = ms;
338      d = Math.floor(d / 1000);
339      var s = d % 60;
340      d = Math.floor(d / 60);
341      var m = d % 60;
342      d = Math.floor(d / 60);
343      var h = d % 60;
344      return [h, m, s].map(leadingZero).join(':');
345   }
346
347   /**
348    * Converts the time string in the form hh:mm:ss to milliseconds.
349    * @param {string} str - The time in a string as hh:mm:ss.
350    * @returns {number} - The time in milliseconds or NaN for unparseable time string.
351    */
352   function convertStrToMs(str) {
353      var time = str.split(/:/);
354      // insert missing values
355      for (var i = 3 - time.length; i > 0; i--) {
356         time.unshift("0");
357      }
358      var seconds = 0;
359      for (var i = 0; i < time.length; i++) {
360         seconds += parseInt(time[i], 10) * Math.pow(60, 2 - i);
361      }
362      return seconds * 1000;
363   }
364
365   /**
366    * Converts the time in milliseconds to a string with the time units e.g. 1h 2m 0s.
367    * @param {number} ms - The time in milliseconds.
368    * @returns {string} - The time in string with units e.g. 1h 2m 0s.
369    */
370   function convertMsToUnits(ms) {
371      var time = convertMsToStr(ms).split(":");
372      var units = "";
373      var getTimeUnit = function (i, unit) {
374         var n = parseInt(time[i], 10);
375         if (n > 0) {
376            units += n + unit;
```

```
377         }
378      };
379
380      if (ms <= 0) {
381         return "0 s";
382      }
383      if (ms < 1000) {
384         return "~1 s";
385      }
386      if (time.length === 3) {
387         getTimeUnit(0, " h");
388         getTimeUnit(1, " m");
389         getTimeUnit(2, " s");
390      } else {
391         return "0 s";
392      }
393      return units.replace(/([hms])(\d)/g, "$1 $2");
394   }
395
396   /**
397    * Returns the given number as a string and appends a leading zero
398    * to it if the number is a single digit number.
399    * @param {number} n - The given number.
400    * @returns {string} - number with possible leading zero.
401    */
402   function leadingZero(n) {
403      return (n < 10 ? "0" + n : n.toString());
404   }
405
406   /**
407    * Calculates the percentage of two values.
408    * @param {number} a - The number of share.
409    * @param {number} b - The number of total quantity.
410    * @returns {number} - percentage ratio.
411    */
412   function percentOf(a, b) {
413      if (a === 0 || b === 0) {
414         return 0;
415      }
416      return Math.round((a / b) * 100);
417   }
418
419   /**
420    * Gets the percentage of two values as a span element string.
421    * @param {number} a - The number of share.
422    * @param {number} b - The number of total quantity.
423    * @returns {string} - percent as span element string.
424    */
425   function spanPercentOf(a, b) {
426      var percent = percentOf(a, b);
427      var str = " (" + percent.toString() + "%)";
428      if (percent < 10) {
429         str = " " + str;
430      }
431      if (percent < 100) {
432         str = " " + str;
433      }
434      return '<span class="percent">' + str + "</span>";
```

```
435 }
436
437 /**
438  * Gets the "zero" date with the time zone offset.
439  * @returns {date} - The zero date with the time zone offset.
440  */
441 function getLocalZeroDate() {
442    var localDate = new Date(0);
443    var zeroDate = new Date(localDate.getTimezoneOffset() * 60 * 1000);
444    return zeroDate;
445 }
446
447 /**
448  * Converts the date object to the timeline component time.
449  * @param {date} date - The date object of the time to be converted.
450  * @returns {number} - The converted time in milliseconds.
451  */
452 function toTimelineTime(date) {
453    return Math.abs(TIMELINE_BEGIN.getTime() - date.getTime());
454 }
455
456 /**
457  * Encodes HTML markup characters to HTML entities.
458  * @param {string} str - The string to be encoded.
459  * @returns {str} - The encoded string.
460  */
461 function encodeHTML(str) {
462    return str
463          .replace(/&/g, '&amp;')
464          .replace(/</g, '&lt;')
465          .replace(/>/g, '&gt;')
466          .replace(/"/g, '&quot;');
467 }
468
469 /**
470  * Checks if the user has scrolled to the bottom of the page.
471  * @param {number} padding - An extra padding to be checked.
472  * @return {boolean} - true if at bottom otherwise false.
473  */
474 function isBottomOfDocument(padding) {
475    return $(window).scrollTop() >= $(document).height() - padding - $(window).
       height();
476 }
```

Chapter 3

XHTML pages

3.1 WEB-INF/commonPages/commonFooter.xhtml

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:h="http://xmlns.jcp.org/jsf/html"
     xmlns:ui="http://xmlns.jcp.org/jsf/facelets">

    <body>
        <ui:composition>
            <h:panelGrid styleClass="naviFooter">
                <h:outputLink value="#{msg.foot_copyUrl}">
                    <h:outputText value="#{msg.foot_copyText}" style="text-decoration:
                        underline"/>
                </h:outputLink>
            </h:panelGrid>
        </ui:composition>
    </body>
</html>
```

3.2 WEB-INF/commonPages/commonHeader.xhtml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html>
<html
    xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
    xmlns:p="http://primefaces.org/ui"
    xmlns:f="http://xmlns.jcp.org/jsf/core">

    <body>
        <ui:composition>
            <h:form>
```

```
13              <p:menubar styleClass="naviHeader" autoDisplay="false" >
14                 <f:facet name="options">
15                    <h:outputLink value="/index.xhtml">
16                    <h:outputText value="Moveatis" styleClass="facetLeft"/>
17                    </h:outputLink>
18                 </f:facet>
19
20                 <p:submenu icon="fa fa-user" label="#{sessionBean.
                       loggedIdentifiedUser.givenName}" rendered="#{sessionBean.
                       isIdentifiedUser()}">
21                    <p:menuitem value="#{msg.head_controlPage}" icon="fa fa-sitemap"
                          url="/app/control/index.xhtml"/>
22                    <p:menuitem value="#{msg.head_logout}" icon="fa fa-sign-out"
23                             actionListener="#{logoutBean.logOut}" />
24                 </p:submenu>
25
26                 <p:submenu icon="fa fa-cogs">
27                    <p:menuitem value="#{msg.head_info}" icon="fa fa-info" url="/
                          info/index.xhtml"/>
28                    <p:menuitem value="#{userManagedBean.optionLanguageString}" icon
                          ="fa fa-flag-o"
29                             actionListener="#{userManagedBean.changeLocale}" ajax="
                                false" />
30                    <p:menuitem value="#{msg.head_resetObs}" rendered="#{sessionBean
                       .isResetObsAvailable()}" icon="fa fa-eraser" url="/app/
                       observer/index.xhtml" />
31                    <p:menuitem value="#{msg.head_backToCatEd}" rendered="#{
                       sessionBean.isBackToCatEdAvailable()}" icon="fa fa-edit" url=
                       "/app/categoryselection/index.xhtml"/>
32                    <p:menuitem value="#{msg.head_toFrontPage}" rendered="#{
                       sessionBean.isToFrontPageAvailable()}" icon="fa fa-rotate-
                       left" url="/index.xhtml"/>
33                 </p:submenu>
34              </p:menubar>
35           </h:form>
36        </ui:composition>
37     </body>
38  </html>
```

3.3 WEB-INF/template.xhtml

```
1  <?xml version='1.0' encoding='UTF-8' ?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
       TR/xhtml1/DTD/xhtml1-transitional.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml"
4       xmlns:h="http://xmlns.jcp.org/jsf/html"
5       xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
6       xmlns:f="http://xmlns.jcp.org/jsf/core">
7  <f:view locale="#{userManagedBean.locale}">
8     <h:head>
9        <title>
10          #{msg.applicationTitle} <ui:insert name="pageTitle"/>
11       </title>
12       <h:outputStylesheet library="css" name="base.css"/>
13       <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
```

```
14    </h:head>
15
16    <h:body>
17       <div class="wrapper">
18          <div id="naviHead">
19             <ui:insert name="header">
20                <ui:include src="commonPages/commonHeader.xhtml"/>
21             </ui:insert>
22
23          </div>
24
25          <div id="mainPage">
26             <ui:insert name="mainpage">
27                Main content
28             </ui:insert>
29
30          </div>
31
32          <div class="push"></div>
33       </div>
34
35       <div id="Footer">
36          <ui:insert name="footer" >
37             <ui:include src="commonPages/commonFooter.xhtml"/>
38          </ui:insert>
39
40       </div>
41    </h:body>
42 </f:view>
43 </html>
```

3.4 app/categoryselection/index.xhtml

```
1  <?xml version='1.0' encoding='UTF-8' ?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
       TR/xhtml1/DTD/xhtml1-transitional.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml"
4       xmlns:h="http://xmlns.jcp.org/jsf/html"
5       xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
6       xmlns:p="http://primefaces.org/ui"
7       xmlns:f="http://xmlns.jcp.org/jsf/core">
8     <h:body>
9        <ui:composition template="/WEB-INF/template.xhtml">
10
11          <ui:define name="pageTitle">#{msg.cs_title}</ui:define>
12
13          <ui:define name="mainpage">
14             <h:outputStylesheet library="css" name="categoryselection.css"/>
15
16             <div id="container">
17                <h:panelGroup layout="block" rendered="#{empty
                      categorySelectionBean.eventGroup}">
18                <h1>#{msg.cs_heading}</h1>
19                </h:panelGroup>
20                <h:panelGroup layout="block" rendered="#{not empty
                      categorySelectionBean.eventGroup}">
```

```
21                  <h1>
22                  <h:outputFormat value="#{msg.cs_headingEvent}">
23                     <f:param value="#{categorySelectionBean.eventGroup.label}"/>
24                  </h:outputFormat>
25                  </h1>
26               </h:panelGroup>
27
28               <h2>#{msg.cs_addedCategorySets}</h2>
29
30               <h:form id="form" onkeypress="return captureEnter(event);">
31                  <p:growl id="growl" showDetail="true"/>
32
33                  <p:confirmDialog global="true">
34                     <p:commandButton value="#{msg.dialogConfirmYes}" type="button
                        " styleClass="ui-confirmdialog-yes" icon="ui-icon-check"
                        />
35                     <p:commandButton value="#{msg.dialogConfirmNo}" type="button"
                         styleClass="ui-confirmdialog-no" icon="ui-icon-close" />
36                  </p:confirmDialog>
37
38
39                  <h:outputText value="#{msg.cs_noCategorySetsText}" rendered="#{
                     empty categorySelectionBean.categorySetsInUse}"/>
40
41                  <ui:repeat value="#{categorySelectionBean.categorySetsInUse}"
                     var="categorySet" varStatus="status">
42                  <h:panelGroup id="category-set" styleClass="category-set
                     category-set-#{status.index}">
43
44                     <p:commandLink styleClass="fa fa-remove category-set-remove-
                        button"
45                                action="#{categorySelectionBean.removeCategorySet(
                                   categorySet)}" update="@form">
46                        <p:confirm header="#{msg.dialogConfirmTitle}" message="#{
                           msg.cs_confirmRemoveCategorySet}" icon="ui-icon-alert"
                           />
47                     </p:commandLink>
48
49                     <h2>#{categorySet.name}</h2>
50
51                     <h:panelGroup layout="block" styleClass="no-categories-text"
                        rendered="#{empty categorySet.categories}">
52                        <h:outputText value="#{msg.cs_noCategoriesText}"/>
53                     </h:panelGroup>
54
55                     <ui:repeat value="#{categorySet.categories}" var="category">
56                     <div class="category">
57                        <p:selectBooleanButton value="#{category.typeAsBoolean}"
58                                         onLabel="#{msg.cs_countedCategory}"
59                                         offLabel="#{msg.cs_timedCategory}"
60                                         styleClass="category-type-button"/>
61                        <p:inputText value="#{category.name}" id="category-name-
                           input" styleClass="category-text"/>
62                        <p:watermark value="#{msg.cs_categoryName}" for="category-
                           name-input"/>
63                        <p:commandButton icon="fa fa-remove" class="category-
                           remove-button"
```

```
64                                 action="#{categorySet.remove(category)}"
                                       update="@form"/>
65                         </div>
66                         </ui:repeat>
67
68                         <p:commandButton value="#{msg.cs_newCategory}" icon="fa fa-
                               plus" styleClass="new-category-button"
69                                 action="#{categorySet.addEmpty}" update="category
                                       -set :form:continue-button"
70                                 oncomplete="focusCategory('.category-set-#{status
                                       .index}');"/>
71                 </h:panelGroup>
72                 </ui:repeat>
73
74                 <div id="continue-div">
75                    <p:commandButton value="#{msg.cs_continueToObservation}" id="
                          continue-button"
76                                 disabled="#{categorySelectionBean.
                                       continueDisabled}"
77                                 action="#{categorySelectionBean.checkCategories}"
                                        update="@form"/>
78                 </div>
79
80                 <h2>#{msg.cs_addCategorySets}</h2>
81
82                 <h:panelGroup layout="block" styleClass="category-set-list">
83                    <p:outputLabel for="new-category-set" value="#{msg.
                          cs_newCategorySet}:" styleClass="category-set-list-label"
                          />
84                    <p:inputText value="#{categorySelectionBean.
                          newCategorySetName}" id="new-category-set" styleClass="
                          category-set-list-input" />
85                    <p:watermark value="#{msg.cs_categorySetName}" for="new-
                          category-set" />
86                    <p:commandButton value="#{msg.cs_addCategorySet}" style="
                          vertical-align: middle"
87                                 action="#{categorySelectionBean.addNewCategorySet
                                       }" update="@form"/>
88                 </h:panelGroup>
89
90                 <h:panelGroup layout="block" styleClass="category-set-list"
91                           rendered="#{not empty categorySelectionBean.
                                 privateCategorySets}">
92                    <p:outputLabel for="private-category-sets" value="#{msg.
                          cs_privateCategorySets}:"
93                                  styleClass="category-set-list-label"/>
94                    <p:selectOneMenu id="private-category-sets" value="#{
                          categorySelectionBean.selectedPrivateCategorySet}"
95                                 styleClass="category-set-list-select">
96                       <f:selectItems value="#{categorySelectionBean.
                             privateCategorySets}" var="categorySet"
97                                   itemLabel="#{categorySet.name}" itemValue="#{
                                       categorySet.id}"/>
98                    </p:selectOneMenu>
99                    <p:commandButton value="#{msg.cs_addCategorySet}" style="
                          vertical-align: middle"
100                                action="#{categorySelectionBean.
                                      addPrivateCategorySet}" update="@form"/>
```

```
101                     </h:panelGroup>
102
103                     <h:panelGroup layout="block" styleClass="category-set-list-last"
104                               rendered="#{not empty categorySelectionBean.
                                     defaultCategorySets}">
105                        <p:outputLabel for="default-category-sets" value="#{msg.
                              cs_defaultCategorySets}:"
106                                       styleClass="category-set-list-label"/>
107                        <p:selectOneMenu id="default-category-sets" value="#{
                              categorySelectionBean.selectedDefaultCategorySet}"
108                                        styleClass="category-set-list-select">
109                           <f:selectItems value="#{categorySelectionBean.
                                 defaultCategorySets}" var="categorySet"
110                                          itemLabel="#{categorySet.name}" itemValue="#{
                                             categorySet.id}"/>
111                        </p:selectOneMenu>
112                        <p:commandButton value="#{msg.cs_addCategorySet}" style="
                              vertical-align: middle"
113                                        action="#{categorySelectionBean.
                                             addDefaultCategorySet}" update="@form"/>
114                     </h:panelGroup>
115                  </h:form>
116               </div>
117
118               <h:outputScript>
119                  function focusCategory(category_set_class) {
120                     var category_set = $(category_set_class);
121                     if (category_set.length) {
122                        setTimeout(function () {
123                           category_set.find(".category-text").last().focus();
124                        }, 200);
125                     }
126                  }
127                  function captureEnter(event) {
128                     if (event.keyCode !== 13)
129                        return true;
130                     var focused = $(":focus");
131                     if (focused.hasClass("category-text")) {
132                        focused.blur();
133                        var button = focused.parent().next("button");
134                        if (button.length) button.click();
135                     } else if (focused.hasClass("category-set-list-input")) {
136                        var button = focused.next("button");
137                        if (button.length) button.click();
138                     }
139                     return false;
140                  }
141               </h:outputScript>
142            </ui:define>
143         </ui:composition>
144      </h:body>
145   </html>
```

3.5 app/control/index.xhtml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:h="http://xmlns.jcp.org/jsf/html"
     xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
     xmlns:p="http://primefaces.org/ui"
     xmlns:f="http://xmlns.jcp.org/jsf/core">

    <h:body>
       <ui:composition template="/WEB-INF/template.xhtml" >

          <f:event type="preRenderView"
                 listener="#{facesContext.externalContext.response.setHeader('Cache-
                     Control', 'no-cache, no-store')}"/>

          <ui:define name="pageTitle"> - #{msg.con_title}</ui:define>

          <ui:define name="mainpage">
             <h:outputStylesheet library="css" name="control.css"/>
             <h:outputScript library="js" name="control.js"/>

             <h1>#{msg.con_title}</h1>
             <p>#{msg.con_introduction}</p>

             <h2>#{msg.con_titleOwnEvents}</h2>
             <h:form id="eventGroupsForm">
                <p:dataTable id="eventGroupsTable" var="eventGroup" value="#{
                    controlBean.eventGroups}" widgetVar="eventsWdgt"
                       selectionMode="single" selection="#{controlBean.
                           selectedEventGroup}" rowKey="#{eventGroup.id}"
                       editable="true" styleClass="eventGroupsTable"
                       emptyMessage="#{msg.con_emptyMessageEvents}"
                       sortBy="#{eventGroup.id}" reflow="true">

                   <p:ajax event="rowEdit" listener="#{controlBean.onEditEventGroup
                       }"/>
                   <p:ajax event="rowSelect" update=":startObservationForm"/>
                   <p:ajax event="rowUnselect" update=":startObservationForm"/>

                   <p:column styleClass="row-toggler">
                      <p:rowToggler expandLabel="&#xf138;" collapseLabel="&#xf13a;"
                          />
                   </p:column>

                   <p:column headerText="#{msg.con_event}" styleClass="column-name"
                       >
                      <p:cellEditor>
                         <f:facet name="output">
                            <h:outputText value="#{eventGroup.label}" title="#{
                                eventGroup.description}"/>
                         </f:facet>
                         <f:facet name="input">
                            <p:message for="input-name-eg" />
                            <p:inputText id="input-name-eg" required="true"
                                requiredMessage="#{msg.dlg_notEmpty}"
                                      placeholder="#{msg.con_eventName}" styleClass=
                                          "input-name"
                                      value="#{eventGroup.label}"
```

```
50                                validator="#{validationBean.
                                     validateShortString}"/>
51                        <p:message for="input-description-eg" />
52                        <p:inputTextarea id="input-description-eg" styleClass="
                             input-description"
53                                 placeholder="#{msg.con_eventDescription}"
54                                 value="#{eventGroup.description}"
55                                 validator="#{validationBean.
                                     validateLongString}"
56                                 rows="2" cols="10" autoResize="true" />
57                  </f:facet>
58               </p:cellEditor>
59            </p:column>
60
61            <p:column headerText="#{msg.con_categorySets}" styleClass="
                 column-category-groups">
62               <h:outputText rendered="#{eventGroup.categorySets.size() ==
                     0}" value=""/>
63               <ul class="list-categorysets">
64                  <ui:repeat value="#{eventGroup.categorySets.toArray()}"
                       var="categorySet">
65                     <li>
66                        <p:commandLink value="#{categorySet.label}" title="
                             #{categorySet.description}"
67                                 action="#{controlBean.
                                     setSelectedCategorySet(categorySet)}"
68                                 onclick="PF('dlgCategorySet').show()"
                                     styleClass="link-categoryset"
69                                 update=":categorySetForm :eventGroupsForm
                                     :eventGroupsTable :
                                     startObservationForm">
70                           <f:actionListener binding="#{controlBean.
                                setSelectedEventGroup(eventGroup)}"/>
71                        </p:commandLink>
72                     </li>
73                  </ui:repeat>
74                  <li>
75                     <p:commandLink styleClass="fa fa-plus link-new-
                          categoryset link-categoryset"
76                             action="#{controlBean.setSelectedEventGroup(
                                 eventGroup)}"
77                             oncomplete="PF('dlgCategorySet').show()"
78                             update=":categorySetForm :eventGroupsForm:
                                 eventGroupsTable">
79                        <span>#{msg.dlg_add}</span>
80                        <f:actionListener binding="#{controlBean.
                             addNewCategorySet()}" />
81                     </p:commandLink>
82                  </li>
83               </ul>
84            </p:column>
85
86            <p:column headerText="#{msg.con_key}" styleClass="column-group-
                 keys">
87               <p:commandLink styleClass="fa fa-key new-key"
88                           action="#{controlBean.setSelectedEventGroup(
                               eventGroup)}"
89                           rendered="#{!controlBean.hasGroupKey(eventGroup)}"
```

```
90                              title="#{msg.con_setGroupKey}"
91                              onclick="PF('dlgNewKey').show();"
92                              update=":newKeyForm :eventGroupsForm:
                                    eventGroupsTable"/>
93                        <p:commandLink class="fa fa-key edit-key"
94                              action="#{controlBean.setSelectedEventGroup(
                                    eventGroup)}"
95                              rendered="#{controlBean.hasGroupKey(eventGroup)}"
96                              title="#{msg.con_manageGroupKey}"
97                              onclick="PF('dlgEditKey').show();"
98                              update=":editKeyForm :eventGroupsForm:
                                    eventGroupsTable"/>
99                     </p:column>
100
101                    <p:column headerText="#{msg.dlg_edit}" styleClass="row-editor">
102                       <p:rowEditor styleClass="fa"
103                                 editTitle="#{msg.con_editEvent}"
104                                 saveTitle="#{msg.dlg_save}"
105                                 cancelTitle="#{msg.dlg_cancel}"/>
106                    </p:column>
107
108                    <p:column headerText="#{msg.dlg_remove}" styleClass="column-
                          remove">
109                       <p:commandLink styleClass="fa fa-trash"
110                                 action="#{controlBean.removeEventGroup(eventGroup)
                                       }"
111                                 title="#{msg.con_removeEvent}"
112                                 update=":eventGroupsForm:eventGroupsTable">
113                          <p:confirm header="#{msg.dlg_confirmRemoveDialog}"
114                                 message="#{msg.dlg_confirmRemoveMessage} #{
                                       eventGroup.label}. #{msg.dlg_confirmMessage}"
                                       />
115                       </p:commandLink>
116                    </p:column>
117
118                    <p:column styleClass="row-toggler-mobile">
119                       <p:rowToggler expandLabel="#{msg.con_showObservations}"
                             collapseLabel="#{msg.con_hideObservations}"/>
120                    </p:column>
121
122                    <p:rowExpansion>
123                       <p:dataList value="#{controlBean.getObservations(eventGroup)}
                             "
124                             var="observation" emptyMessage="#{msg.
                                   con_emptyMessageObservations}"
125                             lazy="true">
126                          <f:facet name="header">
127                             #{msg.con_eventObservations}
128                          </f:facet>
129                          <p:commandLink update=":observationDetail :
                                observationFormCommands" oncomplete="PF('dlgObservation
                                ').show()"
130                                   title="#{msg.con_showObservationDetails}"
                                         styleClass="link-observation">
131                             <f:setPropertyActionListener value="#{observation}"
                                   target="#{controlBean.selectedObservation}" />
132                             <h:outputText value="#{observation.name}"/>
133                          </p:commandLink>
```

```
134                     </p:dataList>
135                 </p:rowExpansion>
136             </p:dataTable>
137         </h:form>
138
139         <p:dialog header="#{msg.con_observationDetails}" widgetVar="
                dlgObservation"
140             position="top" resizable="false" responsive="true"
141             modal="true" fitViewport="true" styleClass="coloredDialog
                    observation-dialog">
142             <h:form id="observationDetail">
143                 <p:panelGrid layout="grid" columns="2" rendered="#{not empty
                        controlBean.selectedObservation}" columnClasses="label,value"
                        >
144
145                     <h:outputText value="#{msg.con_observationDetailsName}" />
146                     <p:outputPanel styleClass="editable">
147                         <p:message for="input-name-observation" />
148                         <p:inplace id="input-name-observation" editor="true"
149                                 saveLabel="#{msg.dlg_save}" cancelLabel="#{msg.
                                        dlg_cancel}">
150                             <p:ajax event="save" listener="#{controlBean.
                                    onEditObservation}" update=":eventGroupsForm" />
151                             <p:inputText required="true" requiredMessage="#{msg.
                                    dlg_notEmpty}"
152                                     value="#{controlBean.selectedObservation.name}
                                        "
153                                     validator="#{validationBean.
                                        validateShortString}"/>
154                         </p:inplace>
155                     </p:outputPanel>
156
157                     <h:outputText value="#{msg.con_observationDetailsDescription}
                        " />
158                     <p:outputPanel styleClass="editable">
159                         <p:message for="input-description-observation" />
160                         <p:inplace id="input-description-observation" editor="true
                            "
161                                 emptyLabel="#{msg.
                                        con_observationDetailsDescriptionEmptyLabel}"
162                                 saveLabel="#{msg.dlg_save}" cancelLabel="#{msg.
                                        dlg_cancel}">
163                             <p:ajax event="save" listener="#{controlBean.
                                    onEditObservation}" update=":eventGroupsForm" />
164                             <p:inputTextarea value="#{controlBean.
                                    selectedObservation.description}"
165                                     validator="#{validationBean.
                                        validateLongString}"
166                                     rows="2" cols="16" autoResize="true" />
167                         </p:inplace>
168                     </p:outputPanel>
169
170                     <h:outputText value="#{msg.con_observationDetailsObserver}"
                        />
171                     <h:outputText value="#{controlBean.getObserverName()}" />
172
173                     <h:outputText value="#{msg.con_observationDetailsTarget}" />
174                     <p:outputPanel styleClass="editable">
```

```
175                     <p:message for="input-target-observation" />
176                     <p:inplace id="input-target-observation" editor="true"
177                             emptyLabel="#{msg.
                                con_observationDetailsTargetEmptyLabel}"
178                             saveLabel="#{msg.dlg_save}" cancelLabel="#{msg.
                                dlg_cancel}">
179                        <p:ajax event="save" listener="#{controlBean.
                           onEditObservation}" update=":eventGroupsForm" />
180                        <p:inputText value="#{controlBean.selectedObservation.
                           target}"
181                              validator="#{validationBean.
                                 validateShortString}"/>
182                     </p:inplace>
183                  </p:outputPanel>
184
185                  <h:outputText value="#{msg.con_observationDetailsDuration}"
                        />
186                  <h:outputText value="#{controlBean.msToUnits(controlBean.
                        selectedObservation.duration)}" />
187
188                  <h:outputText value="#{msg.con_observationDetailsRecordAmount
                        }" />
189                  <h:outputText value="#{controlBean.selectedObservation.
                        records.size()}" />
190               </p:panelGrid>
191            </h:form>
192            <h:form id="observationFormCommands">
193               <p:commandButton value="#{msg.con_showSummary}" icon="fa fa-bar-
                     chart"
194                           action="#{controlBean.showObservationInSummaryPage()
                              }"/>
195               <p:commandButton value="#{msg.con_removeObservation}" icon="fa
                     fa-trash"
196                           action="#{controlBean.removeObservation}"
197                           oncomplete="onDialogSuccess(args, 'dlgObservation',
                              '#{msg.con_observationRemoved}')"
198                           update=":eventGroupsForm:eventGroupsTable">
199                  <p:confirm header="#{msg.dlg_confirmRemoveDialog}"
200                          message="#{msg.dlg_confirmRemoveMessage} #{
                             controlBean.selectedObservation.name}. #{msg.
                             dlg_confirmMessage}"/>
201               </p:commandButton>
202               <p:commandButton value="#{msg.con_closeObservationDetails}" icon
                     ="ui-icon-close"
203                           onclick="PF('dlgObservation').hide();" />
204            </h:form>
205         </p:dialog>
206
207         <h:panelGroup id="newEventGroupPanel">
208            <h:form rendered="#{controlBean.creatingNewEventGroup}" styleClass=
                  "new-eventgroup-form">
209               <p:focus for="input-name-eventgroup"/>
210               <p:message for="input-name-eventgroup" />
211               <p:inputText id="input-name-eventgroup" required="true"
                     requiredMessage="#{msg.dlg_notEmpty}"
212                           value="#{eventGroupManagedBean.eventGroupName}"
                              styleClass="input-name"
213                           placeholder="#{msg.con_eventName}"
```

```
                        validator="#{validationBean.validateShortString}"/>

                <p:commandButton value="#{msg.dlg_add}" icon="fa fa-plus"
                            action="#{eventGroupManagedBean.createNewEventGroup}
                                "
                            update="@form :eventGroupsForm:eventGroupsTable :
                                newEventGroupPanel :newEventGroupButton" process
                                ="@form"
                            validateClient="true"/>
            </h:form>

            <p:commandButton value="#{msg.dlg_cancel}" icon="fa fa-close"
                        rendered="#{controlBean.creatingNewEventGroup}"
                        action="#{controlBean.setCreatingNewEventGroup(false)}"
                        update=":newEventGroupPanel :newEventGroupButton"/>
        </h:panelGroup>

        <h:form id="newEventGroupButton">
            <p:commandButton value="#{msg.con_newEvent}" icon="fa fa-plus"
                        rendered="#{!controlBean.creatingNewEventGroup}"
                        action="#{controlBean.setCreatingNewEventGroup(true)}"
                        update="newEventGroupPanel @form" />
        </h:form>

        <h:form id="startObservationForm">
            <p:commandButton value="#{msg.con_startObservationForEvent}" icon="
                fa fa-caret-square-o-right"
                        action="#{controlBean.newObservation}" styleClass="
                            start-button"
                        disabled="#{empty controlBean.selectedEventGroup}"/>
        </h:form>

        <p:outputPanel rendered="#{controlBean.otherObservations.size() > 0}">
            <h2>#{msg.con_titleOtherObservations}</h2>
            <p:fieldset legend="#{msg.con_showHide}" collapsed="true"
                toggleable="true"
                     styleClass="other-observations-fieldset">
                <h:form id="otherObservationsForm">
                    <p:dataTable id="otherObservations" value="#{controlBean.
                        otherObservations}" var="otherObservation"
                            selectionMode="single" selection="#{controlBean.
                                selectedObservation}"
                            rowKey="#{otherObservation.id}" sortBy="#{
                                otherObservation.id}" reflow="true"
                            lazy="true">

                        <p:ajax process="@this" event="rowSelect" update=":
                            observationDetail :observationFormCommands" oncomplete=
                            "PF('dlgObservation').show()" />

                        <p:column headerText="#{msg.con_name}" styleClass="column-
                            other-observation-name">
                            <h:outputText value="#{otherObservation.name}"/>
                        </p:column>

                        <p:column headerText="#{msg.con_event}" styleClass="column
                            -other-eventgroup-name">
```

```
                        <h:outputText value="#{controlBean.
                            getObservationEventGroupName(otherObservation)}"/>
                    </p:column>
                </p:dataTable>
            </h:form>
        </p:fieldset>
    </p:outputPanel>

    <p:confirmDialog global="true" closable="false" styleClass="
        coloredButton">
        <p:commandButton value="#{msg.dlg_no}" type="button" styleClass="ui
            -confirmdialog-no" icon="ui-icon-close" />
        <p:commandButton value="#{msg.dlg_yes}" type="button" styleClass="
            ui-confirmdialog-yes" icon="ui-icon-check" />
    </p:confirmDialog>


    <p:dialog header="#{msg.con_setGroupKey}" widgetVar="dlgNewKey"
            position="top" resizable="false" responsive="true"
            modal="true" fitViewport="true" styleClass="coloredDialog key-
                dialog ui-fluid">
        <h:form id="newKeyForm">
            <p:inputText id="input-new-groupkey" autocomplete="off"
                        value="#{eventGroupManagedBean.eventGroupKey}"
                        validator="#{validationBean.validateGroupKey}"/>
            <p:commandButton value="#{msg.con_ok}" icon="ui-icon-check"
                            action="#{eventGroupManagedBean.addGroupKey(
                                controlBean.selectedEventGroup)}"
                            oncomplete="onDialogSuccess(args, 'dlgNewKey', '#{
                                msg.con_keySet}')"
                            update="input-new-groupkey :validationErrorGrowl :
                                eventGroupsForm:eventGroupsTable" validateClient
                                ="true"/>
        </h:form>
        <p:commandButton value="#{msg.dlg_cancel}" icon="ui-icon-close"
                        onclick="PF('dlgNewKey').hide();" />
    </p:dialog>


    <p:dialog header="#{msg.con_manageGroupKey}" widgetVar="dlgEditKey"
            position="top" resizable="false" responsive="true"
            modal="true" fitViewport="true" styleClass="coloredDialog key-
                dialog ui-fluid">
        <h:form id="editKeyForm">
            <p:inputText id="input-edit-groupkey" autocomplete="off"
                        value="#{controlBean.selectedEventGroup.groupKey.
                            groupKey}"
                        validator="#{validationBean.validateGroupKey}"/>
            <p:commandButton value="#{msg.dlg_change}" icon="ui-icon-check"
                            action="#{eventGroupManagedBean.editGroupKey(
                                controlBean.selectedEventGroup, controlBean.
                                selectedEventGroup.groupKey.groupKey)}"
                            oncomplete="onDialogSuccess(args, 'dlgEditKey', '#{
                                msg.con_keyChanged}')"
                            update="input-edit-groupkey :validationErrorGrowl"
                                validateClient="true"/>
        </h:form>
        <h:form id="removeKeyForm">
```

```
301                <p:commandButton value="#{msg.dlg_remove}" icon="ui-icon-trash"
302                             action="#{eventGroupManagedBean.removeGroupKey(
                                    controlBean.selectedEventGroup)}"
303                             oncomplete="onDialogSuccess(args, 'dlgEditKey', '#{
                                    msg.con_keyRemoved}')"
304                             update=":eventGroupsForm:eventGroupsTable">
305                   <p:confirm header="#{msg.dlg_confirmRemoveDialog}"
306                           message="#{msg.con_confirmRemoveGroupKeyMessage}"/>
307                </p:commandButton>
308             </h:form>
309          </p:dialog>
310
311          <p:dialog header="#{msg.con_categorySetDetails}" widgetVar="
                dlgCategorySet"
312                 position="top" resizable="false" responsive="true"
313                 modal="true" fitViewport="true" class="coloredDialog">
314             <h:form id="categorySetForm">
315                <div class="ui-fluid">
316                   <h:outputLabel for="input-name-categoryset" value="#{msg.
                          con_categorySetDetailsName}" />
317                   <p:message for="input-name-categoryset" />
318                   <p:inputText id="input-name-categoryset" required="true"
                          requiredMessage="#{msg.dlg_notEmpty}"
319                             value="#{controlBean.selectedCategorySet.label}"
320                             validator="#{validationBean.validateShortString}"/>
321
322                   <h:outputLabel for="input-description-categoryset" value="#{
                          msg.con_categorySetDetailsDescription}" />
323                   <p:message for="input-description-categoryset" />
324                   <p:inputTextarea id="input-description-categoryset"
325                               value="#{controlBean.selectedCategorySet.
                                      description}"
326                               validator="#{validationBean.validateLongString}"
327                               rows="2" cols="16" autoResize="true" />
328                </div>
329
330                <p:dataTable id="categoriesTable" var="category" value="#{
                       controlBean.categories}"
331                         selectionMode="single" selection="#{controlBean.
                             selectedCategory}" rowKey="#{category.orderNumber}"
332                         editable="true" editMode="row" reflow="true"
                             draggableRows="false" styleClass="categories-table"
333                         sortBy="#{category.orderNumber}" emptyMessage="#{msg.
                             con_emptyMessageCategories}"
334                         lazy="true">
335
336                   <p:ajax event="rowSelect" update=":categorySetForm:
                          categoryRemoveButton"/>
337                   <p:ajax event="rowUnselect" update=":categorySetForm:
                          categoryRemoveButton"/>
338                   <p:ajax event="rowReorder" listener="#{controlBean.
                          onCategoryReorder}" />
339
340                   <p:message for="input-name-category" />
341                   <p:column headerText="#{msg.con_categorySetDetailsCategory}">
342                      <p:cellEditor>
343                         <f:facet name="output"><h:outputText class="output-name
                                " value="#{category.label.text}"/></f:facet>
```

```
344                         <f:facet name="input">
345                            <p:inputText id="input-name-category"
346                                     required="true" requiredMessage="#{msg.
                                        dlg_notEmpty}"
347                                     value="#{category.label.text}"
348                                     validator="#{validationBean.
                                        validateShortString}"/>
349                         </f:facet>
350                      </p:cellEditor>
351                   </p:column>
352                   <p:column headerText="#{msg.con_categorySetDetailsType}"
353                          styleClass="column-categorytype">
354                      <p:cellEditor>
355                         <f:facet name="output"><h:outputText value="#{msg['con_
                              ' += category.categoryType]}" /></f:facet>
356                         <f:facet name="input">
357                            <p:selectOneMenu value="#{category.categoryType}">
358                               <f:selectItems value="#{controlBean.categoryTypes
                                    }" var="type"
359                                       itemValue="#{type}" itemLabel="#{msg['
                                          con_' += type]}" />
360                            </p:selectOneMenu>
361                         </f:facet>
362                      </p:cellEditor>
363                   </p:column>
364
365                   <p:column styleClass="row-editor">
366                      <p:rowEditor styleClass="fa"
367                                editTitle="#{msg.con_editCategory}"
368                                saveTitle="#{msg.dlg_save}"
369                                cancelTitle="#{msg.dlg_cancel}"/>
370                   </p:column>
371
372                </p:dataTable>
373                <div class="categorySetCommandButtons">
374                   <p:commandButton value="#{msg.con_addCategory}" icon="fa fa-
                         plus"
375                                action="#{controlBean.addNewCategory}"
376                                onclick="submitDataTableEditInputs('.categories-
                                    table')"
377                                oncomplete="focusDataTableEditInput('.categories-
                                    table');"
378                                update="categoriesTable" validateClient="true"/>
379                   <p:commandButton value="#{msg.con_removeCategory}" icon="fa
                         fa-minus"
380                                id="categoryRemoveButton"
381                                action="#{controlBean.removeCategory}"
382                                update="categoriesTable"
383                                disabled="#{empty controlBean.selectedCategory}"
                                    />
384                   <br/>
385                   <br/>
386                   <p:outputPanel rendered="#{!empty controlBean.
                         selectedCategorySet.id}">
387                      <p:commandButton value="#{msg.con_removeCategorySet}" icon
                            ="fa fa-trash"
388                                action="#{controlBean.removeCategorySet}"
389
```

```
                                    oncomplete="onDialogSuccess(args, '
                                        dlgCategorySet', '#{msg.
                                        con_categorySetRemoved}')"
                                    update=":eventGroupsForm:eventGroupsTable">
                                <p:confirm header="#{msg.dlg_confirmRemoveDialog}"
                                        message="#{msg.dlg_confirmRemoveMessage} #{
                                            controlBean.selectedCategorySet.label}. #{
                                            msg.dlg_confirmMessage}"/>
                            </p:commandButton>
                            <br/>
                            <br/>
                        </p:outputPanel>
                        <p:commandButton value="#{msg.con_categorySetDetailsSave}"
                            icon="ui-icon-check"
                                    action="#{controlBean.saveCategorySet}"
                                    onclick="submitDataTableEditInputs('.categories-
                                        table')"
                                    oncomplete="onDialogSuccess(args, 'dlgCategorySet
                                        ', '#{msg.con_categorySetSaved}')"
                                    update=":eventGroupsForm:eventGroupsTable @form :
                                        validationErrorGrowl" validateClient="true"/>
                        <p:commandButton value="#{msg.con_categorySetDetailsCancel}"
                            icon="ui-icon-close"
                                    onclick="PF('dlgCategorySet').hide();" />
                    </div>
                </h:form>
            </p:dialog>

            <p:growl id="validationErrorGrowl" showDetail="true" severity="error"
                life="3000"/>
            <p:growl showDetail="false" widgetVar="growlWdgt" severity="info" life
                ="3000"/>

        </ui:define>
    </ui:composition>
  </h:body>
</html>
```

3.6 app/observer/index.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
    TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
    xmlns:p="http://primefaces.org/ui">
  <h:body>
    <ui:composition template="/WEB-INF/template.xhtml">

        <ui:define name="pageTitle">#{msg.obs_title}</ui:define>

        <ui:define name="mainpage">
            <script type="text/javascript">
                function getMessages() {
```

```
15                  return {
16                     dialogErrorTitle: "#{msg.dialogErrorTitle}",
17                     countAbbreviation: "#{msg.countAbbreviation}",
18                     obs_errorCouldntSendStart: "#{msg.obs_errorCouldntSendStart}"
                          ,
19                     obs_errorCouldntSendData: "#{msg.obs_errorCouldntSendData}",
20                     obs_errorKeepAliveFailed: "#{msg.obs_errorKeepAliveFailed}"
21                  };
22               }
23               function getCategoryTypes() {
24                  return Object.freeze({
25                     TIMED: 0,
26                     COUNTED: 1
27                  });
28               }
29               function getCategorySets() {
30                  return [
31                  <ui:repeat value="#{observationBean.categorySetsInUse}" var="
                       categorySet">
32                     {name: "#{categorySet.name}", categories: [
33                     <ui:repeat value="#{categorySet.categories}" var="category">
34                        { name: "#{category.name}", type: #{category.getTypeAsInt
                             ()}, id: #{category.tag} },
35                     </ui:repeat>
36                     ]},
37                  </ui:repeat>
38                  ];
39
40               }
41            </script>
42
43            <h:outputStylesheet library="css" name="observer.css"/>
44            <h:outputScript library="primefaces" name="jquery/jquery.js" />
45            <h:outputScript library="js" name="observer.js"/>
46
47            <p:growl id="growl" widgetVar="growlWdgt" showDetail="true" severity="
                 error"/>
48
49            <p:confirmDialog global="true" style="font-size: 1.1rem">
50               <p:commandButton value="#{msg.dialogConfirmYes}" type="button"
                     styleClass="ui-confirmdialog-yes" icon="ui-icon-check" />
51               <p:commandButton value="#{msg.dialogConfirmNo}" type="button"
                     styleClass="ui-confirmdialog-no" icon="ui-icon-close" />
52            </p:confirmDialog>
53
54            <div id="content" class="no-text-select">
55               <div id="total-time"></div>
56               <div id="controls">
57                  <div id="start" class="control-button"><span class="fa fa-play"
                        ></span> #{msg.obs_start}</div>
58                  <div id="continue" class="control-button"><span class="fa fa-
                        play"></span> #{msg.obs_continue}</div>
59                  <div id="pause" class="control-button"><span class="fa fa-pause"
                        ></span> #{msg.obs_pause}</div>
60                  <div id="stop-disabled" class="control-button disabled"><span
                        class="fa fa-stop"></span> #{msg.obs_stop}</div>
61                  <p:commandLink id="stop" class="control-button" onclick="
                        stopObservation();">
```

```
                        <span class="fa fa-stop"></span> #{msg.obs_stop}
                        <!--<p:confirm header="#{msg.dialogConfirmTitle}" message="#{
                            msg.obs_confirmStopObservation}" icon="ui-icon-alert"/>-->
                    </p:commandLink>
                </div>
                <div id="category-list"></div>
            </div>
        </ui:define>

    </ui:composition>
  </h:body>
</html>
```

3.7 app/settings/index.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
    TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
    xmlns:p="http://primefaces.org/ui">


  <h:body>
    <ui:composition template="/WEB-INF/template.xhtml">

        <ui:define name="pageTitle">Asetukset</ui:define>

        <ui:define name="mainpage">
            <h:outputStylesheet library="css" name="settings.css"/>
            <nav>
                <div>
                    <h:form>
                        <div class="accessButtons" style="min-height: 200px;">
                            <p:commandButton id="languageButton" value="Kieli" icon="
                                fa fa-chevron-down" style="width: 200px; margin-top: 15
                                px" class="coloredButton">
                                <p:menu overlay="true" trigger="languageButton" my="
                                    left top" at="left bottom">
                                    <p:menuitem value="suomi" class="coloredButton"/>
                                    <p:menuitem value="english" class="coloredButton"/>
                                </p:menu>
                            </p:commandButton>
                            <p:commandButton value="Omat asetukset" type="button"
                                onclick="PF('dlg1').show();" style="width: 200px;
                                margin-top: 15px" class="coloredButton"/>
                            <p:commandButton value="Luo tapahtuma" type="button"
                                onclick="PF('dlg2').show();" style="width: 200px;
                                margin-top: 15px" class="coloredButton"/>
                            <p:commandButton value="Tee ny jotain" type="button"
                                onclick="PF('dlg3').show();" style="width: 200px;
                                margin-top: 15px" class="coloredButton"/>

                        </div>
```

```
31
32                      <p:dialog header="Omat asetukset" widgetVar="dlg1" minHeight=
                            "80" position="center" class="coloredButton">
33                         <p:inputText />
34                         <p:button icon="ui-icon-check">
35                         </p:button>
36                      </p:dialog>
37
38                      <p:dialog header="Luo observointitapahtumatapahtuma"
                            widgetVar="dlg2" minHeight="80" position="center" class="
                            coloredButton">
39                         <h:outputText value="Tähän se tulee..."/>
40                      </p:dialog>
41
42                      <p:dialog header="Tein jo jotain" widgetVar="dlg3" minHeight=
                            "80" position="center" class="coloredButton">
43                         <h:outputText value="Press anykey"/>
44                      </p:dialog>
45                   </h:form>
46                </div>
47             </nav>
48             <div>
49                <h:panelGroup styleClass="welcomeContent">
50                   <p>
51                      <h:outputText value="Moveatis asetukset."/>
52                   </p>
53                   <p>
54                      <h:outputText value="- Yleiset asetukset ja
                            tapahtumakohtaiset asetukset."/>
55                   </p>
56                </h:panelGroup>
57             </div>
58
59          </ui:define>
60       </ui:composition>
61    </h:body>
62 </html>
```

3.8 app/summary/index.xhtml

```
1  <?xml version='1.0' encoding='UTF-8' ?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
       TR/xhtml1/DTD/xhtml1-transitional.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml"
4       xmlns:h="http://xmlns.jcp.org/jsf/html"
5       xmlns:p="http://primefaces.org/ui"
6       xmlns:pe="http://primefaces.org/ui/extensions"
7       xmlns:f="http://java.sun.com/jsf/core"
8       xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
9     <h:body>
10       <ui:composition template="/WEB-INF/template.xhtml">
11
12          <ui:define name="pageTitle">#{msg.sum_title} #{summaryBean.observation.
               name}</ui:define>
13
```

```
14          <ui:define name="mainpage">
15
16              <h:outputStylesheet library="css" name="summary.css"/>
17              <h:outputStylesheet library="css" name="timeline.css"/>
18              <h:outputStylesheet library="primefaces" name="grid/grid.css"/>
19              <h:outputScript library="primefaces" name="jquery/jquery.js" />
20              <h:outputScript library="js" name="locales.js"/>
21
22              <f:event type="preRenderView" listener="#{summaryBean.
                    showObservationSavedMessage()}" />
23
24              <div id="summary-content">
25
26                  <h1>#{msg.sum_title} : #{summaryBean.observation.name}</h1>
27
28              </div>
29
30              <div id="recordings-details" class="ui-fluid">
31                  <div class="inline-bar">
32
33                      <div><h3>#{msg.sum_recordings}</h3></div>
34
35                      <div class="timeframe">
36                          <span>#{msg.sum_timeFrameStart}</span>
37                          <p:calendar id="startTime" placeholder="hh:mm:ss" size="8"
                                widgetVar="startTimeWdgt"
38                                    pattern="HH:mm:ss" timeOnly="true" mask="true"
                                        showOn="button"
39                                    locale="#{userManagedBean.locale}" showButtonPanel="
                                        true"
40                                    value="#{summaryBean.min}"
41                                    title="#{msg.sum_timeFrameTitle}"
42                                    timeZone="#{timeBean.serverTimeZone}">
43                              <f:validateRegex pattern="(^$|^__:__:__$|^(\d+:)?(\d+:)?\d
                                    +$)" />
44                              <p:clientValidator event="keyup"/>
45                          </p:calendar>
46                      </div>
47                      <div class="timeframe">
48                          <span>#{msg.sum_timeFrameEnd}</span>
49                          <p:calendar id="endTime" placeholder="hh:mm:ss" size="8"
                                widgetVar="endTimeWdgt"
50                                    pattern="HH:mm:ss" timeOnly="true" mask="true"
                                        showOn="button"
51                                    locale="#{userManagedBean.locale}" showButtonPanel="
                                        true"
52                                    value="#{summaryBean.duration}"
53                                    title="#{msg.sum_timeFrameTitle}"
54                                    timeZone="#{timeBean.serverTimeZone}">
55                              <f:validateRegex pattern="(^$|^__:__:__$|^(\d+:)?(\d+:)?\d
                                    +$)" />
56                              <p:clientValidator event="keyup"/>
57                          </p:calendar>
58                      </div>
59                  </div>
60
61                  <div class="ui-grid ui-grid-responsive-480">
62                      <div class="ui-grid-row header-row">
```

```
                <div class="ui-grid-col-5"></div>
                <div class="ui-grid-col-3">#{msg.sum_count} #{msg.
                    sum_countAbr}</div>
                <div class="ui-grid-col-3">#{msg.sum_duration}</div>
             </div>
          </div>
          <div id="records" class="ui-grid ui-grid-responsive-480"/>

          <h:form>
             <p:commandButton value="#{msg.dlg_saveAndSend}"
                         rendered="#{sessionBean.identifiedUser}"
                         styleClass="save-button"
                         onclick="PF('dlgSave').show();"
                         process="@this">
                <p:ajax update=":saveForm" resetValues="true" />
             </p:commandButton>

             <p:commandButton value="#{msg.dlg_saveDialogHeader}"
                         rendered="#{!sessionBean.identifiedUser}"
                         styleClass="save-button"
                         onclick="PF('dlgSave').show();"
                         process="@this">
                <p:ajax update=":saveForm" resetValues="true" />
             </p:commandButton>
          </h:form>
       </div>

       <pe:timeline id="timeline" widgetVar="timelineWdgt"
                 value="#{summaryBean.timeline}"
                 timeZone="#{timeBean.serverTimeZone}"
                 browserTimeZone="#{timeBean.userTimeZone}"
                 min="#{summaryBean.min}"
                 max="#{summaryBean.max}"
                 start="#{summaryBean.start}"
                 zoomMin="#{summaryBean.zoomMin}"
                 zoomMax="#{summaryBean.zoomMax}"
                 selectable="true"
                 zoomable="false"
                 moveable="true"
                 animate="false"
                 animateZoom="false"
                 axisOnTop="true"
                 groupsOrder="false"
                 groupsChangeable="false"
                 groupMinHeight="16"
                 editable="false"
                 timeChangeable="true"
                 showCurrentTime="false"
                 showMajorLabels="false"
                 stackEvents="false"
                 eventMarginAxis="0"
                 eventMargin="16">
       </pe:timeline>

       <div id="timelineControls" class="bottom">
          <div>
             <a href="#summary-content"><span> #{msg.sum_backToTop}</span><
                 span class="fa fa-chevron-circle-up"></span></a>
```

```
119                 </div>
120             </div>
121
122             <p:dialog header="#{msg.dlg_saveDialogHeader}"
123                     id="saveDialog"
124                     widgetVar="dlgSave"
125                     position="top"
126                     resizable="false"
127                     responsive="true"
128                     fitViewport="true"
129                     visible="#{facesContext.validationFailed}"
130                     styleClass="coloredDialog">
131                 <h:form id="saveForm" styleClass="dialog-form ui-fluid">
132
133                     <p:selectManyCheckbox id="basic" value="#{summaryBean.
                            selectedSaveOptions}"
134                                           rendered="#{sessionBean.identifiedUser}"
135                                           columns="1" required="true" requiredMessage=""
                                                  layout="grid">
136                         <p:ajax update="input-email" />
137                         <f:selectItem itemLabel="#{msg.dlg_saveOption}" itemValue="
                                save"
138                                       itemDisabled="#{!sessionBean.identifiedUser}"/>
139                         <f:selectItem itemLabel="#{msg.dlg_downloadOption}" itemValue
                                ="download"
140                                       itemDisabled="#{!sessionBean.loggedIn}"/>
141                         <f:selectItem itemLabel="#{msg.dlg_mailOption}" itemValue="
                                mail"
142                                       itemDisabled="#{!sessionBean.identifiedUser}"/>
143                     </p:selectManyCheckbox>
144
145                     <h:outputLabel for="input-name" value="#{msg.sum_observationName
                            }">
146                         <h:outputText rendered="#{sessionBean.identifiedUser}" value=
                                " (#{msg.dlg_defaultFileName})" />
147                     </h:outputLabel>
148                     <p:message for="input-name" />
149                     <p:inputText id="input-name" required="true" requiredMessage="#{
                            msg.dlg_notEmpty}"
150                                  value="#{summaryBean.observation.name}"
151                                  validator="#{validationBean.validateShortString}"/>
152
153                     <h:outputLabel for="input-description" value="#{msg.
                            sum_observationDescription}" />
154                     <p:message for="input-description" />
155                     <p:inputTextarea id="input-description"
156                                      value="#{summaryBean.observation.description}"
157                                      validator="#{validationBean.validateLongString}"
158                                      rows="3" cols="16" autoResize="true" />
159
160                     <h:outputLabel for="input-target" value="#{msg.
                            sum_observationTarget}" />
161                     <p:message for="input-target" />
162                     <p:inputText id="input-target"
163                                  value="#{summaryBean.observation.target}"
164                                  validator="#{validationBean.validateShortString}"/>
165
166                     <p:outputPanel rendered="#{sessionBean.identifiedUser}">
```

```
                        <h:outputLabel for="input-email" value="#{msg.dlg_giveEmail}" />
                        <p:message for="input-email"/>
                        <p:inputText id="input-email" required="true" requiredMessage="#{msg.dlg_notEmpty}"
                                value="#{summaryBean.recipientEmail}" disabled="#{!summaryBean.mailOptionChecked}">
                            <f:validator validatorId="emailValidator"/>
                        </p:inputText>
                    </p:outputPanel>

                    <p:commandButton value="#{msg.dlg_save}" icon="ui-icon-check"
                                rendered="#{sessionBean.identifiedUser}"
                                action="#{summaryBean.doSelectedSaveOperation()}"
                                onclick="window.onbeforeunload = null;"
                                ajax="false" validateClient="true"/>
                    <p:commandButton value="#{msg.dlg_save}" icon="ui-icon-check"
                                rendered="#{!sessionBean.identifiedUser}"
                                action="#{summaryBean.downloadCurrentObservation()}"
                                onclick="window.onbeforeunload = null;"
                                ajax="false" validateClient="true"/>
                    <p:commandButton value="#{msg.dlg_cancel}" icon="ui-icon-close"
                                onclick="PF('dlgSave').hide();"/>
                </h:form>
            </p:dialog>

            <p:growl id="growl" widgetVar="growlWdgt" severity="info" showDetail="true" sticky="true" escape="false" />

            <h:outputScript>
                var SummaryIndex = {
                getObservationDuration: function () {
                return #{summaryBean.observation.duration};
                },
                getMessages: function (){
                return msg = {
                sum_total: "#{msg.sum_total}",
                sum_begin: "#{msg.sum_begin}",
                sum_end: "#{msg.sum_end}",
                sum_duration: "#{msg.sum_duration}",
                sum_countAbr: "#{msg.sum_countAbr}",
                dlg_confirmLeave: "#{msg.dlg_confirmLeave}"
                };
                },
                };
            </h:outputScript>
            <h:outputScript library="js" name="summary.js"/>
        </ui:define>
    </ui:composition>
</h:body>
</html>
```

3.9 app/superuser/index.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:h="http://xmlns.jcp.org/jsf/html"
     xmlns:p="http://primefaces.org/ui"
     xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
   <body>
      <ui:composition template="/WEB-INF/template.xhtml">

         <ui:define name="pageTitle">Pääkäyttäjäsivu</ui:define>

         <ui:define name="mainpage">
            <h:outputStylesheet library="css" name="superuser.css"/>

      <div class="superuser_actions">

         <H2> Pääkäyttäjän toiminnot </H2>


         <!--Kaikkien tapahtumaryhmien, tapahtumien ja observointien listaaminen -->

         <h3> Tapahtumaryhmät ja tapahtumat</h3>

         <h:form>
            <p:growl autoUpdate="true"/>
            <p:panelMenu style="width: 80%; max-width: 400px; min-width: 250px; font-size: 1em; overflow: hidden; white-space: no-wrap">
               <p:submenu label="Omat" expanded="false" >
                  <p:submenu label="Tapahtumaryhmä 1" >
                     <p:submenu label="Tapahtuma 1" >
                        <p:menuitem value="Observointi 1" />
                     </p:submenu>
                     <p:submenu label="Tapahtuma 2">
                        <p:menuitem value="Observointi 1" />
                        <p:menuitem value="Observointi 2" />
                        <p:menuitem value="Uusi observointi..." icon="ui-icon-plusthick"/>
                     </p:submenu>
                     <p:menuitem value="Uusi tapahtuma..." icon="ui-icon-plusthick" style="color: black;"/>
                  </p:submenu>
                  <p:menuitem value="Uusi tapahtumaryhmä..." icon="ui-icon-plusthick"/>
               </p:submenu>

               <p:submenu label="Yleiset" expanded="false">
                  <p:submenu label="Tapahtumaryhmä 1" >
                     <p:submenu label="Tapahtuma 1" >
                        <p:menuitem value="Observointi 1" />
                        <p:menuitem value="Observointi 2" />
                        <p:menuitem value="Uusi observointi..." icon="ui-icon-plusthick"/>
                     </p:submenu>
                     <p:submenu label="Tapahtuma 2" >
                        <p:menuitem value="Observointi 1" />
                        <p:menuitem value="Observointi 2" />
```

```
52                      <p:menuitem value="Uusi observointi..." icon="ui-icon-
                           plusthick"/>
53                   </p:submenu>
54                   <p:menuitem value="Uusi tapahtuma..." icon="ui-icon-plusthick
                        "/>
55                </p:submenu>
56                <p:menuitem value="Uusi tapahtumaryhmä..." icon="ui-icon-
                     plusthick"/>
57             </p:submenu>
58          </p:panelMenu>
59       </h:form>
60
61       <h:form >
62          <!--Pääkäyttäjien listaaminen, poistamis- ja lisäämistoiminnot-->
63          <h3>Pääkäyttäjät</h3>
64          <p:dataTable id="superusers" var="???" value="#{superuserBean.
               getSuperusers}" style="font-size: 1em;">
65             <p:column headerText="Käyttäjä">
66                <h:outputText value="#{superuser.id}" />
67             </p:column>
68
69             <p:column headerText="Voimassa">
70                <h:outputText value="#{superuser.validUntil}" />
71             </p:column>
72
73             <p:column headerText="Poista">
74                <input type="submit" value="" name="removeSuperuserRights" />
75             </p:column>
76          </p:dataTable>
77       </h:form>
78
79
80       <h:form >
81          <h:panelGrid columns="3" style="width: 80%; max-width: 400px; min-
               width: 250px;" >
82             <p:inputText id="user" placeholder="Käyttäjätunnus" style="margin-
                  right: 0.2em; width:90%; font-size: 1em;" size="8" maxlength="8"
                   />
83             <p:inputMask id="expirationDate" placeholder="01.01.2017" value="#{
                  maskView.date}" mask="99/99/9999" size="8" style="margin-right:
                  0.2em; width:90%; font-size: 1em;" />
84             <p:commandButton id="addSuperuserButton" value="Lisää" type="button
                  " onclick="PF('addSuperuserConfirmation').show();" style=" width
                  :95%; height: 90%; font-size: 1em; float: right;"/>
85          </h:panelGrid>
86
87          <p:dialog id ="userToSuperuser" header="Lisätään pääkäyttäjäoikeudet"
               widgetVar="addSuperuserConfirmation" modal="true" width="300px">
88
89             <br />
90             <p:commandButton value="Ei" type="button" styleClass="ui-
                  confirmdialog-No" style="float: left;" />
91             <p:commandButton value="Kyllä" type="button" styleClass="ui-
                  confirmdialog-Yes" actionListener="#{superuserBean.addSuperuser}
                  " style="float: right;" />
92
93          </p:dialog>
94       </h:form>
```

```
95
96
97          <!--Julkisten kategoriaryhmien luominen-->
98
99          <h3>Julkiset kategoriaryhmät</h3>
100         <h:form >
101            <p:panelMenu style="width: 80%; max-width: 400px; min-width: 250px;
                   font-size: 1em; font-size: 1em; overflow: hidden; white-space: no-
                   wrap">
102               <p:submenu label="Julkiset kategoriaryhmät" expanded ="false">
103               <p:submenu label="Opettajan toiminnot" expanded="false">
104                  <p:menuitem value="Järjestelyt" style="color: black;"/>
105                  <p:menuitem value="Tehtävän selitys" style="color: black;"/>
106                  <p:menuitem value="Ohjaus" style="color: black;"/>
107                  <p:menuitem value="Palautteen anto" style="color: black;"/>
108                  <p:menuitem value="Tarkkailu" style="color: black;"/>
109                  <p:menuitem value="Muu toiminta" style="color: black;"/>
110               </p:submenu>
111               <p:menuitem value="Uusi kategoriaryhmä..." icon="ui-icon-plusthick"
                       style="color: black;" />
112               </p:submenu>
113            </p:panelMenu>
114         </h:form>
115         <!--Estä julkinen käyttö-->
116         <!--Julkisen ja avainkäytön aikaraja-->
117      </div>
118         </ui:define>
119      </ui:composition>
120   </body>
121</html>
```

3.10 error/index.xhtml

```
1  <?xml version='1.0' encoding='UTF-8' ?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
       TR/xhtml1/DTD/xhtml1-transitional.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml"
4       xmlns:h="http://xmlns.jcp.org/jsf/html"
5       xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
6
7     <h:body>
8        <ui:composition template="/WEB-INF/template.xhtml">
9           <ui:define name="pageTitle">#{errormessages.errorPageTitle}</ui:define>
10          <ui:define name="mainpage">
11             <h:outputText value="#{errormessages.errorMessageToUser}"/>
12             <h:outputLink value="../">
13                <h:outputText value="#{errormessages.toFrontPage}"/>
14             </h:outputLink>
15          </ui:define>
16       </ui:composition>
17    </h:body>
18 </html>
```

3.11 index.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
    xmlns:p="http://primefaces.org/ui">

   <h:body>
      <ui:composition template="/WEB-INF/template.xhtml">
         <ui:define name="mainpage">
            <nav>
               <h:form>
                  <h:panelGroup>
                     <div class="accessButtons" style="min-height: 200px;">
                        <p:commandButton value="#{msg.fp_tagUserButton}" onclick="PF('dlg1').show();" style="width: 200px; margin-top: 15px"
                                         class="coloredButton"/>
                        <p:commandButton value="#{msg.fp_identifiedUserButton}" style="width: 200px; margin-top: 15px"
                                         class="coloredButton" actionListener="#{loginBean.doIdentityLogin}" />
                        <p:commandButton value="#{msg.fp_anonUserButton}" style="width: 200px; margin-top: 15px"
                                         class="coloredButton" action="#{loginBean.doAnonymityLogin()}"/>
                     </div>
                  </h:panelGroup>
               </h:form>

               <!-- TODO: REQUIRES VALIDATION -->
               <p:dialog header="#{msg.fp_giveGroupKey}" widgetVar="dlg1" minHeight="80" position="top" resizable="false" responsive="true" modal="true" fitViewport="true" class="coloredDialog">
                  <h:form>
                     <p:inputText id="tagField" autocomplete="off" value="#{loginBean.tag}" required="true"/>
                     <p:commandButton icon="ui-icon-check" action="#{loginBean.doTagLogin}"/>
                  </h:form>
               </p:dialog>
            </nav>

            <h:panelGroup styleClass="welcomeContent" rendered="#{applicationManagedBean.installed}">
               <h:outputText escape="false" value="#{msg.fp_infoText}"/>
               <h:outputLink value="#{msg.fp_infoLink}">
                  <h:outputText value="#{msg.fp_infoLinkText}" style="text-decoration: underline"/>
               </h:outputLink>
            </h:panelGroup>
            <h:panelGroup rendered="#{!applicationManagedBean.installed}" style="margin-top: 10em;">
               <h:form>
                  <h:outputText value="#{msg.installationHelp} "/>
```

```
                    <h:commandLink value="#{msg.installationLink}" style="text-
                        decoration: underline;"
                                action="#{applicationManagedBean.doInstall}"/>
                </h:form>
            </h:panelGroup>
        </ui:define>
    </ui:composition>
  </h:body>
</html>
```

3.12 info/index.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
    TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:h="http://xmlns.jcp.org/jsf/html"
     xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
   <h:body>
      <ui:composition template="/WEB-INF/template.xhtml">

         <ui:define name="title">#{msg.applicationTitle}</ui:define>

         <ui:define name="mainpage">
            <div>
               <form>
                  <h:panelGroup styleClass="textContentArea">
                     <div>
                        <h:outputText escape="false" value="#{msg.inf_headerText}"
                            />
                        <h:outputLink value="#whatIs">
                           <h:outputText value="-> #{msg.inf_whatIs}"/>
                        </h:outputLink>
                        <br/>
                        <h:outputLink value="#userGuide">
                           <h:outputText value="-> #{msg.inf_userGuide}"/>
                        </h:outputLink>
                        <br/>
                        <h:outputLink value="#team">
                           <h:outputText value="-> #{msg.inf_credits}"/>
                        </h:outputLink>
                        <br/>
                        <h:outputLink value="#licenses">
                           <h:outputText value="-> #{msg.inf_licensedUnder}"/>
                        </h:outputLink>
                        <p><br/></p>
                     </div>
                     <div id="whatIs">
                        <h3>#{msg.inf_whatIs}</h3>
                        <h:outputText escape="false" value="#{msg.inf_whatIsText}"
                            />
                        <p><br/></p>
                     </div>

                     <div id="userGuide">
```

```
                        <h3>#{msg.inf_userGuide}</h3>
                        <h:outputText escape="false" value="#{msg.
                            inf_userGuideText}"/>
                        <p><br/></p>
                     </div>

                     <div id="team">
                        <h3>#{msg.inf_credits}</h3>
                        <h:outputText escape="false" value="#{msg.inf_creditsText}
                            "/>
                        <p><br/></p>
                     </div>

                     <div id="licenses">
                        <h3>#{msg.inf_licensedUnder}</h3>
                        <h:outputText value="#{msg.inf_licenseText}"/>
                        <br/>
                        <h:outputLink value="#{msg.inf_licenseUrl1}">
                           <h:outputText escape="false" value="#{msg.inf_licSource
                               }"/>
                        </h:outputLink>
                        <br/>
                        <h:outputLink value="#{msg.inf_licenseUrl2}">
                           <h:outputText escape="false" value="#{msg.inf_licDoc}"
                               />
                        </h:outputLink>
                     </div>


                  </h:panelGroup>
               </form>
            </div>
         </ui:define>

      </ui:composition>
   </h:body>
</html>
```

3.13 install.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
    TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:h="http://xmlns.jcp.org/jsf/html"
     xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
     xmlns:installation="http://xmlns.jcp.org/jsf/composite/installation">
   <h:body>
      <ui:composition template="/WEB-INF/template.xhtml">
         <ui:define name="mainpage">
            <h:panelGroup layout="block" rendered="#{!applicationManagedBean.
                installed}">
               <h:panelGroup layout="block">
                  <h3>
```

```
                        <h:outputText value="#{msg.installationTitle}"/>
                     </h3>
                  </h:panelGroup>
                  <h:panelGroup layout="block">
                     <h:outputText value="#{msg.installationMessage}"/>
                  </h:panelGroup>
                  <h:panelGroup layout="block">
                     <installation:identityprovider/>
                  </h:panelGroup>
               </h:panelGroup>
            </ui:define>
         </ui:composition>
      </h:body>
</html>
```

3.14 jyutesting/fail.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
    TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
   <h:head>
      <title>Ei mitään nähtävää</title>
   </h:head>
   <h:body>
      <h:panelGroup layout="block" id="testing-JYU" rendered="#{develJYULoginBean.
          isLocalhost}">
         <p>
            Entiteettiä ei löytynyt, rekisteröidy.
         </p>
         <p>
            Vaihtoehtoisesti jokin muu meni vikaan, katso logeja.
         </p>
      </h:panelGroup>
   </h:body>
</html>
```

3.15 jyutesting/index.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
    TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:p="http://primefaces.org/ui">
   <h:head>
      <title>Ei mitään nähtävää</title>
   </h:head>
   <h:body>
      <h:panelGroup layout="block" id="testing-JYU" rendered="#{develJYULoginBean.
          isLocalhost}">
         <p>
```

```
12              Näyttää, että sivua käytetään localhostista, jossa JYU-
                    tunnistautuminen ei ole käytössä,
13              joten tämä sivua tarjoaa mockUp -tyylisen vaihtoehdon kehitystyötä
                    varten.
14          </p>
15          <p>
16              Syötä tunnukset käsin, siinä muodossa kuin ne tulisivat JYU-
                    tunnistuspalvelulta.
17              <h:form>
18                  <p>
19                      <p:inputText placeholder="sammarju@jyu.fi" value="#{
                            develJYULoginBean.username}"/>
20                  </p>
21                  <p>
22                      <p:inputText placeholder="Sami Kallio" value="#{
                            develJYULoginBean.givenName}"/>
23                  </p>
24                  <p>
25                      <p:inputText placeholder="affiliate;student" value="#{
                            develJYULoginBean.affiliation}"/>
26                  </p>
27                  <p>
28                      <h:panelGroup layout="block" id="develOhje">
29                          Valitse Kirjaudu, jos testaat normaalia käyttöä. Jos testaat
30                          sovelluksen rekisteröintiä, valitse joko tavallinen Rekisteröi
                                dy
31                          tai pääkäyttäjän Rekisteröidy.
32                      </h:panelGroup>
33                      <h:panelGroup>
34                          <p:commandButton value="Kirjaudu" action="#{develJYULoginBean
                                .doLogin}"/>
35                      </h:panelGroup>
36                      <h:panelGroup>
37                          TAI
38                      </h:panelGroup>
39                      <h:panelGroup>
40                          <p:commandButton value="Rekisteröidy (tavallinen)" action="#{
                                develJYULoginBean.doRegistration}"/>
41                      </h:panelGroup>
42                      <h:panelGroup>
43                          TAI
44                      </h:panelGroup>
45                      <h:panelGroup>
46                          <p:commandButton value="Rekisteröidy (pääkäyttäjä)" action="
                                #{develJYULoginBean.doSuperUserRegistration}"/>
47                      </h:panelGroup>
48                  </p>
49              </h:form>
50          </p>
51      </h:panelGroup>
52      </h:body>
53  </html>
```

3.16 resources/installation/identityprovider.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:cc="http://xmlns.jcp.org/jsf/composite"
     xmlns:h="http://xmlns.jcp.org/jsf/html">

   <!-- INTERFACE -->
   <cc:interface>
   </cc:interface>

   <!-- IMPLEMENTATION -->
   <cc:implementation>
      <h:panelGroup layout="block" style="width: 25em; margin: 5em auto 0 auto;">
         <h:form style="margin: 0 auto 0 auto; align-content: center;">
            <h:commandButton value="#{identitymessages.register}" style="width: 11 em; height: 2em;
                             margin: 1.5em auto 0 auto;"
                             class="coloredButton"
                             actionListener="#{identityProviderRegistrationBean.registerSuperUser}"/>
         </h:form>
      </h:panelGroup>

   </cc:implementation>
</html>
```

Chapter 4

Style sheets

4.1 resources/css/base.css

```
1
2  /*
3  The base stylesheet for Moveatis-webapplication
4  */
5  /*
6     Created on : 24.2.2016, 11:55:11
7     Author : Jarmo Juujärvi (based on base.css by Sami kallio)
8  */
9  @import url(https://fonts.googleapis.com/css?family=Roboto);
10
11 html {
12    width: 100%;
13    height: 100%;
14    margin: 0;
15    padding: 0;
16 }
17
18 body {
19    width: 100%;
20    height: 100%;
21    margin: 0;
22    padding: 0;
23    font-size: 100%;
24    color: #ffffff;
25    background-color: #131D24;
26    background-image: repeating-linear-gradient(
27       45deg,
28       #080b0d,
29       #080b0d 20px,
30       #0a0f13 20px,
31       #0a0f13 40px);
32    background-size: 100%;
33    background-attachment: fixed;
34    font-family: 'Roboto', sans-serif;
35 }
36
37 /* Use same font family on widgets for consistency
38    (excluding .fa elements) */
39 .ui-widget:not(.fa) {
40    font-family: 'Roboto', sans-serif;
```

```
}

/* Sticky footer based on this:
  http://ryanfait.com/resources/footer-stick-to-bottom-of-page/ */
.wrapper {
   min-height: 100%;
   height: auto !important;
   height: 100%;
   margin: 0 auto -30px;
}
.push {
   height: 30px;
}
#Footer {
   height: 30px;
   font-size: 16px;
   background-color: #ff6600;
   clear: both;
}


nav {
   float: left;
   width: 24%;
   height: 40%;
   border: none;
   margin-right: 0.5em;
   padding-bottom: 1.5em;
   padding-right: 0.5em;
}

.welcomeContent {
   float: right;
   width: 70%;
   display: block;
   padding: 0.5em;
}

.textContentArea {
   width: 70%;
   margin: 1% auto 1% auto;
   display: block;
   padding: 2.5em;
}

.naviHeader {
   height: 28px;
   background: #ff6600;
   background: #ff6600 linear-gradient(top, rgba(255,255,255,0.8), rgba
       (255,255,255,0));
   background: #ff6600 -webkit-gradient(linear, left top, left bottom, from(rgba
       (255,255,255,0.8)), to(rgba(255,255,255,0)));
   background: #ff6600 -moz-linear-gradient(top, rgba(255,255,255,0.8), rgba
       (255,255,255,0));
   background-size: 100%;
   text-align: center;
   border: 0;
   border-radius: 0;
```

```
96      color: white!important;
97  }
98
99  #naviHead .ui-icon.fa {
100     border-radius: 0;
101     width: auto;
102     height: auto;
103     padding: 0 0.8em;
104     font-size: 120%;
105 }
106
107 .naviHeader > ul {
108     float: right;
109 }
110
111 .ui-menu .ui-menuitem-link {
112     line-height: 22px!important;
113 }
114 .ui-menubar > .ui-menuitem-link {
115     width: auto;
116     padding: 0.3em 0.9em;
117     line-height: 22px!important;
118 }
119
120 .ui-menu .ui-menu-parent .ui-icon-triangle-1-s {
121     display: none;
122 }
123
124 .ui-menu .ui-menu-parent .ui-icon-triangle-1-e {
125     display: none;
126 }
127
128 ul.ui-menu-child {
129     white-space: nowrap;
130     width: 260px!important;
131 }
132
133 .naviButtons {
134     display:table-cell;
135     text-align: center;
136     float: right;
137 }
138
139 .accessButtons {
140
141     width: 9.5em;
142     height: 2.0em;
143     margin: auto auto auto auto;
144     /* display: block;*/
145     text-align: center;
146     padding: 1.0em;
147 }
148
149 .coloredButton {
150     background: #ff6600;
151     color: white!important;
152     font-size: 100%!important;
153     text-shadow: none;
```

```
154 }
155
156 /* Dialog style */
157 .coloredDialog {
158    background: #19232A!important;
159    color: white;
160    border: none;
161 }
162
163 .coloredDialog .ui-dialog-titlebar {
164    background:#ff6600;
165    background:#ff6600 linear-gradient(top, rgba(255,255,255,0.8), rgba
          (255,255,255,0));
166    background:#ff6600 -webkit-gradient(linear, left top, left bottom, from(rgba
          (255,255,255,0.8)), to(rgba(255,255,255,0)));
167    background:#ff6600 -moz-linear-gradient(top, rgba(255,255,255,0.8), rgba
          (255,255,255,0));
168    color: white;
169    text-shadow: none;
170    font-weight: normal;
171 }
172
173 .coloredDialog form label {
174    color: white;
175 }
176
177 .coloredDialog .ui-fluid .ui-button {
178    width: auto;
179 }
180
181 .coloredDialog form input,
182 .coloredDialog form textarea,
183 .coloredDialog form table {
184    margin-bottom: 0.8em;
185 }
186 .coloredDialog form button {
187    margin-bottom: 0.4em;
188 }
189
190 .ui-dialog-content {
191    overflow-y: auto;
192    max-height: calc(100vh - 65px);
193    max-width: 24em;
194 }
195
196 .ui-dialog-content .ui-state-error {
197    border-width: 2px;
198 }
199
200 /* Naviheader styles menu */
201 .naviHeader.ui-menu, .naviHeader.ui-menu .ui-menu-child{
202    background:#ff6600;
203    background:#ff6600 linear-gradient(top, rgba(255,255,255,0.8), rgba
          (255,255,255,0));
204    background:#ff6600 -webkit-gradient(linear, left top, left bottom, from(rgba
          (255,255,255,0.8)), to(rgba(255,255,255,0)));
205    background:#ff6600 -moz-linear-gradient(top, rgba(255,255,255,0.8), rgba
          (255,255,255,0));
```

```
206     color: white!important;
207 }
208 
209 .naviHeader .ui-state-hover, .naviHeader .ui-widget-content .ui-state-hover, .
        naviHeader .ui-widget-header .ui-state-hover, .naviHeader .ui-state-focus, .
        naviHeader .ui-widget-content .ui-state-focus, .naviHeader .ui-widget-header .
        ui-state-focus{
210     border:1px solid #ff6600;
211     background:#ff6600;
212     background:#ff6600 linear-gradient(top, rgba(255,255,255,0.6), rgba
            (255,255,255,0));
213     background:#ff6600 -webkit-gradient(linear, left top, left bottom, from(rgba
            (255,255,255,0.6)), to(rgba(255,255,255,0)));
214     background:#ff6600 -moz-linear-gradient(top, rgba(255,255,255,0.6), rgba
            (255,255,255,0));
215     font-weight:normal;
216     color:black;
217     -moz-box-shadow:none;
218     -webkit-box-shadow:none;
219     box-shadow:none;
220 }
221 
222 .naviHeader .ui-widget-content {
223     border: none;
224 }
225 
226 #naviHead .naviHeader.ui-menu-parent > .ui-menu-list .ui-menuitem-link {
227     width: 95%;
228 }
229 
230 #naviHead .ui-menuitem,
231 #naviHead .ui-menuitem-link {
232     margin: 0;
233 }
234 
235 .naviHeader > .ui-menu-list {
236     margin-top: -0.2em;
237 }
238 
239 .naviFooter {
240     width: 100%;
241     float: right;
242     height: 20px;
243     background-size: 50%;
244     background-color: #ff6600;
245     align-content: flex-end;
246     text-align: center;
247     color: white !important;
248 
249 }
250 
251 .facetLeft {
252     position: absolute;
253     left: 0;
254     top: 0;
255     padding-left: 0.5em;
256     padding-top: 0.1em;
257     font-size: 140%;
```

```
}

.fa-user {
   float: right!important;
}

a:link {text-decoration: none; color: white;}
a:visited {text-decoration: none; color: white;}
a:active {text-decoration: none; color: white;}
a:hover {text-decoration: none; color: white;}

.ui-dialog-footer {
   background: transparent none repeat scroll 0% 0%;
   border: medium none;
   float: right;
}

.ui-confirmdialog-no {
   float: right;
}

.ui-growl .ui-growl-icon-close {
   display: block !important;
   background-color: white;
   opacity: 0.7;
   background-position: -96px -128px;
   -webkit-box-shadow: 2px 2px 1px -1px rgba(0,0,0,0.6);
   -moz-box-shadow: 2px 2px 1px -1px rgba(0,0,0,0.6);
   box-shadow: 2px 2px 1px -1px rgba(0,0,0,0.6);
   border-radius: 5px;
   border: 5px solid;
}

.ui-growl .ui-growl-icon-close:hover {
   opacity: 1;
}

/* Error icon change */
.ui-message-error-icon {
   background: none!important;
}
.ui-message-error-icon:before {
   content: "\f12a";
   font-family: FontAwesome;
   margin-left: .2em;
}

@media screen and (max-width: 800px) {

   nav {
      padding-top: 1.0em;
      width: 96%;
      margin: 1% auto 1% auto;
      border: none;
      padding-bottom: 0em;

   }

```

```
316     .welcomeContent {
317        width: 96%;
318        margin: 1% auto 1% auto;
319        padding: 0em;
320     }
321
322     .textContentArea {
323        width: 70%;
324        margin: auto auto auto auto;
325        display: block;
326        padding: 2.5em;
327     }
328
329     .naviHeader {
330        height: 28px;
331        background-color: #ff6600;
332        background-size: 100%;
333        text-align: center;
334        color: white !important;
335     }
336
337     .naviButtons {
338        width: 50px;
339        height: 10px;
340        display:table-cell;
341        text-align: center;
342        float: right;
343     }
344
345     .accessButtons {
346        width: 9.5em;
347        height: 2.0em;
348        /* margin: 2.5% auto 2.5% auto;*/
349        display:table;
350        text-align: center;
351        padding: 0.2em;
352        color: white;
353     }
354
355     .coloredButton {
356        background: #ff6600;
357     }
358
359     .naviFooter {
360        width: 100%;
361        float: right;
362        height: 20px;
363        background-size: 50%;
364        background-color: #ff6600;
365        text-align: center;
366        color: white !important;
367     }
368
369 }
```

4.2 resources/css/categoryselection.css

```
1
2  /*
3     Created on : Mar 31, 2016, 2:38:10 PM
4     Author : Ilari Paananen
5  */
6
7  #mainPage {
8     text-align: center;
9     margin-bottom: 1em;
10    clear: both;
11 }
12
13 #container {
14    width: 320px;
15    display: inline-block;
16    text-align: left;
17 }
18
19 .category-set-list {
20    margin-bottom: 0.5em;
21 }
22
23 .category-set-list-last {
24    margin-bottom: 2em;
25 }
26
27 .category-set-list-label {
28    display: inline-block;
29    width: 220px;
30    vertical-align: middle;
31 }
32
33 .category-set-list-select {
34    width: 170px;
35    vertical-align: middle;
36    margin-right: 1em;
37 }
38
39 .category-set-list-input{
40    width: 197px;
41    vertical-align: middle;
42    margin-right: 1em;
43 }
44
45 .category-set {
46    width: 280px;
47    display: inline-block;
48    vertical-align: top;
49    margin: 0 10px 1em 0;
50    background: rgba(85, 102, 119, 0.4);
51    border-radius: 10px;
52    padding: 10px;
53 }
54
55 .category-set h2 {
56    margin-top: 0;
```

```
57     white-space: nowrap;
58     overflow: hidden;
59     text-overflow: ellipsis;
60     font-size: 1.3em;
61 }
62
63 .category-set-remove-button {
64     float: right;
65     font-size: 150%;
66 }
67
68 .no-categories-text {
69     margin-bottom: 0.5em;
70 }
71
72 .category {
73     margin-bottom: 0.5em;
74     white-space: nowrap;
75 }
76
77 .ui-chkbox-box {
78     vertical-align: middle;
79     margin-right: 0.5em;
80 }
81
82 .category .ui-chkbox-label {
83     vertical-align: middle;
84     font-size: 1.2em;
85     display: inline;
86 }
87
88 .category-type-button {
89     text-align: center;
90     width: 4.5em;
91     margin-right: 0.5em;
92 }
93
94 .category-type-button .ui-button-text {
95     padding-left: 0;
96     padding-right: 0;
97 }
98
99 .category-text {
100    width: calc(100% - 7.5em - 13px);
101    margin-right: 0.5em;
102 }
103
104 .category-remove-button {
105    width: 2em;
106 }
107
108 .category .ui-message-error {
109    background-color: transparent;
110    border: 0;
111 }
112
113 .new-category-button {
114    width: 100%;
```

```
115     text-align: left;
116 }
117 
118 #continue-div {
119     margin-top: 1em;
120     text-align: right;
121 }
122 
123 .helpText {
124     font-size: 1.15em;
125 }
126 
127 @media screen and (min-width: 640px) {
128     #container {
129         width: 640px;
130     }
131 }
132 
133 @media screen and (min-width: 960px) {
134     #container {
135         width: 960px;
136     }
137 }
```

4.3 resources/css/control.css

```
1 
2 /*
3     Created on : Mar 15, 2016, 2:16:45 PM
4     Author : Juha Moisio
5 */
6 
7 #mainPage {
8     max-width: 960px;
9     margin: 0 auto;
10    margin-bottom: 3em;
11 }
12 
13 #mainPage > form,
14 #mainPage .other-observations-fieldset {
15    max-width: 50rem;
16    margin: 1em 0;
17 }
18 
19 #mainPage a.edit-key {
20    color: gold;
21    text-shadow: 1px 1px 1px rgb(51, 51, 51);
22 }
23 
24 .key-dialog {
25    max-width: 12em;
26 }
27 
28 a.new-key {
29    color: grey;
```

```
}

.column-group-keys,
.row-editor,
.row-toggler,
.ui-widget.fa-trash {
   font-size: 1.6em;
}

.row-toggler {
   font-family: FontAwesome;
   width: 1em;
}

.row-editor,
.column-remove,
.column-group-keys {
   width: 1em;
   text-align: center;
}

.ui-datatable thead th {
   text-align: left;
   font-size: 1em;
}

.ui-datatable .ui-row-editor span {
   float: none;
}

.column-name,
.new-eventgroup-form .input-name {
   width: 12em;
}

.column-name input,
.column-name textarea {
   width: 100%;
}

.column-name textarea {
   font-size: 0.8em;
}

.new-eventgroup-form {
   display: inline;
}

.new-eventgroup-form .input-name {
   margin-right: .5em;
}

.column-group-keys .ui-icon,
.row-editor .ui-icon,
.fa.ui-icon {
   background: none;
   text-indent: 0;
   width: auto;
```

```
    height: auto;
    border-radius: 0;
}

.column-remove > .ui-commandlink {
    display: none;
}

.ui-row-editing .column-remove > .ui-commandlink {
    display: block;
}

.row-editor .ui-icon-pencil:before {
    content:"\f040";
}
.row-editor .ui-icon-close:before {
    content:"\f00d";
}
.row-editor .ui-icon-check:before {
    content:"\f046";
}

.row-editor .ui-icon-check:before{
    margin-right: .1em;
    display: inline-block;
    vertical-align: middle;
}

.row-editor .ui-icon-close:before {
    display: inline-block;
    vertical-align: middle;
    margin-top: -4px;
}

.row-editor .ui-icon-check:hover {
    color: green;
}

.row-editor .ui-icon-close:hover,
.ui-widget.fa-trash:hover {
    color:red;
}

.list-categorysets {
    padding: 0;
    margin: 0;
    display: inline-block;
}

.list-categorysets > li {
    display: inline-block;
    padding: .2em;
    background: #2D363F;
    border-radius: 5px;
    margin: .2em;
    border: 1px solid rgba(79, 79, 79, .4);
    pointer-events: auto;
    font-size: .9em;
```

```
146     box-shadow: 1px 1px 1px grey;
147 }
148 
149 .list-categorysets > li:hover {
150     box-shadow: none;
151 }
152 
153 .link-new-categoryset > span {
154     font-family: "Roboto",sans-serif;
155 }
156 
157 .link-categoryset {
158     color: white!important;
159     text-shadow: 1px 1px 1px rgb(51, 51, 51);
160 }
161 
162 .link-categoryset:hover {
163     text-shadow: 1px 1px 1px grey;
164 }
165 
166 .ui-sortable-helper {
167     z-index: 2000!important;
168     font-size: 0.9em!important;
169     opacity:0.4;
170 }
171 
172 .row-toggler-mobile {
173     display: none;
174 }
175 
176 .ui-growl {
177     z-index: 2000!important;
178 }
179 
180 .categories-table {
181     margin-top: .5em;
182 }
183 
184 #categorySetForm .ui-column-title {
185     font-size: .8em;
186     color: #777;
187 }
188 
189 #categorySetForm .ui-state-highlight .ui-column-title {
190     color: white;
191 }
192 
193 #categorySetForm .ui-row-editor > .ui-icon-close {
194     display:none!important;
195 }
196 
197 .ui-datatable-selectable:not(.ui-row-editing) .ui-cell-editor {
198     pointer-events: none;
199 }
200 .ui-cell-editor-input {
201     pointer-events: auto;
202 }
203 
```

```
#eventGroupsForm .ui-datatable table {
   table-layout: auto;
}

#commandButtons > button:not(:last-of-type) {
   font-size: 100%;
}

.categories-table .ui-cell-editor-input > input {
   display: inline-block;
   max-width: 90%;
   margin: 0.3em 1%;
}

#observationDetail .ui-panelgrid-cell.label {
   width: 7em;
}

#observationDetail .ui-panelgrid-cell.value {
   width: auto;
}

#observationDetail .ui-inplace-content .ui-inputfield {
   display: block;
}

#observationFormCommands {
   font-size: .9em;
}

.ui-datalist a.ui-commandlink:link,
.ui-datalist a.ui-commandlink:visited,
.ui-datalist a.ui-commandlink:active,
.ui-datalist a.ui-commandlink:hover {
   color: #4f4f4f;
}

.ui-selectonemenu,
.ui-inputfield {
   max-width: 100%;
   box-sizing: border-box;
}

.ui-selectonemenu {
   padding-right: 0;
}

.ui-datatable-empty-message {
   color: white;
   border-color: transparent;
   background: #131D24;
}

.row-editor .ui-icon-close {
   display: none!important;
}

.ui-selectonemenu {
```

```
262     width: 100%;
263 }
264
265 .column-categorytype {
266     width: 8em;
267 }
268
269 .other-observations-fieldset {
270     padding: 0;
271 }
272
273 .ui-fieldset-legend {
274     font-size: .8em;
275 }
276
277 #eventGroupsForm .ui-datalist-header,
278 #otherObservationsForm thead {
279     font-size: 0.9em;
280 }
281
282 #startObservationForm {
283     text-align: right;
284 }
285
286 #observationDetail {
287     margin-bottom: .6em;
288 }
289
290 #observationDetail .ui-panelgrid-content {
291     background: none;
292     border: none;
293     color: white;
294 }
295
296 #observationDetail .ui-outputpanel.editable {
297     padding: 0.2em 0.5em;
298     border: 1px inset #444;
299     border-radius: 3px;
300 }
301
302 #observationDetail .ui-inplace-display.ui-state-highlight {
303     background: none;
304     text-decoration: underline;
305 }
306
307 .link-observation {
308     text-decoration: underline!important;
309 }
310
311 .link-observation:hover {
312     color: #ff6600!important;
313 }
314
315 @media screen and (max-width: 800px) {
316
317     #mainPage {
318         max-width: 40em;
319     }
```

```
320
321     #mainPage > form {
322        margin-left: auto;
323        margin-right: auto;
324     }
325
326 }
327
328 @media screen and (max-width: 640px) {
329
330     .eventGroupsTable thead {
331        display:none!important;
332     }
333
334     .row-toggler-mobile {
335        display: block;
336        font-size: 0.9em;
337     }
338
339     .row-toggler-mobile span:before {
340        font-family: FontAwesome;
341        margin-right: .3em;
342     }
343
344     .row-toggler-mobile span:first-child:before {
345        content:"\f138";
346     }
347
348     .row-toggler-mobile span:last-child:before {
349        content:"\f13a";
350     }
351
352     .column-name {
353        margin-top: .6em;
354        font-size: 1.3em;
355     }
356
357     #eventGroupsForm .row-editor,
358     #eventGroupsForm .column-remove,
359     #eventGroupsForm .column-group-keys {
360        display: inline-block;
361        float: none;
362        width: auto;
363     }
364
365     #eventGroupsForm .ui-column-title,
366     #eventGroupsForm .row-toggler {
367        display: none;
368     }
369
370     #categorySetForm .ui-cell-editing .ui-column-title,
371     #categorySetForm .ui-cell-editing .ui-cell-editor-input {
372        margin-left: .4em;
373     }
374
375 }
376
377 @media screen and (max-width: 480px) {
```

```
    #commandButtons button {
        width: 100%;
        text-align: left;
    }
}

@media screen and (max-width: 370px) {
    .categorySetCommandButtons > button {
        width: 100%;
        text-align: left;
    }
}
```

4.4 resources/css/observer.css

```

/*
    Created on : Feb 17, 2016, 10:29:25 AM
    Author : Ilari Paananen
*/

body {
    font-size: 0; /* NOTE: Removes extra space between elements. */
}

#naviHead, #Footer {
    font-size: 16px;
}

#mainPage {
    width: 100%;
    margin: 0;
    padding: 0;
    text-align: center;
    clear: both;
}

#growl_container {
    font-size: 1.1rem;
}

#content {
    max-width: 100%;
    margin: 0;
    padding: 0;
    display: inline-block;
}

#total-time {
    margin: 0;
    padding: 0;
    cursor: default;
}

#controls {
```

```
41    white-space: nowrap;
42 }
43
44 .control-button {
45    display: inline-block;
46    width: 5em;
47    margin: 0 0.2em;
48    padding: 0.2em 0;
49    background-color: #567;
50    background: linear-gradient(0deg, #345, #567);
51    cursor: pointer;
52 }
53
54 .no-text-select {
55    /* Disable text selection.
56       Source: http://stackoverflow.com/questions/826782/css-rule-to-disable-text-
             selection-highlighting */
57    -webkit-touch-callout: none; /* iOS Safari */
58    -webkit-user-select: none; /* Chrome/Safari/Opera */
59    -khtml-user-select: none; /* Konqueror */
60    -moz-user-select: none; /* Firefox */
61    -ms-user-select: none; /* IE/Edge */
62    user-select: none; /* non-prefixed version, currently
63                          not supported by any browser */
64 }
65
66 #category-list {
67    max-width: 100%;
68    margin: 0;
69    padding: 0;
70 }
71
72 .category-set {
73    max-width: 100%;
74    list-style-type: none;
75    background: rgba(85, 102, 119, 0.4);
76 }
77
78 .category-item {
79    max-width: 100%;
80    padding: 0;
81    position: relative;
82    text-align: left;
83    white-space: nowrap;
84    background-color: #567;
85    background: linear-gradient(0deg, #345, #567);
86    cursor: pointer;
87 }
88
89 .category-item:last-child {
90    margin: 0;
91 }
92
93 .category-item.down {
94    background: #f82;
95 }
96
97 .category-name {
```

```
98     max-width: 90%;
99     margin: 0 0.3em;
100    padding: 0;
101    display: inline-block;
102    vertical-align: middle;
103    position: relative;
104    top: 15%;
105    overflow: hidden;
106    text-overflow: ellipsis;
107 }
108
109 .category-value {
110    margin: 0;
111    margin-left: 0.3em;
112    padding: 0 0.15em;
113    float: right;
114    position: relative;
115    z-index: 1;
116    background-color: #345;
117 }
118
119 .disabled {
120    color: #345;
121    cursor: default;
122 }
123
124 .category-item.disabled > .category-value {
125    color: #567;
126 }
127
128 /*
129
130    TODO: Use browser specific calc?
131    http://stackoverflow.com/a/14101451
132
133 */
134
135 #total-time {
136    height: calc((100vh - 70px) * 0.12);
137    font-size: 9vh;
138 }
139
140 #controls {
141    height: calc((100vh - 70px) * 0.15);
142 }
143
144 .control-button {
145    border-radius: 1vh;
146    font-size: 6vh;
147 }
148
149 .category-set {
150    margin: 0 0 1.5vh 0;
151    padding: 1.5vh;
152    border-radius: 1vh;
153 }
154
155 .category-item {
```

```
156     height: calc((100vh - 70px) * 0.73 * 0.127);
157     margin: 0;
158     margin-bottom: 1vh;
159     border-radius: 1vh;
160 }
161 
162 .category-name {
163     font-size: 5vh;
164 }
165 
166 .category-value {
167     font-size: 4vh;
168     border-radius: 1vh;
169 }
170 
171 @media screen and (min-height: 480px) {
172 
173     #total-time {
174         height: calc((480px - 70px) * 0.12);
175         font-size: calc(480px * 0.09);
176     }
177 
178     #controls {
179         height: calc((480px - 70px) * 0.15);
180     }
181 
182     .control-button {
183         border-radius: calc(480px * 0.01);
184         font-size: calc(480px * 0.06);
185     }
186 
187     .category-set {
188         margin: 0 0 calc(480px * 0.015) 0;
189         padding: calc(480px * 0.015);
190         border-radius: calc(480px * 0.01);
191     }
192 
193     .category-item {
194         height: calc((480px - 70px) * 0.73 * 0.127);
195         margin: 0;
196         margin-bottom: calc(480px * 0.01);
197         border-radius: calc(480px * 0.01);
198     }
199 
200     .category-name {
201         font-size: calc(480px * 0.05);
202     }
203 
204     .category-value {
205         font-size: calc(480px * 0.04);
206         border-radius: calc(480px * 0.01);
207     }
208 
209 }
210 
211 @media screen and (max-height: 240px) {
212 
213     #total-time {
```

```
214       height: calc((240px - 70px) * 0.12);
215       font-size: calc(240px * 0.09);
216    }
217
218    #controls {
219       height: calc((240px - 70px) * 0.15);
220    }
221
222    .control-button {
223       border-radius: calc(240px * 0.01);
224       font-size: calc(240px * 0.06);
225    }
226
227    .category-set {
228       margin: 0 0 calc(240px * 0.015) 0;
229       padding: calc(240px * 0.015);
230       border-radius: calc(240px * 0.01);
231    }
232
233    .category-item {
234       height: calc((240px - 70px) * 0.73 * 0.127);
235       margin: 0;
236       margin-bottom: calc(240px * 0.01);
237       border-radius: calc(240px * 0.01);
238    }
239
240    .category-name {
241       font-size: calc(240px * 0.05);
242    }
243
244    .category-value {
245       font-size: calc(240px * 0.04);
246       border-radius: calc(240px * 0.01);
247    }
248
249 }
```

4.5 resources/css/settings.css

```
1
2  /*
3     Created on : Mar 8, 2016, 5:33:35 PM
4     Author : jaanjuuj
5  */
6
7  @import url(https://fonts.googleapis.com/css?family=Roboto);
8
9  html {
10    width: 100%;
11    height: 100%;
12 }
13
14 body {
15    width: 100%;
16    margin: auto auto auto auto;
```

```
17    font-size: 100%;
18    color: #ffffff;
19    background-color: #131D24;
20    background-image: repeating-linear-gradient(
21       45deg,
22       #080b0d,
23       #080b0d 20px,
24       #0a0f13 20px,
25       #0a0f13 40px);
26    background-size: 100%;
27    background-attachment: fixed;
28    font-family: 'Roboto', sans-serif;
29 }
30
31 nav {
32    float: left;
33    width: 24%;
34    border: none;
35    display: table;
36    margin: auto auto auto auto;
37 }
38
39 .accessButtons {
40
41    width: 9.5em;
42    height: 2.0em;
43    margin: auto auto auto auto;
44
45    text-align: center;
46    padding: 1.0em;
47
48 }
49
50    .coloredButton {
51       background: #ff6600;
52       opacity: 0.9;
53    }
54
55 a:link {text-decoration: none; color: whitesmoke;}
56 a:visited {text-decoration: none; color: whitesmoke;}
57 a:active {text-decoration: none; color: whitesmoke;}
58 a:hover {text-decoration: none; color: orange;}
59
60
61 @media screen and (max-width: 800px) {
62
63    nav {
64       padding-top: 0.5em;
65       width: 96%;
66       margin: 1% auto 1% auto;
67       border: none;
68       padding-bottom: 0.5em;
69
70    }
71
72    .welcomeContent {
73       width: 90%;
74       margin: 1% auto 1% auto;
```

```
75        padding: 1.0em;
76     }
77
78     .accessButtons {
79        width: 9.5em;
80        height: 2.0em;
81  /* margin: 2.5% auto 2.5% auto;*/
82        display:table;
83        text-align: center;
84        padding: 1.0em;
85     }
86
87     .coloredButton {
88        background: #ff6600;
89        opacity: 0.9;
90     }
91  }
```

4.6 resources/css/summary.css

```
1
2   /*
3      Summary page style.
4      Created on : Feb 24, 2016, 2:10:56 PM
5      Author : Juha Moisio <juha.pa.moisio at student.jyu.fi>
6   */
7
8   #mainPage {
9      max-width: 960px;
10     margin: 0 auto;
11  }
12
13  #summary-content {
14     padding: .1em .5em;
15     position: relative;
16     z-index: 2;
17  }
18
19  #recordings-details {
20     font-size: 1.1em;
21     padding: .1em .5em;
22     background-color: #19232A;
23     border-radius: 10px 10px 0 0;
24     position: relative;
25     z-index: 2;
26  }
27
28  #recordings-details .ui-grid-col-5 {
29     display: table;
30  }
31
32  #recordings-details .categoryNumber,
33  #recordings-details .categoryLabel {
34     display: table-cell;
35  }
```

```
36
37 #recordings-details .categoryNumber,
38 #recordings-details .header-row,
39 #recordings-details .summary-row {
40    font-family: Arial,sans-serif;
41 }
42
43 .header-row {
44    margin-bottom: -1.3em;
45    font-size: 1em;
46 }
47
48 #recordings-details .categoryNumber {
49    padding-right: 10px;
50    width: 1em;
51 }
52
53 .content,
54 .ui-grid-row {
55    max-width: 38em;
56 }
57 .inline-bar > div {
58    display: inline-block;
59    vertical-align: middle;
60 }
61 .total {
62    font-size: 2em;
63 }
64
65 .timeframe > span:first-child {
66    margin-right: .3em;
67    font-size: 1.1em;
68    font-weight: bold;
69 }
70
71 .timeframe .ui-state-error {
72    border-width: 3px;
73 }
74
75 .ui-fluid .timeframe .ui-calendar,
76 .ui-fluid .timeframe .ui-calendar input {
77    width: auto;
78 }
79
80 .align-bar > div {
81    text-align: center;
82 }
83
84 .ui-grid {
85    position: relative;
86 }
87 .ui-grid-row {
88    padding: 0.4em 0;
89    margin-left: 3px;
90 }
91 .ui-grid-row:not(.header-row):not(:last-child):after {
92    content:"";
93    border-bottom: 1px solid #4F4F4F;
```

```
94     width: 100%;
95     position: absolute;
96     left: 0;
97     margin-top: -0.4em;
98     opacity: 0.5;
99  }
100
101 .ui-grid-row > div:not(:first-child) {
102    text-align: right;
103 }
104
105 .summary-row > div:not(:first-child) {
106    white-space: pre;
107 }
108
109 .ui-grid-col-3 {
110    display: inline-block;
111    min-width: 10em;
112 }
113
114 #timelineControls {
115    position: fixed;
116    bottom: 0.6em;
117    z-index: 1;
118    font-size: .76em;
119    -webkit-transition: bottom 1s;
120    transition: bottom .5s;
121    text-align: right;
122    width: 100%;
123    max-width: 960px;
124 }
125 #Footer {
126    z-index: 2;
127    position: relative;
128 }
129 #timelineControls.bottom {
130    /*bottom: 3em;*/
131 }
132 #timelineControls > div {
133    background: rgba(42, 29, 29, 0.2) none repeat scroll 0% 0%;
134    display: inline-block;
135    padding: 0.3em;
136    border-radius: 0.3em;
137    margin-right: 0.6em;
138 }
139 #timelineControls a:hover {
140    color: #fa4;
141 }
142 #timelineControls .fa {
143    cursor: pointer;
144    font-size: 1.6em;
145    margin-left: .1em;
146 }
147
148 .percent {
149    white-space: pre;
150 }
151
```

```
.ui-growl .categoryNumber {
   display: none!important;
}

.ui-button > .fa {
   border-radius: 0;
}

.categorySet {
   display: none!important;
}

#records .gapBefore {
   margin-top: 1em;
   border-top: 2px solid #567;
}

#select-save {
   margin-top: .1em;
   margin-bottom: .6em;
}

.login-panel {
   margin: 2em;
}

#startTime_input,
#endTime_input {
   text-align: center;
}

.ui-selectonebutton.fa span {
   font-family: FontAwesome;
   font-weight: normal;
}

#saveDialog {
   max-width: 19em;
}

#saveDialog .ui-selectonebutton .ui-button-text {
   padding: .1em;
}

.ui-timepicker-div .ui-slider {
   margin-left:1em;
}

.ui-fluid .save-button {
   width: auto;
   margin: .6em 0;
}

.ui-datepicker-current {
   display: none;
}

.ui-datepicker-trigger {
```

```
    font-size: 0;
}

.ui-datepicker-trigger:before {
    content: "\f017";
    font-size: 1rem;
    font-family: FontAwesome;
}

/* Responsive */
@media (max-width:700px){
    .mobileCentered {
        text-align: center;
    }
    .inline-bar .timeframe {
        display: block;
    }
    .timeframe > span:first-child {
        min-width: 5em;
        display: inline-block;
    }
}
#saveDialog .ui-selectonebutton .ui-buttonset.ui-buttonset-3 .ui-button {
    width: 33%;
}
@media (max-width:480px){
    .ui-grid.ui-grid-responsive-480 .ui-grid-row {
        display:block;
    }
    .ui-grid.ui-grid-responsive-480 .ui-grid-row > div {
        width:100%;
        width: calc(100% - 2em);
        float:none;
    }
    .ui-grid.ui-grid-responsive-480 .ui-grid-row > .ui-grid-col-5 {
        float: left;
        width: auto;
    }
    .ui-grid.ui-grid-responsive-480 .ui-grid-row:not(.header-row):after {
        margin-top: .3em;
    }
    #recordings-details .ui-grid-col-5 {
        margin-bottom: .3em;
    }
    .ui-grid-row > div:not(:first-child) {
        text-align: right;
        margin-left: 2em;
    }
}
```

4.7 resources/css/superuser.css

```

/*
    Created on : Mar 17, 2016, 1:49:24 PM
```

```
   Author : kavakorh
*/

.superuser_actions{
   padding-top: 0.5em;
   width: 80%;
   margin: 1% auto 1% auto;
   padding-bottom: 0.5em;
   font-size: 1em;
}

#superusers{
   width: 80%;
   max-width: 400px;
   min-width: 250px;
}
```

4.8 resources/css/timeline.css

```

/*
   Created on : Mar 2, 2016, 11:38:42 AM
   Author : Juha Moisio <juha.pa.moisio at student.jyu.fi>
*/

/* Custom styles for the Timeline component. */

#timeline {
   clear: both;
   margin-bottom: 3em;
   margin-right: 2px;
   margin-left: 2px;
}

#timeline .timeline-event {
   color: #AEAEAE;
   background: #F82;
   box-shadow: none;
   margin-top: 0px;
}

#timeline .timeline-event-content {
   margin: 0;
   padding: 10px;
   text-shadow: none;
}

#timeline .timeline-axis,
#timeline .timeline-groups-axis {
   background: transparent linear-gradient(0deg, #345, #567) repeat scroll 0% 0%;
   opacity: 1;
}

#timeline .timeline-axis-text {
   font-size: 0.9em;
```

```
    color: white;
}

#timeline .timeline-axis-grid {
    border-left-style: dashed!important;
    border-top-style: dashed!important;
}

#timeline .timeline-groups-axis > .timeline-groups-text {
    color: white;
}

#timeline .timeline-event-selected .timeline-event-content {
    cursor: default;
}

#timeline .timeline-event-selected {
    background-color: #cedff0;
}

#timeline .categoryLabel {
    display: none!important;
}

#timeline .timeline-frame {
    border-radius: 0;
    border: none;
}

#timeline .ui-widget-content:not(.timeline-axis) {
    color: #000;
}

#timeline .timeline-axis {
    color: #4F4F4F;
}

#timeline .timeline-groups-text,
#timeline .timeline-axis {
    font-family: Arial,sans-serif;
}
.dummyRecord {
    display: none!important;
}
```