LightGuard (owner)

Revisions

- 111ab9 LightGuard March 24, 2011
- 6f16f0 LightGuard March 24, 2011

Private Gist

All pages are served over SSL and all pushing and pulling is done over SSH. No one may fork, clone, or view it unless they are given this private URL.

Every gist with this icon (🔒) is private.

Public Gist

Anyone may fork, clone, or view it.

Every repository with this icon (☺) is public.

Description:	EntityConverter for Seam 3
Public Clone URL:	git://gist.github.com/885926.git
Embed All Files:	show embed

EntityConverter.java #
embed raw

```
1  package org.jboss.seam.faces.conversion;
2
3  import java.io.Serializable;
4  import java.lang.reflect.Field;
5  import java.lang.reflect.InvocationTargetException;
6  import java.lang.reflect.Member;
7  import java.lang.reflect.Method;
8  import java.util.Collections;
9  import java.util.Map;
10 import java.util.concurrent.ConcurrentHashMap;
11
12 import javax.el.ValueExpression;
13 import javax.enterprise.inject.Any;
14 import javax.faces.component.PartialStateHolder;
15 import javax.faces.component.UIComponent;
16 import javax.faces.context.FacesContext;
17 import javax.faces.convert.*;
18 import javax.inject.Inject;
19 import javax.persistence.EntityManager;
20 import javax.persistence.metamodel.EntityType;
21
22 import org.jboss.seam.solder.properties.Properties;
23 import org.jboss.seam.solder.properties.Property;
24
25 /**
26  * Converter for JPA entities. Uses the id of the entity for the key.
27  * @author <a href="http://community.jboss.org/people/LightGuard">Jason Porter</a>
28  */
29 @FacesConverter("org.jboss.seam.EntityCOnverter")
30 public class EntityConverter implements javax.faces.convert.Converter, PartialStateHolder {
31
32     private static final Object[] ZERO_ARGS = new Object[] {};
33
34     @Inject @Any
35     private EntityManager entityManager;
36
37     private Map<String, Serializable> stateMap = new ConcurrentHashMap<String, Serializable>();
38     private boolean initialStateMarked;
39     private boolean isTransient;
40
41     @Override
42     public Object getAsObject(FacesContext context, UIComponent component, String value) {
```

```
43          final ValueExpression ve = component.getValueExpression("value");
44          final Class<?> entityType = ve.getExpectedType();
45          return this.entityManager.find(entityType, this.stateMap.get(value));
46      }
47
48      @Override
49      public String getAsString(FacesContext context, UIComponent component, Object value) {
50          final Serializable id = this.findId(value);
51          final String idKey = value.getClass().getSimpleName() + id.hashCode();
52          if (!this.stateMap.containsKey(idKey)) {
53              this.stateMap.put(value.getClass().getSimpleName() + id.hashCode(), id);
54          }
55
56          return idKey;
57      }
58
59      private Serializable findId(Object entity) {
60          final EntityType<?> entityType = this.entityManager.getMetamodel().entity(entity.getClass());
61
62          final Member idAttribute = entityType.getId(entity.getClass()).getJavaMember();
63
64          Property p = Properties.createProperty(idAttribute);
65
66          return (Serializable) p.getValue(entity);
67      }
68
69      // ---------- State saving --------------
70      @Override
71      public void markInitialState() {
72          this.initialStateMarked = true;
73      }
74
75      @Override
76      public boolean initialStateMarked() {
77          return this.initialStateMarked;
78      }
79
80      @Override
81      public void clearInitialState() {
82          this.stateMap = new ConcurrentHashMap<String, Serializable>();
83          this.initialStateMarked = false;
84      }
85
86      @Override
87      public Object saveState(FacesContext context) {
88          if (context == null)
89          {
90              throw new IllegalArgumentException("FacesContext must not be null");
91          }
92
93          return Collections.unmodifiableMap(this.stateMap);
94      }
95
96      @Override
97      public void restoreState(FacesContext context, Object state) {
98          if (state != null)
99          {
100             this.stateMap = new ConcurrentHashMap<String, Serializable>((Map<? extends String,? extends Serializable>) state);
101             return;
102         }
103
104         this.stateMap.clear();
105     }
106
107     @Override
108     public boolean isTransient() {
109         return this.isTransient;
110     }
111
112     @Override
113     public void setTransient(boolean newTransientValue) {
114         this.isTransient = newTransientValue;
115     }
116 }
117
```