Student/list.xhtml

Display error messages

pagination

```
<?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:ui="http://xmlns.jcp.org/jsf/facelets"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:f="http://xmlns.jcp.org/jsf/core">

  <ui:composition template="/template.xhtml">
    <ui:define name="title">
      <h:outputText value="#{bundle.ListStudentTitle}"></h:outputText>
    </ui:define>

    <ui:define name="body">
      <h:form styleClass="jsfcrud_list_form">
        <h:panelGroup id="messagePanel" layout="block">
          <h:messages errorStyle="color: red" infoStyle="color: green" layout="table"/>
        </h:panelGroup>

        <h:outputText escape="false" value="#{bundle.ListStudentEmpty}" rendered="#{studentController.items.rowCount == 0}"/>

<h:panelGroup rendered="#{studentController.items.rowCount > 0}">
          <h:outputText value="#{studentController.pagination.pageFirstItem + 1}..#{studentController.pagination.pageLastItem +
1}/#{studentController.pagination.itemsCount}"/>&nbsp;
          <h:commandLink action="#{studentController.previous}" value="#{bundle.Previous}
#{studentController.pagination.pageSize}" rendered="#{studentController.pagination.hasPreviousPage}"/>&nbsp;
          <h:commandLink action="#{studentController.next}" value="#{bundle.Next} #{studentController.pagination.pageSize}"
rendered="#{studentController.pagination.hasNextPage}"/>&nbsp;

          <h:dataTable value="#{studentController.items}" var="item" border="0" cellpadding="2" cellspacing="0"
rowClasses="jsfcrud_odd_row,jsfcrud_even_row" rules="all" style="border:solid 1px">
            <h:column>
              <f:facet name="header">
                <h:outputText value="#{bundle.ListStudentTitle_id}"/>
              </f:facet>
              <h:outputText value="#{item.id}"/>
            </h:column>
            <h:column>
              <f:facet name="header">
                <h:outputText value="#{bundle.ListStudentTitle_course}"/>
              </f:facet>
              <h:outputText value="#{item.course}"/>
            </h:column>
            <h:column>
              <f:facet name="header">
                <h:outputText value="#{bundle.ListStudentTitle_firstName}"/>
              </f:facet>
              <h:outputText value="#{item.firstName}"/>
            </h:column>
            <h:column>
              <f:facet name="header">
                <h:outputText value="#{bundle.ListStudentTitle_lastName}"/>
              </f:facet>
              <h:outputText value="#{item.lastName}"/>
            </h:column>
```

Student/list.xhtml

```
            <h:column>
              <f:facet name="header">
                <h:outputText value="#{bundle.ListStudentTitle_age}"/>
              </f:facet>
              <h:outputText value="#{item.age}"/>
            </h:column>
            <h:column>
              <f:facet name="header">
                <h:outputText value="#{bundle.ListStudentTitle_email}"/>
              </f:facet>
              <h:outputText value="#{item.email}"/>
            </h:column>
            <h:column>
              <f:facet name="header">
                <h:outputText value="&nbsp;"/>
              </f:facet>
              <h:commandLink action="#{studentController.prepareView}" value="#{bundle.ListStudentViewLink}"/>
              <h:outputText value=" "/>
              <h:commandLink action="#{studentController.prepareEdit}" value="#{bundle.ListStudentEditLink}"/>
              <h:outputText value=" "/>
              <h:commandLink action="#{studentController.destroy}" value="#{bundle.ListStudentDestroyLink}"/>
            </h:column>
          </h:dataTable>
        </h:panelGroup>
        <br />
        <h:commandLink action="#{studentController.prepareCreate}" value="#{bundle.ListStudentCreateLink}"/>
        <br />
        <br />
        <h:link outcome="/index" value="#{bundle.ListStudentIndexLink}"/>
      </h:form>
    </ui:define>
  </ui:composition>

</html>
```

Student Controller

```
package jsf;

import entity.Student;
import jsf.util.JsfUtil;
import jsf.util.PaginationHelper;
import session.StudentFacade;

import java.io.Serializable;
import java.util.ResourceBundle;
import javax.ejb.EJB;
import javax.inject.Named;
import javax.enterprise.context.SessionScoped;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.convert.Converter;
import javax.faces.convert.FacesConverter;
import javax.faces.model.DataModel;
import javax.faces.model.ListDataModel;
import javax.faces.model.SelectItem;

@Named("studentController")
@SessionScoped
public class StudentController implements Serializable {

  private Student current;
  private DataModel items = null;
  @EJB
  private session.StudentFacade ejbFacade;
  private PaginationHelper pagination;
  private int selectedItemIndex;

  public StudentController() {}

  public Student getSelected() {
    if (current == null) {
      current = new Student();
      selectedItemIndex = -1;        }
    return current;    }
```

Return current student

```
  private StudentFacade getFacade() {
    return ejbFacade;        }
```

Return student session bean

```
  public PaginationHelper getPagination() {
    if (pagination == null) {
      pagination = new PaginationHelper(10) {
        @Override
        public int getItemsCount() {
          return getFacade().count();                    }

        @Override
        public DataModel createPageDataModel() {
          return new ListDataModel(getFacade().findRange(new int[]{getPageFirstItem(), getPageFirstItem() + getPageSize()}));
        }               };                    }
    return pagination;       }
```

Override createPageDataModel to use findRange query to read first 10 items.

Student Controller

Methods called by the 5 xhtml pages

```
public String prepareList() {
   recreateModel();                                   return "List";              }

 public String prepareView() {
   current = (Student) getItems().getRowData();
   selectedItemIndex = pagination.getPageFirstItem() + getItems().getRowIndex();
   return "View";                  }

 public String prepareCreate() {
   current = new Student();
   selectedItemIndex = -1;
   return "Create";          }

 public String create() {
   try {
      getFacade().create(current);
      JsfUtil.addSuccessMessage(ResourceBundle.getBundle("/Bundle").getString("StudentCreated"));
      return prepareCreate();
   } catch (Exception e) {
      JsfUtil.addErrorMessage(e, ResourceBundle.getBundle("/Bundle").getString("PersistenceErrorOccured"));
      return null;               }       }

 public String prepareEdit() {
   current = (Student) getItems().getRowData();
   selectedItemIndex = pagination.getPageFirstItem() + getItems().getRowIndex();
   return "Edit";          }

 public String update() {
   try {
      getFacade().edit(current);
      JsfUtil.addSuccessMessage(ResourceBundle.getBundle("/Bundle").getString("StudentUpdated"));
      return "View";
   } catch (Exception e) {
      JsfUtil.addErrorMessage(e, ResourceBundle.getBundle("/Bundle").getString("PersistenceErrorOccured"));
      return null;
   }
 }

 public String destroy() {
   current = (Student) getItems().getRowData();
   selectedItemIndex = pagination.getPageFirstItem() + getItems().getRowIndex();
   performDestroy();
   recreatePagination();
   recreateModel();
   return "List";
 }

 public String destroyAndView() {
   performDestroy();
   recreateModel();
   updateCurrentItem();
   if (selectedItemIndex >= 0) {
      return "View";
   } else {
```

Student Controller

```
        // all items were removed - go back to list
        recreateModel();
        return "List";
    }
}

private void performDestroy() {
    try {
        getFacade().remove(current);
        JsfUtil.addSuccessMessage(ResourceBundle.getBundle("/Bundle").getString("StudentDeleted"));
    } catch (Exception e) {
        JsfUtil.addErrorMessage(e, ResourceBundle.getBundle("/Bundle").getString("PersistenceErrorOccured"));
    }
}

private void updateCurrentItem() {
    int count = getFacade().count();
    if (selectedItemIndex >= count) {
        // selected index cannot be bigger than number of items:
        selectedItemIndex = count - 1;
        // go to previous page if last page disappeared:
        if (pagination.getPageFirstItem() >= count) {
            pagination.previousPage();
        }
    }
    if (selectedItemIndex >= 0) {
        current = getFacade().findRange(new int[]{selectedItemIndex, selectedItemIndex + 1}).get(0);
    }
}

public DataModel getItems() {
    if (items == null) {
        items = getPagination().createPageDataModel();     }
    return items;          }

private void recreateModel() {      items = null;          }   %clears items

private void recreatePagination() {          pagination = null;      }

public String next() {
    getPagination().nextPage();
    recreateModel();
    return "List";
}

public String previous() {
    getPagination().previousPage();
    recreateModel();
    return "List";
}

public SelectItem[] getItemsAvailableSelectMany() {
    return JsfUtil.getSelectItems(ejbFacade.findAll(), false);
}
```

See override for createDataModel in constructor
for Pagination Helper above

Student Controller

```
  public SelectItem[] getItemsAvailableSelectOne() {
    return JsfUtil.getSelectItems(ejbFacade.findAll(), true);
  }

  public Student getStudent(java.lang.String id) {
    return ejbFacade.find(id);
  }

  @FacesConverter(forClass = Student.class)
  public static class StudentControllerConverter implements Converter {

    @Override
    public Object getAsObject(FacesContext facesContext, UIComponent component, String value) {
      if (value == null || value.length() == 0) {
        return null;
      }
      StudentController controller = (StudentController) facesContext.getApplication().getELResolver().
          getValue(facesContext.getELContext(), null, "studentController");
      return controller.getStudent(getKey(value));
    }

    java.lang.String getKey(String value) {
      java.lang.String key;
      key = value;
      return key;
    }

    String getStringKey(java.lang.String value) {
      StringBuilder sb = new StringBuilder();
      sb.append(value);
      return sb.toString();
    }

    @Override
    public String getAsString(FacesContext facesContext, UIComponent component, Object object) {
      if (object == null) {
        return null;
      }
      if (object instanceof Student) {
        Student o = (Student) object;
        return getStringKey(o.getId());
      } else {
        throw new IllegalArgumentException("object " + object + " is of type " + object.getClass().getName() + "; expected type: " +
Student.class.getName());
      }
    }
  }
}
```

Makes this JSF facelets converter class, converting the current selected item to a Student object.

PaginationHelper

```
package jsf.util;

import javax.faces.model.DataModel;

public abstract class PaginationHelper {

   private int pageSize;
   private int page;

   public PaginationHelper(int pageSize) {
      this.pageSize = pageSize;
   }

   public abstract int getItemsCount();

   public abstract DataModel createPageDataModel();

   public int getPageFirstItem() {
      return page * pageSize;  }

   public int getPageLastItem() {
      int i = getPageFirstItem() + pageSize - 1;
      int count = getItemsCount() - 1;
      if (i > count) {
         i = count;
      }
      if (i < 0) {
         i = 0;
      }
      return i;   }

   public boolean isHasNextPage() {
      return (page + 1) * pageSize + 1 <= getItemsCount();
   }

   public void nextPage() {
      if (isHasNextPage()) {
         page++;
      }   }

   public boolean isHasPreviousPage() {
      return page > 0;
   }

   public void previousPage() {
      if (isHasPreviousPage()) {
         page--;
      }
   }

   public int getPageSize() {
      return pageSize;
   }
}
```

Studentfacade (session bean)

```
package session;

import entity.Student;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

@Stateless
public class StudentFacade extends AbstractFacade<Student> {
  @PersistenceContext(unitName = "JPAappPU")
  private EntityManager em;

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

  public StudentFacade() {            super(Student.class);          }               }
```

Abstract facade (implementing generic session bean methods)

```
package session;

import java.util.List;
import javax.persistence.EntityManager;

public abstract class AbstractFacade<T> {
  private Class<T> entityClass;

  public AbstractFacade(Class<T> entityClass) {
    this.entityClass = entityClass;   }

  protected abstract EntityManager getEntityManager();

  public void create(T entity) {   getEntityManager().persist(entity);        }
public void edit(T entity) {      getEntityManager().merge(entity);     }
public void remove(T entity) {         getEntityManager().remove(getEntityManager().merge(entity));   }
public T find(Object id) {      return getEntityManager().find(entityClass, id);          }

  public List<T> findAll() {
    javax.persistence.criteria.CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();
    cq.select(cq.from(entityClass));
    return getEntityManager().createQuery(cq).getResultList();       }

  public List<T> findRange(int[] range) {
    javax.persistence.criteria.CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();
    cq.select(cq.from(entityClass));
    javax.persistence.Query q = getEntityManager().createQuery(cq);
    q.setMaxResults(range[1] - range[0]);
    q.setFirstResult(range[0]);
    return q.getResultList();
  }

  public int count() {
    javax.persistence.criteria.CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();
    javax.persistence.criteria.Root<T> rt = cq.from(entityClass);
    cq.select(getEntityManager().getCriteriaBuilder().count(rt));
    javax.persistence.Query q = getEntityManager().createQuery(cq);
    return ((Long) q.getSingleResult()).intValue();          }      }
```