```java
package fi.paatti.mobile.views.listview;


import com.vaadin.data.Item;
import com.vaadin.data.util.sqlcontainer.SQLContainer;
import fi.paatti.mobile.mobileview.MobileViewLogic;
import fi.paatti.mobile.paattiapplication.MobileViewHandler;
import fi.paatti.mobile.views.mobilecontentview.MobileContentView;


/**
 * A base class for other list views.
 * <p/>
 * @author Lauri Satokangas, lauri.n.satokangas@student.jyu.fi @date 16.3.2012
 * @since 0.0.1
 */
public abstract class ListView extends MobileContentView {

  private final MobileViewHandler viewHandler;
  private String IDColumnName = null;
  private String headerTextColumnName = null;
  private String text1ColumnName = null;
  private String text2ColumnName = null;
  private ListViewLogic logic;

  /**
   * Constructor.
   * <p/>
   * @param viewHandler The applications view handler.
   */
  public ListView(MobileViewHandler viewHandler) {
    this.viewHandler = viewHandler;
    createUi();
  }

  /**
   * Create the user interface.
   */
  private void createUi() {
    this.setSpacing(true);

  }

  @Override
  public void setContentViewLogic(MobileViewLogic logic) {
    this.logic = (ListViewLogic) logic;
```

```
  }

  /**
   * Create the list items from database.
   * <p/>
   * @param tableName The table where the items will be created.
   * @param idColumn  The id column of the table.
   */
  public void createListItemsFromDb(String tableName, String idColumn) {
    createList(tableName, idColumn, false);
  }

  /**
   * Create a list from database. This list is just for showing the items.
   * The items cannot be activated, opened or anything else.
   * <p/>
   * @param tableName The table where the items will be created.
   * @param idColumn  The id column of the table.
   */
  public void createReadOnlyListItemsFromDb(String tableName, String idColumn) {
    createList(tableName, idColumn, true);
  }

  /**
   * Create the list items from database.
   * <p/>
   * @param tableName    Name of the table in the database.
   * @param idColumnName Name of the id-column in the table.
   * @param readOnly     Are the list items read only.
   */
  private void createList(String tableName, String tableIdColumnName, boolean readOnly) {
    // Remove old components before adding new.
    this.removeAllComponents();

    SQLContainer listItemsContainer = viewHandler.getDbService().getSQLContainerFromDBTable(tableName, tableIdColumnName,
        true);

    Object itemId = listItemsContainer.firstItemId();
    Item item;

    // Go through all the items in the container and create listitems from those.
    while (itemId != null) {
      ListItem listItem = null;

      item = listItemsContainer.getItem(itemId);
      if (IDColumnName != null) {
```

```
        if (text2ColumnName != null && text1ColumnName != null && headerTextColumnName != null) {
          listItem = new ListItem(item, IDColumnName, headerTextColumnName, text1ColumnName, text2ColumnName);
        } else if (text1ColumnName != null && headerTextColumnName != null) {
          listItem = new ListItem(item, IDColumnName, headerTextColumnName, text1ColumnName);
        } else if (headerTextColumnName != null) {
          listItem = new ListItem(item, IDColumnName, headerTextColumnName);

        }
        if (!readOnly) {
          listItem.setItemClickLogic(logic);
        } else {
          listItem.setOpenButtonEnabled(false);
        }
        addComponent(listItem);

      }
      itemId = listItemsContainer.nextItemId(itemId);
    }
  }

  /**
   * Get the applications view handler.
   * <p/>
   * @return The view handler of the application.
   */
  public MobileViewHandler getViewHandler() {
    return viewHandler;
  }

  /**
   * Set the name of the property/columnname that will be used when creating a list item.
   * <p/>
   * @param headerTextColumnName The name of the property/columnname containing the value for items header text.
   */
  protected void setHeaderTextColumnName(String headerTextColumnName) {
    this.headerTextColumnName = headerTextColumnName;
  }

  /**
   * Get the name of the property/columnname of the items ID.
   * <p/>
   * @return the idColumnName The property/columnname of the items ID.
   */
  protected String getIDColumnName() {
    return IDColumnName;
  }
```

```
  /**
   * Get the name of the property/columnname that will be used as item header when creating a list item.
   * <p/>
   * @return the headerTextColumnName The name of the property/columnname containing the value for items header text.
   */
  protected String getHeaderTextColumnName() {
    return headerTextColumnName;
  }

  /**
   * Get the name of the property/columnname that will be used as the first text trow when creating a list item.
   * <p/>
   * @return the tex1ColumnName The name of the property/columnname containing the value for items first text row.
   */
  protected String getText1ColumnName() {
    return text1ColumnName;
  }

  /**
   * Get the name of the property/columnname that will be used as the second text trow when creating a list item.
   * <p/>
   * @return the tex2ColumnName The name of the property/columnname containing the value for items second text row.
   */
  protected String getText2ColumnName() {
    return text2ColumnName;
  }

  /**
   * Set the ID column name.
   * <p/>
   * @param idColumnName the idColumnName to set
   */
  protected void setIdColumnName(String idColumnName) {
    this.IDColumnName = idColumnName;
  }

  /**
   * Set the first text row column name.
   * <p/>
   * @param text1ColumnName the tex1ColumnName to set
   */
  protected void setText1ColumnName(String text1ColumnName) {
    this.text1ColumnName = text1ColumnName;
  }
```

```
  /**
   * Set the second text row column name.
   * <p/>
   * @param text2ColumnName the text2ColumnName to set
   */
  protected void setText2ColumnName(String text2ColumnName) {
    this.text2ColumnName = text2ColumnName;
  }
}
```