```
package fi.paatti.research.paattiapplication.views.eventview.tools;


import com.vaadin.ui.Button.ClickEvent;
import com.vaadin.ui.*;
import fi.paatti.research.paattiapplication.views.eventview.EventTool;
import fi.paatti.research.paattiapplication.views.eventview.tools.NodeFactory.MultiPathNode;
import fi.paatti.research.paattiapplication.views.eventview.tools.NodeFactory.MultiPathNode.NodePath;
import fi.paatti.research.paattiapplication.views.eventview.tools.NodeFactory.RadioNode;
import fi.paatti.research.paattiapplication.views.eventview.tools.NodeFactory.TreeNode;
import java.io.Serializable;


/**
 * SettingsView is the part of the UI that displays the selected TreeNode's
 * information, allowing the user to modify the node's content through the
 * UI elements of the layout.
 *
 * @author Tapio Keränen, t.tapio.keranen@student.jyu.fi
 */
public class SettingsView extends VerticalLayout implements Serializable {
  private static final long serialVersionUID = 1L;

  private EventTool eventTool;
  private SettingsViewListener settingsViewListener;
  private Label idLabel;
  private VerticalLayout parentsList, childrenList;
  private TreeNode activeNode;

  private TextField descriptionField;
  private RichTextArea contentArea;
  private ReturnValueButton addPathButton;
  private VerticalLayout pathLayout;

  private static final String ADD_PATH_BUTTON_CAPTION = "Lisää polku";
  private static final String REMOVE_PATH_BUTTON_CAPTION = "Poista";
  private static final String REMOVE_CHILD_BUTTON_CAPTION = "Poista lapsi";
  private static final String DESCRIPTION_FIELD_CAPTION = "Otsikko:";
  private static final String CONTENT_AREA_CAPTION = "Sisältö:";
  private static final String NODE_ID_LABEL_CAPTION = "Tunniste: ";
  private static final String NODE_PARENTS_LABEL_CAPTION = "Vanhemmat:";
  private static final String NODE_CHILDREN_LABEL_CAPTION = "Lapset:";

  /**
   * SettingsView constructor.
   *
```

```
   * @param eventTool the event tool.
   */
  public SettingsView(EventTool eventTool) {
    this.eventTool = eventTool;

    settingsViewListener = new SettingsViewListener();

    idLabel = new Label();

    descriptionField = new TextField(DESCRIPTION_FIELD_CAPTION);
    descriptionField.setWidth("100%");
    descriptionField.setEnabled(false);

    contentArea = new RichTextArea(CONTENT_AREA_CAPTION);
    contentArea.setSizeFull();
    contentArea.setVisible(false);

    pathLayout = new VerticalLayout();
    pathLayout.setSizeFull();

    addPathButton = new ReturnValueButton(ADD_PATH_BUTTON_CAPTION, null);
    addPathButton.addListener(settingsViewListener);
    addPathButton.setSizeFull();
    addPathButton.setVisible(false);
    addPathButton.setEnabled(true);

    parentsList = new VerticalLayout();
    parentsList.setSpacing(true);
    childrenList = new VerticalLayout();
    childrenList.setSpacing(true);

    HorizontalLayout idRow = new HorizontalLayout();

    idRow.addComponent(new Label(NODE_ID_LABEL_CAPTION));
    idRow.addComponent(idLabel);
    idRow.setSpacing(true);

    addComponent(new Label(NODE_PARENTS_LABEL_CAPTION));
    addComponent(parentsList);
    addComponent(new Label(NODE_CHILDREN_LABEL_CAPTION));
    addComponent(childrenList);
    addComponent(descriptionField);
    addComponent(contentArea);
    addComponent(addPathButton);
    addComponent(pathLayout);
    setSpacing(true);
```

```
    setMargin(true);
  }

  /**
   * Sets the contents of the setting's view, using the type of the given node
   * (<code>instanceof</code>) to determine the type (and amount) of relevant
   * UI elements needed to modify the node's contents.
   *
   * @param node the TreeNode.
   */
  public void setViewContent(TreeNode node) {
    if (node == null) {
      clear();
      return;
    }

    idLabel.setValue(node.toString());

    parentsList.removeAllComponents();
    for (TreeNode parent : node.getParents()) {
      parentsList.addComponent(new Label(NODE_ID_LABEL_CAPTION + parent));
    }

    childrenList.removeAllComponents();
    for (TreeNode child : node.getChildren()) {
      childrenList.addComponent(createRow(child));
    }

    descriptionField.setPropertyDataSource(node.getProperty("description"));

    contentArea.setPropertyDataSource(node.getProperty("content"));
    contentArea.setEnabled(true);

    pathLayout.removeAllComponents();

    if (node instanceof RadioNode) {
      RadioNode radio = (RadioNode) node;
      Object[] nodePaths = radio.getNodePaths().toArray();

      NodePath nodePath;

      for (int i = 0; i < nodePaths.length; i++) {
        nodePath = (NodePath) nodePaths[i];

        ReturnValueButton btn = new ReturnValueButton(REMOVE_PATH_BUTTON_CAPTION, nodePath);
```

```
        btn.addListener(settingsViewListener);
        btn.setSizeFull();

        Select select = new Select("Polku");

        for (TreeNode child : node.getChildren()) {
          select.addItem(child);
        }
        select.setPropertyDataSource(nodePath.getProperty("destinationNode"));
        select.setWidth("40px");

        TextField value = new TextField("Arvo", nodePath.getProperty("value"));

        value.setWidth("40px");

        TextField description = new TextField("Teksti", nodePath.getProperty("description"));

        description.setWidth("100%");

        HorizontalLayout pathRow = new HorizontalLayout();

        pathRow.addComponent(btn);
        pathRow.addComponent(select);
        pathRow.addComponent(value);
        pathRow.addComponent(description);
        pathRow.setExpandRatio(btn, 0);
        pathRow.setExpandRatio(select, 0);
        pathRow.setExpandRatio(value, 0);
        pathRow.setExpandRatio(description, 1f);
        pathRow.setSpacing(true);

        pathLayout.addComponent(pathRow);
      }
      addPathButton.setVisible(true);
      contentArea.setVisible(false);
    } else {
      addPathButton.setVisible(false);
      contentArea.setVisible(true);
    }

    descriptionField.setEnabled(true);

    if (activeNode != null) {
      activeNode.setActive(false);
    }
    activeNode = node;
```

```
    activeNode.setActive(true);
  }

  /**
   * Clears the contents of the settings view.
   */
  private void clear() {
    idLabel.setValue("");
    parentsList.removeAllComponents();
    childrenList.removeAllComponents();
    pathLayout.removeAllComponents();
    contentArea.setVisible(false);
    addPathButton.setVisible(false);
    descriptionField.setEnabled(false);

    if (activeNode != null) {
      activeNode.setActive(false);
      activeNode = null;
    }
  }

  /**
   * Creates a horizontal (layout) row that contains the child node's id and a
   * button that unlinks the child from its parent.
   *
   * @param node the node to unlink.
   * @return created row.
   */
  private HorizontalLayout createRow(TreeNode node) {
    HorizontalLayout row = new HorizontalLayout();

    ReturnValueButton unlinkButton = new ReturnValueButton(REMOVE_CHILD_BUTTON_CAPTION, node);

    unlinkButton.addListener(settingsViewListener);

    row.addComponent(new Label(NODE_ID_LABEL_CAPTION + node));
    row.addComponent(unlinkButton);
    return row;
  }

  /**
   * SettingsViewListener is the SettingsView's button click event listener.
   */
  private class SettingsViewListener implements Button.ClickListener {

    /**
```

```
     * Handles the view's button click events.
     * <p>
     * The handled events are:
     * <ul>
     * <li>remove the selected child from the node,</li>
     * <li>add a new path to the node, and</li>
     * <li>remove the selected path from the node.</li>
     * </ul>
     *
     * @param event click event.
     */
    public void buttonClick(ClickEvent event) {
      ReturnValueButton button = (ReturnValueButton) event.getButton();
      String buttonCaption = button.getCaption();
      TreeNode node;

      if (buttonCaption.equals(REMOVE_CHILD_BUTTON_CAPTION)) {
        node = (TreeNode) button.getReturnValue();
        activeNode.removeChild(node);
      } else if (buttonCaption.equals(ADD_PATH_BUTTON_CAPTION)) {
        MultiPathNode multiPathNode = (MultiPathNode) activeNode;

        multiPathNode.createNodePath();
      } else if (buttonCaption.equals(REMOVE_PATH_BUTTON_CAPTION)) {
        MultiPathNode multiPathNode = (MultiPathNode) activeNode;

        multiPathNode.removeNodePath((NodePath) button.getReturnValue());
      }
      setViewContent(activeNode);
      eventTool.repaint();
    }
  }
}
```