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


import com.vaadin.data.Item;
import com.vaadin.data.Property;
import com.vaadin.data.Property.ValueChangeEvent;
import com.vaadin.data.util.IndexedContainer;
import com.vaadin.data.util.sqlcontainer.SQLContainer;
import com.vaadin.terminal.ExternalResource;
import com.vaadin.ui.Alignment;
import com.vaadin.ui.Label;
import com.vaadin.ui.OptionGroup;
import com.vaadin.ui.Video;
import fi.paatti.mobile.applicationinfo.CssStyleNames;
import fi.paatti.mobile.applicationinfo.Resources;
import fi.paatti.mobile.mobileview.MobileView;
import fi.paatti.mobile.mobileview.MobileViewLogic;
import fi.paatti.mobile.paattiapplication.MobileViewHandler;
import fi.paatti.mobile.views.mobilecontentview.MobileContentView;
import fi.paatti.paattidatabaseutils.names.PaattiColumnNames;
import java.util.ArrayList;


/**
 * A view for an event.
 * <p/>
 * @author Jari Salokangas, jari.p.t.salokangas@student.jyu.fi
 * @author Lauri Satoakangas, lauri.n.satokangas@student.jyu.fi
 */
public class EventView extends MobileContentView {

  private static final long serialVersionUID = 1L;
  private final MobileViewHandler viewhandler;

  /**
   * The start of the event.
   */
  public static final int EVENT_START = 0;

  /**
   * The middle of the event.
   */
  public static final int EVENT_MIDDLE = 1;

  /**
   * The end of the event.
```

```
   */
  public static final int EVENT_END = 2;
  // TODO: Task types should be taken from database TASKTYPE-table.
  private final int TASKTYPE_TEXT = 1;
  private final int TASKTYPE_RADIO = 2;
  private final int TASKTYPE_SENSOR = 3;
  private final int TASKTYPE_AUDIO = 4;
  private final int TASKTYPE_VIDEO = 5;
  private final int TASKTYPE_SLIDER = 6;
  //
  private OptionGroup choiceOptionGroup;
  private Label descriptionLabel;
  private Label contentLabel;
  private EventViewLogic eventViewLogic;
  private int selectedChoiceID;
  private int currentTaskID;
  private ArrayList<TaskChoiceChangeEventListener> listeners = new ArrayList<TaskChoiceChangeEventListener>();

  /**
   * Contructor for event view.
   * <p/>
   * @param viewHandler
   */
  public EventView(MobileViewHandler viewHandler) {
    this.viewhandler = viewHandler;
    this.setStyleName("eventview");
    createUi();
  }

  /**
   * Creates event list user interface. Sets views header text, content material and footer buttons.
   */
  private void createUi() {
    // this.setSizeFull();
    choiceOptionGroup = new OptionGroup();
    choiceOptionGroup.setImmediate(true);
    choiceOptionGroup.setStyleName(CssStyleNames.TASKCHOICESOPTGROUP);
    choiceOptionGroup.addListener(
        new Property.ValueChangeListener() {

      private static final long serialVersionUID = 1L;

      @Override
      public void valueChange(ValueChangeEvent event) {
        viewhandler.setFooterButtonEnabled(FOOTER_BUTTON_RIGHT, true);
        if (event.getProperty().getValue() != null) {
```

```
          Object id = event.getProperty().getValue();

          selectedChoiceID = (Integer) choiceOptionGroup.getItem(id).getItemProperty(PaattiColumnNames.CHOICE_choiceID).getValue();
          fireTaskChoiceChangeEvent();
        }
      }
    });

    descriptionLabel = new Label();
    contentLabel = new Label();
    contentLabel.setReadOnly(true);
    contentLabel.setContentMode(Label.CONTENT_XHTML);
    this.setSpacing(true);
    this.setMargin(true);
    updateFooterAndHeader();

  }

  /**
   * Updates footer buttons according to the event phase.
   * <p/>
   * EVENT_START = 0 EVENT_MIDDLE = 1 EVENT_END = 2
   * <p/>
   * @param eventPhase The phase of the event.
   */
  protected void updateFooter(int eventPhase) {
    if (eventPhase == EVENT_START) {
      viewhandler.setFooterButtonEnabled(MobileView.FOOTER_BUTTON_MIDDLE, false);
      viewhandler.setFooterButtonEnabled(MobileView.FOOTER_BUTTON_RIGHT, true);
    } else if (eventPhase == EVENT_MIDDLE) {
      viewhandler.setFooterButtonEnabled(MobileView.FOOTER_BUTTON_MIDDLE, true);
      viewhandler.setFooterButtonEnabled(MobileView.FOOTER_BUTTON_RIGHT, true);
      viewhandler.setFooterButtonIcon(MobileView.FOOTER_BUTTON_RIGHT,
          Resources.getThemeResource(Resources.ICON_ARROW_RIGHT_64));
    }

  }

  @Override
  public void setContentViewLogic(MobileViewLogic logic) {
    this.eventViewLogic = (EventViewLogic) logic;
  }

  @Override
  public void updateFooterAndHeader() {
    viewhandler.setFooterButtonIcon(MobileView.FOOTER_BUTTON_LEFT, Resources.getThemeResource(Resources.ICON_CANCEL_64));
```

```
    viewhandler.setFooterButtonEnabled(MobileView.FOOTER_BUTTON_MIDDLE, false);
    viewhandler.setFooterButtonIcon(MobileView.FOOTER_BUTTON_MIDDLE,
        Resources.getThemeResource(Resources.ICON_ARROW_LEFT_64));
    viewhandler.setFooterButtonIcon(MobileView.FOOTER_BUTTON_RIGHT,
        Resources.getThemeResource(Resources.ICON_ARROW_RIGHT_64));
  }

  /**
   * Start a new event.
   * <p/>
   * @param taskContainer   All the tasks that are in the event.
   * @param choiceContainer All the choices that are in tasks (that are in the event)
   * @param eventName       The name of the event.
   */
  public void startEvent(SQLContainer taskContainer, SQLContainer choiceContainer, String eventName) {
    this.selectedChoiceID = -1;
    this.currentTaskID = -1;
    viewhandler.setHeaderText(eventName);
    eventViewLogic.startEvent(taskContainer, choiceContainer);
  }

  /**
   * @param taskItem    param taskChoices
   * @param taskChoices
   */
  protected void setCurrentTask(Item taskItem, IndexedContainer taskChoices) {

    // Remove everything from the previous task.
    this.removeAllComponents();
    this.requestRepaint();
    viewhandler.setFooterButtonEnabled(FOOTER_BUTTON_RIGHT, false);

    // Set Task description as the datasource for the label.
    if (taskItem.getItemProperty(PaattiColumnNames.TASK_description) != null) {
      descriptionLabel = new Label(taskItem.getItemProperty(PaattiColumnNames.TASK_description).getValue().toString());
      descriptionLabel.setContentMode(Label.CONTENT_XHTML);
      this.addComponent(descriptionLabel);

    }

    // should never be null
    Integer taskType = Integer.parseInt(
        taskItem.getItemProperty(PaattiColumnNames.TASK_TASKTYPE_taskTypeID).getValue().toString());

    if (taskType.equals(TASKTYPE_TEXT)) {
      // If the tasktype is TEXT, there can only be one choice, and therefore only one item in the taskChoices.
```

```
      // The id of the choice-to-be-made can be taken from that item.

      Item taskChoiceItem = taskChoices.getItem(taskChoices.firstItemId());

      selectedChoiceID = (Integer) taskChoiceItem.getItemProperty(PaattiColumnNames.CHOICE_choiceID).getValue();

      if (taskItem.getItemProperty(PaattiColumnNames.TASK_content).getValue() != null) {
        contentLabel = new Label(taskItem.getItemProperty(PaattiColumnNames.TASK_content).getValue().toString());
        contentLabel.setContentMode(Label.CONTENT_XHTML);
        this.addComponent(contentLabel);
      }

      fireTaskChoiceChangeEvent();
    } else if (taskType.equals(TASKTYPE_RADIO)) {
      choiceOptionGroup.setContainerDataSource(taskChoices);
      choiceOptionGroup.setItemCaptionPropertyId(PaattiColumnNames.CHOICE_description);
      choiceOptionGroup.select(null);
      this.addComponent(choiceOptionGroup);
      this.setComponentAlignment(choiceOptionGroup, Alignment.MIDDLE_LEFT);
      viewhandler.setFooterButtonEnabled(FOOTER_BUTTON_RIGHT, false);
    } else if (taskType.equals(TASKTYPE_AUDIO) || taskType.equals(TASKTYPE_VIDEO) || taskType.equals(TASKTYPE_SENSOR)) {

      Object firstItemId = taskChoices.firstItemId();
      Item item = taskChoices.getItem(firstItemId);

      selectedChoiceID = (Integer) item.getItemProperty(PaattiColumnNames.CHOICE_choiceID).getValue();

      if (taskType.equals(TASKTYPE_SENSOR)) {
        // TODO: Do something appropriate with this task type.
        contentLabel.setPropertyDataSource(taskItem.getItemProperty(PaattiColumnNames.TASK_content));
        this.addComponent(contentLabel);
      } else {
        Video audiovideo = new Video();

        audiovideo.setSizeFull();
        audiovideo.setSource(new ExternalResource(taskItem.getItemProperty(PaattiColumnNames.TASK_url).toString()));
        audiovideo.setShowControls(true);
        this.addComponent(audiovideo);

      }
      fireTaskChoiceChangeEvent();
    }
  }

  /**
   * Add a listener for task choice changes.
```

```
   * <p/>
   * @param listener The listener to be added.
   */
  public void addListener(TaskChoiceChangeEventListener listener) {
    listeners.add(listener);
  }

  /**
   * Remove a task choice changes listener
   * <p/>
   * @param listener The listener to be removed.
   */
  public void removeListener(TaskChoiceChangeEventListener listener) {
    listeners.remove(listener);
  }

  /**
   * Fires a TaskChoiceChangeEvent.
   * Also sets the right footer button enabled.
   */
  private void fireTaskChoiceChangeEvent() {
    viewhandler.setFooterButtonEnabled(FOOTER_BUTTON_RIGHT, true);
    for (TaskChoiceChangeEventListener taskChoiceChangeEventListener : listeners) {
      taskChoiceChangeEventListener.taskChange(new TaskChoiceChangeEvent(this, currentTaskID, selectedChoiceID));
    }
  }
}
```