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


import com.vaadin.data.Item;
import com.vaadin.data.util.sqlcontainer.SQLContainer;
import com.vaadin.ui.Component;
import fi.paatti.mobile.applicationinfo.ViewIds;
import fi.paatti.mobile.mobileview.MobileView;
import fi.paatti.mobile.views.listview.ListViewLogic;
import fi.paatti.mobile.paattiapplication.MobileViewHandler;
import fi.paatti.paattidatabaseutils.names.PaattiColumnNames;
import fi.paatti.paattidatabaseutils.names.PaattiTableNames;
import java.util.HashMap;
import org.joda.time.DateTime;


/**
 * The logic for the event list view.
 * <p/>
 * @author Lauri Satokangas, lauri.n.satokangas@student.jyu.fi @date 16.3.2012
 */
public class EventListViewLogic extends ListViewLogic {

  private final EventListView eventListView;
  private final String[] weekdays = new String[] { "MA", "TI", "KE", "TO", "PE", "LA", "SU"};
  private boolean rightButtonPressed = false;
  private boolean middleButtonPressed = false;

  /**
   * Constructor.
   * <p/>
   * @param viewHandler   The apllication´s view handler.
   * @param eventListView The event list view.
   */
  public EventListViewLogic(MobileViewHandler viewHandler, EventListView eventListView) {
    super(viewHandler, eventListView);
    this.eventListView = eventListView;

  }

  @Override
  public void footerButtonAction(Object buttonId) {
    getViewHandler().setFooterButtonEnabled(buttonId, true);
    if (buttonId == EventListView.FOOTER_BUTTON_LEFT) {
      getViewHandler().setContentView(ViewIds.PREVIOUSVIEW);
    } else if (buttonId == EventListView.FOOTER_BUTTON_MIDDLE) {
```

```
      if (middleButtonPressed) {
        eventListView.showAllEvents();
      } else {
        eventListView.showOnlyUnavailableEvents();
        rightButtonPressed = false;
        getViewHandler().setFooterButtonPressed(EventListView.FOOTER_BUTTON_RIGHT, false);
      }
      middleButtonPressed = !middleButtonPressed;
      getViewHandler().setFooterButtonPressed(buttonId, middleButtonPressed);

    } else if (buttonId == MobileView.FOOTER_BUTTON_RIGHT) {

      if (rightButtonPressed) {
        eventListView.showAllEvents();
      } else {
        eventListView.showOnlyUnfinishedEvents();
        middleButtonPressed = false;
        getViewHandler().setFooterButtonPressed(EventListView.FOOTER_BUTTON_MIDDLE, false);
      }
      rightButtonPressed = !rightButtonPressed;
      getViewHandler().setFooterButtonPressed(buttonId, rightButtonPressed);

    }

  }

  @Override
  public HashMap<Object, Component> getFooterComponents() {
    return getViewHandler().getFooterComponents();

  }

  @Override
  public void listItemClicked(Object eventId) {
    SQLContainer taskContainer = getViewHandler().getDbService().getTaskContainer(eventId);
    SQLContainer choiceContainer = getViewHandler().getDbService().getChoiceContainer(eventId);
    String eventTitle = getViewHandler().getDbService().getCellValueFromTable(PaattiTableNames.EVENT, ⇙
PaattiColumnNames.EVENT_title, PaattiColumnNames.EVENT_eventID, eventId).toString();

    getViewHandler().createAndStartEvent(taskContainer, choiceContainer, eventTitle);
  }

  /**
   * Get the event time as String.
   * <p/>
```

```
   * Format: weekday dd.mm.yyyy [klo] hh:mm.
   * <p/>
   * @param eventItem The event item
   * <p/>
   * @return The event time as String or empty String if the time string could not be created.
   */
  public String getEventTimeAsString(Item eventItem) {

    // TODO: code below works only when eventtime is absolute.
    // fix this to also create the time string when event time is not absolute.
    if (!eventItem.getItemProperty(PaattiColumnNames.EVENT_EVENTTIME_eventTimeID).getValue().toString().equals("-1")) {

      String weekday = (eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_weekday).getValue() != null)
          ? weekdays[(Integer) eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_weekday).getValue()]
          : "";
      String day = (eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_day).getValue() != null)
          ? eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_day).getValue().toString()
          : "";
      String month = (eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_month).getValue() != null)
          ? eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_month).getValue().toString()
          : "";
      String year = (eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_month).getValue() != null)
          ? eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_year).getValue().toString()
          : "";
      String hour = (eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_fromHour).getValue() != null)
          ? eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_fromHour).getValue().toString()
          : "";
      String minute = (eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_minute).getValue() != null)
          ? eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_minute).getValue().toString()
          : "";

      minute = (minute.length() == 1) ? "0" + minute : minute;
      return weekday + " " + day + "." + month + "." + year.toString() + " klo: " + hour + ":" + minute;
    } else {
      return "";
    }
  }

  /**
   * Checks the event time and returns the correct status for the event based on its time.
   * <p/>
   * @param eventItem The item that holds the events (time) values.
   * <p/>
   * @return The status of the event.
   */
  public int getEventStatus(Item eventItem) {
```

```
    if ("-1".equals(eventItem.getItemProperty(PaattiColumnNames.EVENT_EVENTTIME_eventTimeID).getValue().toString())) {
      return EventListItem.STATUS_UNAVAILABLE;
    }
    String day = (eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_day).getValue() != null)
        ? eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_day).getValue().toString()
        : "";
    String month = (eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_month).getValue() != null)
        ? eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_month).getValue().toString()
        : "";
    String year = (eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_month).getValue() != null)
        ? eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_year).getValue().toString()
        : "";
    String hour = (eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_fromHour).getValue() != null)
        ? eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_fromHour).getValue().toString()
        : "";
    String minute = (eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_minute).getValue() != null)
        ? eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_minute).getValue().toString()
        : "";

    Integer hoursLeft = (eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_hoursLeft).getValue() != null)
        ? (Integer) eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_hoursLeft).getValue()
        : 0;

    // DateTime uses ISO 8601 format for creating date and time (VVVV-KK-PPTHH:MM)
    DateTime eventStartingTime = new DateTime(year + "-" + month + "-" + day + "T" + hour + ":" + minute);
    DateTime eventDueTime = eventStartingTime.plusHours(hoursLeft);

    if (eventDueTime.isBeforeNow()) {
      return EventListItem.STATUS_UNFINISHED;
    } else if (eventStartingTime.isAfterNow()) {
      return EventListItem.STATUS_UNAVAILABLE;
    } else if (eventStartingTime.isEqualNow() || (eventStartingTime.isBeforeNow() && eventDueTime.isAfterNow())) {
      return EventListItem.STATUS_AVAILABLE;
    }

    return -1;
  }

  /**
   * Get the estimated time of the event as a String.
   * <p/>
   * The format of the returned String is hh:mm.
   * <p/>
   * @param eventItem The event item.
   * <p/>
```

```
   * @return The estimated time as a String or empty String if EVENT_estimated time was null.
   */
  public String getEstimatedTimeAsString(Item eventItem) {

    if (eventItem == null || eventItem.getItemProperty(PaattiColumnNames.EVENT_estimatedTime).getValue() == null) {
      return "";
    }

    Integer estimatedMinutes = (Integer) eventItem.getItemProperty(PaattiColumnNames.EVENT_estimatedTime).getValue();
    int hours = estimatedMinutes / 60;
    int minutes = estimatedMinutes % 60;

    return (String.format("Arvioitu kesto: %02d:%02d", hours, minutes));
  }
}
```