```
package fi.paatti.research.paattiapplication.views.scheduleview;


import com.vaadin.data.Container.Filter;
import com.vaadin.data.Item;
import com.vaadin.data.Property;
import com.vaadin.data.util.IndexedContainer;
import com.vaadin.data.util.filter.Compare;
import com.vaadin.data.util.sqlcontainer.SQLContainer;
import com.vaadin.ui.Button.ClickEvent;
import com.vaadin.ui.*;
import fi.paatti.paattidatabaseutils.names.PaattiColumnNames;
import fi.paatti.paattidatabaseutils.names.PaattiTableNames;
import fi.paatti.research.paattiapplication.ElementNames;
import fi.paatti.research.paattiapplication.views.ApplicationView;
import fi.paatti.research.paattiapplication.views.ApplicationViewTab;
import java.text.DateFormat;
import java.util.Calendar;


/**
 * The form presents the schedule editing in schedule view.
 *
 * @author Jari Salokangas, jari.p.t.salokangas@student.jyu.fi
 * @author Lauri Satokangas, lauri.n.satokangas@student.jyu.fi
 */
public class ScheduleEditTab extends ApplicationViewTab {

  private static final long serialVersionUID = 1L;
  private final String[] tableLeftColumns = {
    PaattiColumnNames.EVENT_title, PaattiColumnNames.EVENT_description, PaattiColumnNames.EVENT_estimatedTime
  };
  private final String[] tableRightColumns = {
    PaattiColumnNames.EVENT_title, PaattiColumnNames.EVENTTIME_day, PaattiColumnNames.EVENTTIME_month,
    PaattiColumnNames.EVENTTIME_year, PaattiColumnNames.EVENTTIME_fromHour, PaattiColumnNames.EVENTTIME_minute
  };
  private VerticalLayout availableEventsLayout;
  private Table availableEventsTable;
  private Button addScheduleButton;
  private VerticalLayout scheduledEventsLayout;
  private Table scheduledEventsTable;
  private Button removeButton;
  private PopupDateField eventDatePopupDateField;
  private IndexedContainer scheduledEventsIndexedCont;
  private HorizontalLayout scheduledEventsActionsLayout;
  private CheckBox voluntaryEventsCheckbox;
```

```
  /**
   * Constructor of the class.
   */
  public ScheduleEditTab(ApplicationView view) {
    super(view);
    createLayout();

    // TODO: move somewhere else
    scheduledEventsIndexedCont = new IndexedContainer();
    createNewScheduleContainer();
  }

  /**
   * Creates a new container for event items.
   */
  private void createNewScheduleContainer() {
    for (String s : PaattiColumnNames.getColumnNames(PaattiTableNames.EVENT)) {
      scheduledEventsIndexedCont.addContainerProperty(s, String.class, null);
    }
    for (String s : PaattiColumnNames.getColumnNames(PaattiTableNames.EVENTTIME)) {
      scheduledEventsIndexedCont.addContainerProperty(s, String.class, null);
    }

    scheduledEventsTable.setContainerDataSource(scheduledEventsIndexedCont);
    scheduledEventsTable.setVisibleColumns(tableRightColumns);
    scheduledEventsTable.setColumnHeaders(ElementNames.SCHEDULE_VIEW_TABLE_RIGHT_COLUMN_TEXTS);
  }

  /**
   * {@inheritDoc }
   */
  @Override
  protected void fillTab(Item schedule) {
    scheduledEventsIndexedCont.removeAllItems();
    if (schedule != null) {

      SQLContainer scheduledEvents = view.getDBConnection().getEventsInSchedule(
          schedule.getItemProperty(PaattiColumnNames.SCHEDULE_scheduleID).getValue());

      if (scheduledEvents.firstItemId() != null) {
        for (int i = 0; i < scheduledEvents.size(); i++) {

          Object containerItemID = scheduledEvents.getIdByIndex(i);
          Item item = scheduledEvents.getItem(containerItemID);
          Object id = scheduledEventsIndexedCont.addItem();
```

```
        for (String s1 : PaattiColumnNames.getColumnNames(PaattiTableNames.EVENT)) {
          if (item.getItemProperty(s1) != null) {
            scheduledEventsIndexedCont.getItem(id).getItemProperty(s1).setValue(item.getItemProperty(s1).getValue());
          }
        }
        for (String s2 : PaattiColumnNames.getColumnNames(PaattiTableNames.EVENTTIME)) {
          if (item.getItemProperty(s2).getValue() != null) {
            scheduledEventsIndexedCont.getItem(id).getItemProperty(s2).setValue(item.getItemProperty(s2).getValue());
          }
        }
      }
    }
    scheduledEventsActionsLayout.setEnabled(false);
  }
  setEnabled(true);
}

/**
 * Sets left tables data source.
 */
private void setAvailableEventsTableDataSource() {
  Filter filter = new Compare.Equal(PaattiColumnNames.EVENT_origEventID, -1);

  availableEventsTable.setContainerDataSource(
      view.getDBConnection().getSQLContainerFromDBTableFiltered(PaattiTableNames.EVENT,
      PaattiColumnNames.EVENT_eventID, true, filter));
  availableEventsTable.setVisibleColumns(tableLeftColumns);
  availableEventsTable.setColumnHeaders(ElementNames.SCHEDULE_VIEW_TABLE_LEFT_COLUMN_TEXTS);
}

/**
 * Copies data from left tables selected event to the right tables new item.
 */
private void copySelectedTableItem() {
  if (availableEventsTable.getValue() != null) {
    Item i = availableEventsTable.getItem(availableEventsTable.getValue());
    Object id = scheduledEventsIndexedCont.addItem();

    for (String string : PaattiColumnNames.getColumnNames(PaattiTableNames.EVENT)) {
      scheduledEventsIndexedCont.getItem(id).getItemProperty(string).setValue(i.getItemProperty(string).getValue());
    }

    scheduledEventsIndexedCont.getItem(id).getItemProperty(PaattiColumnNames.EVENT_status).setValue(0);
    scheduledEventsIndexedCont.getItem(id).getItemProperty(PaattiColumnNames.EVENT_origEventID).setValue(
        i.getItemProperty(PaattiColumnNames.EVENT_eventID).getValue());
```

```
      // scheduledEventsIndexedCont.getItem(id).getItemProperty(PaattiColumnNames.EVENT_EVENTTIME_eventTimeID).setValue(-1);
      setEventAsVoluntary(scheduledEventsIndexedCont.getItem(id));
    }
  }

  /**
   * Returns the schedule container.
   * <p/>
   * @return scheduledEventsCont = container holding schedules events
   */
  public IndexedContainer getScheduleContainer() {
    return scheduledEventsIndexedCont;
  }

  /**
   * Sets selected table items properties.
   */
  private void setTimeForEvent(Item eventItem) {
    if (eventItem != null) {
      try {
        Object[] valueTokens = createDateTime();

        eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_day).setValue(valueTokens[0]);
        eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_month).setValue(valueTokens[1]);
        eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_year).setValue(valueTokens[2]);
        eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_fromHour).setValue(valueTokens[3]);
        eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_minute).setValue(valueTokens[4]);
        eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_weekday).setValue(valueTokens[5]);
        eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_absolute).setValue(valueTokens[6]);
        eventItem.getItemProperty(PaattiColumnNames.EVENTTIME_hoursLeft).setValue(valueTokens[7]);
        // Set event as nonvoluntary.
        eventItem.getItemProperty(PaattiColumnNames.EVENT_EVENTTIME_eventTimeID).setValue(-2);

        getWindow().showNotification(
            "Tapahtuma aikataulutettu ajankohdalle: " + valueTokens[0] + "." + valueTokens[1] + "." + valueTokens[2]
            + " klo " + valueTokens[3] + ":" + valueTokens[4]);
      } catch (Exception ex) {
        getWindow().showNotification("Aseta päivämäärä painikkeen kautta.");
      }
    }
  }

  /**
   * Formats datetime to useable form.
   * <p/>
   * @return date, time and weekday
```

```
   */
  private Object[] createDateTime() {
    DateFormat dateFormat = DateFormat.getDateTimeInstance(DateFormat.DATE_FIELD, DateFormat.FULL);

    dateFormat.format(eventDatePopupDateField.getValue());
    Calendar cal = dateFormat.getCalendar();

    int week = cal.get(Calendar.DAY_OF_WEEK) - 1;

    if (week < 1) {
      week = 7;
    }

    String minute;

    if (cal.get(Calendar.MINUTE) < 10) {
      minute = "0" + cal.get(Calendar.MINUTE);
    } else {
      minute = "" + cal.get(Calendar.MINUTE);
    }

    Object[] dateTimeTokens = {
      cal.get(Calendar.DAY_OF_MONTH), // day
      cal.get(Calendar.MONTH) + 1, // month
      cal.get(Calendar.YEAR), // year
      cal.get(Calendar.HOUR_OF_DAY), // hour
      minute, week, 0, 168, // the event remains doable for 7 days
    };

    return dateTimeTokens;
  }

  /**
   * Set the event as voluntary event. For now, this means that the event will
   * have no EventTime (the eventTimeID will be set to -1)
   */
  private void setEventAsVoluntary(Item eventItem) {
    eventItem.getItemProperty(PaattiColumnNames.EVENT_EVENTTIME_eventTimeID).setValue(-1);
  }

  /**
   * {@inheritDoc }
   */
  @Override
  protected void createLayout() {
    availableEventsTable = new Table();
```

```
    availableEventsTable.setSelectable(true);
    availableEventsTable.setSortDisabled(true);
    availableEventsTable.setWidth("100%");
    availableEventsTable.setRequired(true);

    addScheduleButton = new Button(ElementNames.SCHEDULE_VIEW_ADD_EVENT_TO_SCHEDULE_BUTTON_LABEL);
    addScheduleButton.addListener(this);

    availableEventsLayout = new VerticalLayout();
    availableEventsLayout.addComponent(availableEventsTable);
    availableEventsLayout.addComponent(addScheduleButton);

    scheduledEventsTable = new Table();
    scheduledEventsTable.setImmediate(true);
    scheduledEventsTable.setSelectable(true);
    scheduledEventsTable.setSortDisabled(true);
    scheduledEventsTable.setWidth("100%");
    scheduledEventsTable.setRequired(true);

    scheduledEventsTable.addListener(new Property.ValueChangeListener() {

      private static final long serialVersionUID = 1L;

      /**
       * When item is selected in the table the actions layout below the
       * table is set to enabled. If the item is voluntary event the
       * checkbox will be checked.
       */
      public void valueChange(Property.ValueChangeEvent event) {
        Object selectedItemID = scheduledEventsTable.getValue();

        if (selectedItemID == null) {
          scheduledEventsActionsLayout.setEnabled(false);
          voluntaryEventsCheckbox.setValue(false);
          eventDatePopupDateField.setValue(null);
        } else {
          scheduledEventsActionsLayout.setEnabled(true);
          Item item = scheduledEventsTable.getItem(selectedItemID);

          // if -1 then event is voluntary.
          if ("-1".equals(item.getItemProperty(PaattiColumnNames.EVENT_EVENTTIME_eventTimeID).getValue().toString())) {
            voluntaryEventsCheckbox.setValue(true);
            eventDatePopupDateField.setEnabled(false);
          } else {
            voluntaryEventsCheckbox.setValue(false);
            eventDatePopupDateField.setEnabled(true);
```

```
        }
      }
    }
  });

  removeButton = new Button(ElementNames.SCHEDULE_VIEW_REMOVE_EVENT_FROM_SCHEDULE_BUTTON_LABEL);
  removeButton.addListener(this);

  eventDatePopupDateField = new PopupDateField();
  eventDatePopupDateField.setResolution(PopupDateField.RESOLUTION_MIN);
  eventDatePopupDateField.setImmediate(true);
  eventDatePopupDateField.setInputPrompt(ElementNames.SCHEDULE_VIEW_EVENTDATE_SELECT_PROMPT);

  eventDatePopupDateField.addListener(new Property.ValueChangeListener() {

    private static final long serialVersionUID = 1L;

    public void valueChange(Property.ValueChangeEvent event) {
      if (scheduledEventsTable.getValue() != null) {
        // Get the selected item and send it to the setTimeForEvent()
        setTimeForEvent(scheduledEventsTable.getItem(scheduledEventsTable.getValue()));
      }
    }
  });

  Label voluntaryCheckBoxLabel = new Label(ElementNames.SCHEDULE_VIEW_SET_VOLUNTARY_CHECKBOX_LABEL);

  voluntaryEventsCheckbox = new CheckBox();
  voluntaryEventsCheckbox.setValue(false);
  voluntaryEventsCheckbox.setImmediate(true);
  voluntaryEventsCheckbox.addListener(this);

  scheduledEventsActionsLayout = new HorizontalLayout();
  scheduledEventsActionsLayout.addComponent(removeButton);
  scheduledEventsActionsLayout.addComponent(eventDatePopupDateField);
  scheduledEventsActionsLayout.addComponent(voluntaryCheckBoxLabel);
  scheduledEventsActionsLayout.addComponent(voluntaryEventsCheckbox);
  scheduledEventsActionsLayout.setSpacing(true);
  scheduledEventsActionsLayout.setEnabled(false);

  scheduledEventsLayout = new VerticalLayout();
  scheduledEventsLayout.addComponent(scheduledEventsTable);
  scheduledEventsLayout.addComponent(scheduledEventsActionsLayout);

  VerticalLayout layout = new VerticalLayout();
```

```
    layout.setSpacing(true);
    layout.setMargin(true);
    layout.addComponent(availableEventsLayout);
    layout.addComponent(scheduledEventsLayout);
    layout.setSpacing(true);
    setCompositionRoot(layout);

    setAvailableEventsTableDataSource();
  }

  /**
   * {@inheritDoc }
   */
  @Override
  protected void clearTab() {
    fillTab(null);
    setEnabled(true);
  }

  /**
   * {@inheritDoc}
   * <p/>
   * The handled button click events are:
   * <ul>
   * <li>remove the selected event from the schedule,</li>
   * <li>add the selected event to the schedule, and</li>
   * <li>set the selected event as voluntary.</li>
   * </ul>
   */
  @Override
  public void buttonClick(ClickEvent event) {
    Button buttonCaption = event.getButton();
    String msg = null;

    if (buttonCaption.equals(removeButton)) {
      if (scheduledEventsTable.isValid()) {
        scheduledEventsIndexedCont.removeItem(scheduledEventsTable.getValue());
        scheduledEventsActionsLayout.setEnabled(false);
      } else {
        msg = ElementNames.SOMETHING_REQUIRED;
      }
    } else if (buttonCaption.equals(addScheduleButton)) {
      if (availableEventsTable.isValid()) {
        copySelectedTableItem();
      } else {
        msg = ElementNames.SOMETHING_REQUIRED;
```

```
      }
    } else if (buttonCaption.equals(voluntaryEventsCheckbox)) {
      boolean voluntaryEvent = event.getButton().booleanValue();
      Item eventItem = scheduledEventsTable.getItem(scheduledEventsTable.getValue());

      eventDatePopupDateField.setValue(null);
      eventDatePopupDateField.setEnabled(!voluntaryEvent);
      if (eventItem != null && voluntaryEvent) {
        setEventAsVoluntary(eventItem);
      }
    }

    if (msg != null) {
      getWindow().showNotification(msg);
    }
  }
}
```