```java
package fi.paatti.paattidatabaseutils.dbservice.querydelegates;


import com.vaadin.data.util.sqlcontainer.RowItem;
import com.vaadin.data.util.sqlcontainer.TemporaryRowId;
import com.vaadin.data.util.sqlcontainer.query.generator.StatementHelper;
import com.vaadin.data.util.sqlcontainer.query.generator.filter.QueryBuilder;
import fi.paatti.paattidatabaseutils.names.PaattiColumnNames;
import fi.paatti.paattidatabaseutils.names.PaattiTableNames;
import java.sql.*;


/**
 * A delegate for TASK table.
 *
 * @author Lauri Satokangas, lauri.n.satokangas@student.jyu.fi
 * @since 0.0.2
 */
public class TaskQueryDelegate extends PaattiQueryDelegate {
  private static final long serialVersionUID = 1L;
  private final Object eventId;

  /**
   * Constructor.
   * <p/>
   * @param eventId The ID of the event.
   */
  public TaskQueryDelegate(Object eventId) {
    super(PaattiTableNames.TASK, PaattiColumnNames.TASK_taskID,
        PaattiColumnNames.getColumnNamesInArray(PaattiTableNames.TASK));
    this.eventId = eventId;
  }

  /**
   * {@inheritDoc }
   */
  public int storeRow(Connection conn, RowItem row) throws SQLException {
    PreparedStatement storeStatement;
    StringBuilder querySb = new StringBuilder();
    int retval;

    if (row.getId() instanceof TemporaryRowId) {
      querySb.append(getInsertQueryString());
      storeStatement = conn.prepareStatement(querySb.toString(), Statement.RETURN_GENERATED_KEYS);
      setRowValues(storeStatement, row);
      retval = storeStatement.executeUpdate();
      ResultSet generatedKeys = storeStatement.getGeneratedKeys();

      if (generatedKeys.first()) {
        Object object = generatedKeys.getObject(1);

        fireNewRowIdEvent(object);
      }
    } else {
      querySb.append(getUpdateQueryString());
      storeStatement = conn.prepareStatement(querySb.toString());
      storeStatement.setInt(10, (Integer) row.getItemProperty(PaattiColumnNames.TASK_taskID).getValue());
      setRowValues(storeStatement, row);
      retval = storeStatement.executeUpdate();

    }

    storeStatement.close();
    return retval;
  }

  /**
   * {@inheritDoc }
   * <p/>
   * The column order is (taken from the {@link PaattiColumnNames}:
   * <p> TASK_rowStatus <br> TASK_TASKTYPE_taskTypeID <br>
   * TASK_EVENT_eventID<br> TASK_sequence<br>
   * TASK_url<br> TASK_content<br> TASK_description<br> TASK_taskID<br>
   * <p/>
   */
  @Override
  public void setRowValues(PreparedStatement statement, RowItem row) throws SQLException {
    statement.setString(1, row.getItemProperty(PaattiColumnNames.TASK_rowStatus).toString());
    statement.setInt(2, (Integer) row.getItemProperty(PaattiColumnNames.TASK_TASKTYPE_taskTypeID).getValue());
    statement.setInt(3, (Integer) row.getItemProperty(PaattiColumnNames.TASK_EVENT_eventID).getValue());
    statement.setInt(4, (Integer) row.getItemProperty(PaattiColumnNames.TASK_sequence).getValue());
    statement.setString(5, (String) row.getItemProperty(PaattiColumnNames.TASK_url).getValue());
    statement.setString(6, (String) row.getItemProperty(PaattiColumnNames.TASK_content).getValue());
    statement.setString(7, (String) row.getItemProperty(PaattiColumnNames.TASK_description).getValue());
    statement.setInt(8, (Integer) row.getItemProperty(PaattiColumnNames.TASK_posX).getValue());
    statement.setInt(9, (Integer) row.getItemProperty(PaattiColumnNames.TASK_posY).getValue());
  }

  /**
   * Get the StatementHelper for the SELECT-query.
   * <p/>
   * @param offset Starting row
   * @param limit Number of rows to get.
   * @return @throws UnsupportedOperationException
   */
  public StatementHelper getQueryStatement(int offset, int limit) {
    StatementHelper queryHelper = new StatementHelper();
    StringBuilder querySb = new StringBuilder();

    querySb.append(" SELECT * FROM ").append(PaattiTableNames.TASK);
    querySb.append(" WHERE ").append(PaattiColumnNames.TASK_EVENT_eventID).append(" = ").append(eventId);
    querySb.append(" AND ").append(PaattiColumnNames.TASK_rowStatus).append(" = '").append(ROWSTATUS_ACTIVE).append("'");

    if (getFilters() != null && !getFilters().isEmpty()) {
      querySb.append(getFilterStringWithoutWhere(queryHelper, true));
    }

    querySb.append(" ORDER BY ").append(PaattiColumnNames.TASK_sequence).append(" ASC");
    querySb.append(" LIMIT ").append(offset).append(", ").append(limit);

    queryHelper.setQueryString(querySb.toString());
    return queryHelper;

  }

  /**
   * {@inheritDoc }
   */
  @Override
  public StatementHelper getCountStatement() throws UnsupportedOperationException {
    StatementHelper countHelper = new StatementHelper();
    StringBuilder querySb = new StringBuilder();

    querySb.append("SELECT COUNT (*) FROM ").append(getTableName());

    if (getFilters() != null && !getFilters().isEmpty()) {
      querySb.append(QueryBuilder.getWhereStringForFilters(getFilters(), countHelper));
      querySb.append(" AND ").append(PaattiColumnNames.TASK_EVENT_eventID).append(" = ").append(eventId);
      querySb.append(" AND ").append(PaattiColumnNames.TASK_rowStatus).append(" = '").append(ROWSTATUS_ACTIVE).append(
          "'");
    } else {
      querySb.append(" WHERE ").append(PaattiColumnNames.TASK_EVENT_eventID).append(" = ").append(eventId);
      querySb.append(" AND ").append(PaattiColumnNames.TASK_rowStatus).append(" = '").append(ROWSTATUS_ACTIVE).append(
          "'");
    }

    countHelper.setQueryString(querySb.toString());
    return countHelper;
  }
}
```