Browser calls endpoint:

{relative URL}/trackScheduleInfo

This activates the CalendarController

**package** org.lds.ics.calendar;

import **static** org.springframework.web.bind.annotation.RequestMethod.GET;

import javax.inject.Inject;

import lds.stack.spring.web.Response;

import lds.stack.spring.web.ResponseBuilder;

import org.lds.ics.napi.common.MissionaryTypeCode;

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.context.annotation.Scope;

import org.springframework.security.access.annotation.Secured;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

@RestController

@**Scope**("prototype")

@RequestMapping("api/calendar")

@Secured({ "ROLE\_AssignmentMeetingAssistant"

 , "ROLE\_AssignmentMeetingManager"

 , "ROLE\_DepartmentSeniorMissionaryCoordinator"

 , "ROLE\_SeniorAssignmentAssistant"

 , "ROLE\_SeniorAssignmentCoordinator",

 "ROLE\_ROLE\_ADMIN" })

**public** **class** CalendarController {

 @Inject @Qualifier("jdbcCalendarDao") **private** CalendarDao calendarDao;

 **public** **static** **final** **String** SENIOR\_COUPLE\_SISTER = MissionaryTypeCode.SeniorCoupleSister.code();

 **public** **static** **final** **String** SENIOR\_COUPLE\_ELDER = MissionaryTypeCode.SeniorCoupleElder.code();

 **public** **static** **final** **String** SENIOR\_SISTER = MissionaryTypeCode.SeniorSister.code();

 **public** **static** **final** **String** SENIOR\_ELDER = MissionaryTypeCode.SeniorElder.code();

 **public** **static** **final** **String** ELDER = MissionaryTypeCode.Elder.code();

 **public** **static** **final** **String** SISTER = MissionaryTypeCode.Sister.code();

 @RequestMapping(**value** = "/trackScheduleInfo", method = GET)

 **public** Response getTrackScheduleInfo(

 @RequestParam(**required** = **true**, **value** = "enabledMemberId") **Integer** enabledMemberId,

 @RequestParam(**required** = **false**, **value** = "trackId", defaultValue = "") **Integer** trackId,

 @RequestParam(**required** = **false**, **value** = "curriculumId", defaultValue = "") **Integer** curriculumId,

 @RequestParam(**required** = **false**, **value** = "mtcId", defaultValue = "") **String** mtcId,

 @RequestParam(**required** = **true**, **value** = "componentId", defaultValue = "") **Integer** componentId,

 @RequestParam(**required** = **false**, **value** = "numMonths", defaultValue = "12") **Integer** numMonths

 ) {

 **String**[] parts = mtcId.split(", "); // we **always** want **the** **first** MTC

 **return** ResponseBuilder.forContent("trackScheduleInfo", calendarDao.getTrackScheduleInfo(enabledMemberId, trackId, curriculumId, Integer.parseInt(parts[0]), componentId, numMonths)).get();

 }

}

The calendar Dao looks like this:

**package** org.lds.ics.calendar;

import **static** java.lang.String.format;

import **static** org.apache.commons.lang.StringUtils.equalsIgnoreCase;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.Date;

import java.util.List;

import java.util.Map;

import javax.inject.Inject;

import javax.sql.DataSource;

import oracle.jdbc.OracleTypes;

import org.springframework.dao.DataIntegrityViolationException;

import org.springframework.jdbc.core.RowMapper;

import org.springframework.jdbc.core.SqlOutParameter;

import org.springframework.jdbc.core.SqlParameter;

import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;

import org.springframework.jdbc.core.namedparam.SqlParameterSource;

import org.springframework.jdbc.core.simple.SimpleJdbcCall;

import org.springframework.stereotype.Repository;

@Repository

**public** **class** JdbcCalendarDao implements CalendarDao {

 **private** SimpleJdbcCall getTrackSchedules;

 @Inject

 **public** void setDataSource(DataSource dataSource) {

 this.getTrackSchedules = **new** SimpleJdbcCall(dataSource)

 .withSchemaName("<schemaname>")

 .withCatalogName("sr\_asmt\_pkg")

 .withProcedureName("get\_track\_schedules")

 .declareParameters(

 **new** SqlParameter("i\_emem\_id", OracleTypes.NUMBER),

 **new** SqlParameter("i\_trk\_id", OracleTypes.NUMBER),

 **new** SqlParameter("i\_curr\_id", OracleTypes.NUMBER),

 **new** SqlParameter("i\_mtc\_id", OracleTypes.NUMBER),

 **new** SqlParameter("i\_comp\_id", OracleTypes.NUMBER),

 **new** SqlParameter("i\_num\_months", OracleTypes.NUMBER),

 **new** SqlOutParameter("o\_lead\_time\_passed\_dt", OracleTypes.DATE),

 **new** SqlOutParameter("o\_anniversary\_dt", OracleTypes.DATE),

 **new** SqlOutParameter("o\_earliest\_start\_dt", OracleTypes.DATE),

 **new** SqlOutParameter("o\_stated\_avail\_dt", OracleTypes.DATE),

 **new** SqlOutParameter("o\_calendar\_start\_dt", OracleTypes.DATE),

 **new** SqlOutParameter("o\_calendar\_end\_dt", OracleTypes.DATE),

 **new** SqlOutParameter("o\_trksched\_list\_rc", OracleTypes.CURSOR));

 }

 @Override

 **public** TrackScheduleInfo getTrackScheduleInfo(**Integer** enabledMemberId, **Integer** trackId, **Integer** curriculumId, **Integer** mtcId, **Integer** componentId, **Integer** numMonths) {

 TrackScheduleInfo trackScheduleInfo = **new** TrackScheduleInfo();

 SqlParameterSource **in** = **new** MapSqlParameterSource()

 .addValue("i\_emem\_id", enabledMemberId)

 .addValue("i\_trk\_id", trackId)

 .addValue("i\_curr\_id", curriculumId)

 .addValue("i\_mtc\_id", mtcId)

 .addValue("i\_comp\_id", componentId)

 .addValue("i\_num\_months", numMonths);

 getTrackSchedules.returningResultSet("o\_trksched\_list\_rc", **new** RowMapper<TrackScheduleDTO>() {

 @Override

 **public** TrackScheduleDTO mapRow(ResultSet resultSet, **int** i) throws SQLException {

 TrackScheduleDTO trackScheduleDTO = **new** TrackScheduleDTO();

 trackScheduleDTO.setCalendarId(resultSet.getInt("cal\_id"));

 trackScheduleDTO.setCalendarDayOfWeek(resultSet.getInt("cal\_dow"));

 trackScheduleDTO.setTrainingStartDate(resultSet.getDate("train\_start\_dt"));

 trackScheduleDTO.setTrainingEndDate(resultSet.getDate("train\_end\_dt"));

 trackScheduleDTO.setTrackScheduleId(resultSet.getInt("trksched\_id"));

 trackScheduleDTO.setVisaAvailable(equalsIgnoreCase(resultSet.getString("visa\_avail\_yn"), "Y") ? Boolean.TRUE : Boolean.FALSE);

 **return** trackScheduleDTO;

 }

 });

 **Map**<**String**, **Object**> results;

 try {

 results = getTrackSchedules.execute(**in**);

 } catch (DataIntegrityViolationException e) {

 throw **new** DataIntegrityViolationException(format("No data found for enabledMemberId: %s, trackId: %s, curriculumId: %s mtcId: %s componentId: %s numMonths: %s in %s.%s"

 , enabledMemberId, trackId, curriculumId, mtcId, componentId, numMonths, getTrackSchedules.getCatalogName(), getTrackSchedules.getProcedureName()), e);

 }

 trackScheduleInfo.setLeadTimePassedDate((**Date**) results.get("o\_lead\_time\_passed\_dt"));

 trackScheduleInfo.setAnniversaryDate((**Date**) results.get("o\_anniversary\_dt"));

 trackScheduleInfo.setEarliestStartDate((**Date**) results.get("o\_earliest\_start\_dt"));

 trackScheduleInfo.setStatedAvailabilityDate((**Date**) results.get("o\_stated\_avail\_dt"));

 trackScheduleInfo.setCalendarStartDate((**Date**) results.get("o\_calendar\_start\_dt"));

 trackScheduleInfo.setCalendarEndDate((**Date**) results.get("o\_calendar\_end\_dt"));

 @SuppressWarnings({ "unchecked", "rawtypes" })

 **List**<TrackScheduleDTO> **list** = (**List**) results.get("o\_trksched\_list\_rc");

 trackScheduleInfo.setTrackScheduleDTOList(**list**);

 **return** trackScheduleInfo;

 }

}