package com.micropro.enam.pop.dao;

import javax.servlet.http.HttpServletRequest;

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

import org.springframework.stereotype.Component;

import org.springframework.transaction.annotation.Transactional;

import com.fasterxml.jackson.core.JsonProcessingException;

import com.fasterxml.jackson.databind.ObjectMapper;

import com.google.gson.JsonArray;

import com.google.gson.JsonObject;

import com.google.gson.JsonParser;

import com.micropro.enam.pop.common.ConnectionUtility;

import com.micropro.enam.pop.model.generated.BidCreationModel.PopbidcreationBody;

@Component

public class BidCreationDao {

private @Autowired ConnectionUtility utility;

@Transactional

public long crud(HttpServletRequest request, PopbidcreationBody body, String operation)

throws JsonProcessingException {

ObjectMapper mapper = new ObjectMapper();

JsonObject data = JsonParser.parseString("{\"popbidcreation\":" + mapper.writeValueAsString(body) + "}")

.getAsJsonObject();

return utility.executeDML(data, operation);

}

public String get(HttpServletRequest request, long id) {

JsonObject response = new JsonObject();

// Root Entity

String query = "SELECT \* FROM popbidcreation where popbidcreationid = " + id;

JsonArray data = utility.executeQueryOnPool(query, request);

if (data.size() > 0) {

response = (JsonObject) data.get(0);

}

return response.toString();

}

}

package com.micropro.enam.pop.model.generated;

import java.sql.Timestamp;

import java.util.Objects;

import com.fasterxml.jackson.annotation.JsonCreator;

import com.fasterxml.jackson.annotation.JsonFormat;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import com.fasterxml.jackson.annotation.JsonInclude;

import com.fasterxml.jackson.annotation.JsonInclude.Include;

import com.fasterxml.jackson.annotation.JsonProperty;

import io.swagger.annotations.ApiModel;

import io.swagger.annotations.ApiModelProperty;

public class BidCreationModel {

@JsonInclude(Include.ALWAYS)

@JsonIgnoreProperties(ignoreUnknown = true)

@ApiModel(description = "A request body model used in POST requests to create a Popbidcreation entity")

public static final class PopbidcreationBody {

private @ApiModelProperty(required = false, value = "REST representation of the 'POPBIDCREATIONID\_BODY' (Primary Key)") Long popbidcreationid;

private final @ApiModelProperty(required = true, value = "REST representation of the 'LOTID\_BODY' (required) (Size = 20)") Long lotid;

private final @ApiModelProperty(required = true, value = "REST representation of the 'PLATFORMID\_BODY' (required) (Size = 20)") String platformid;

private final @ApiModelProperty(required = true, value = "REST representation of the 'BIDTYPE\_BODY' (required) (Size = 1)") String bidtype;

private final @ApiModelProperty(required = true, value = "REST representation of the 'BIDSTARTTIME\_BODY' (required)") Timestamp bidstarttime;

private final @ApiModelProperty(required = true, value = "REST representation of the 'BIDCLOSETIME\_BODY' (required)") Timestamp bidclosetime;

private final @ApiModelProperty(required = true, value = "REST representation of the 'BIDSTATUS\_BODY' (required) (Size = 1)") String bidstatus;

private final @ApiModelProperty(required = true, value = "REST representation of the 'BIDSTARTDATE\_BODY' (required)") Timestamp bidstartdate;

private final @ApiModelProperty(required = true, value = "REST representation of the 'BIDENDDATE\_BODY' (required)") Timestamp bidenddate;

private final @ApiModelProperty(required = true, value = "REST representation of the 'BIDOPENDATE\_BODY' (required)") Timestamp bidopendate;

private final @ApiModelProperty(required = true, value = "REST representation of the 'CREATEDON\_BODY' (required)") Timestamp createdon;

@JsonCreator

public PopbidcreationBody(

@JsonProperty("popbidcreationid") Long popbidcreationid,

@JsonProperty("lotid") Long lotid,

@JsonProperty("platformid") String platformid,

@JsonProperty("bidtype") String bidtype,

@JsonProperty("bidstarttime") @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss") Timestamp bidstarttime,

@JsonProperty("bidclosetime") @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss") Timestamp bidclosetime,

@JsonProperty("bidstatus") String bidstatus,

@JsonProperty("bidstartdate") @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss") Timestamp bidstartdate,

@JsonProperty("bidenddate") @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss") Timestamp bidenddate,

@JsonProperty("bidopendate") @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss") Timestamp bidopendate,

@JsonProperty("createdon") @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss") Timestamp createdon) {

this.popbidcreationid = popbidcreationid;

this.lotid = Objects.requireNonNull(lotid, "`lotid` is required");

this.platformid = Objects.requireNonNull(platformid, "`platformid` is required");

this.bidtype = Objects.requireNonNull(bidtype, "`bidtype` is required");

this.bidstarttime = Objects.requireNonNull(bidstarttime, "`bidstarttime` is required");

this.bidclosetime = Objects.requireNonNull(bidclosetime, "`bidclosetime` is required");

this.bidstatus = Objects.requireNonNull(bidstatus, "`bidstatus` is required");

this.bidstartdate = Objects.requireNonNull(bidstartdate, "`bidstartdate` is required");

this.bidenddate = Objects.requireNonNull(bidenddate, "`bidenddate` is required");

this.bidopendate = Objects.requireNonNull(bidopendate, "`bidopendate` is required");

this.createdon = Objects.requireNonNull(createdon, "`createdon` is required");

}

public Long getPopbidcreationid() {

return popbidcreationid;

}

public Long getLotid() {

return lotid;

}

public String getPlatformid() {

return platformid;

}

public String getBidtype() {

return bidtype;

}

public Timestamp getBidstarttime() {

return bidstarttime;

}

public Timestamp getBidclosetime() {

return bidclosetime;

}

public String getBidstatus() {

return bidstatus;

}

public Timestamp getBidstartdate() {

return bidstartdate;

}

public Timestamp getBidenddate() {

return bidenddate;

}

public Timestamp getBidopendate() {

return bidopendate;

}

public Timestamp getCreatedon() {

return createdon;

}

}

}

package com.micropro.enam.pop.controller;

import java.sql.Connection;

import java.util.Map;

import javax.servlet.http.HttpServletRequest;

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

import org.springframework.http.HttpStatus;

import org.springframework.validation.annotation.Validated;

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

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

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

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

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

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

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

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

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

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

import com.fasterxml.jackson.core.JsonProcessingException;

import com.google.gson.JsonObject;

import com.micropro.custom.services.CustomService;

import com.micropro.enam.pop.common.ConnectionUtility;

import com.micropro.enam.pop.dao.BidCreationDao;

import com.micropro.enam.pop.model.generated.BidCreationModel.PopbidcreationBody;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import io.swagger.annotations.ApiResponse;

@RestController

@CrossOrigin(origins = "\*", maxAge = 3600)

@Api(description = ".", tags = { "Bid Creation APIs" })

@RequestMapping("/rest")

public class BidCreationController extends CustomService {

private @Autowired BidCreationDao bidCreationDao;

Map<String, String> errormsg; // <Field, Error\_msg>

HttpServletRequest hRequest;

private @Autowired ConnectionUtility utility;

@ResponseStatus(code = HttpStatus.OK)

@PostMapping(path = "/bid\_creation", consumes = "application/json")

@ApiOperation(value = "CRUD operation on Popbidcreation entity", httpMethod = "POST")

@ApiResponse(code = 200, message = "Returns a 200 response code if a Popbidcreation entity has been created successfully")

public String insert(@RequestBody @Validated PopbidcreationBody body, HttpServletRequest request) {

hRequest = request;

JsonObject response = new JsonObject();

try {

long id = bidCreationDao.crud(hRequest, body, "INSERT");

if (id > 0) {

response.addProperty("txnid", id);

response.addProperty("code", 200);

response.addProperty("status", "Success");

response.addProperty("message", "Saved successfully");

} else {

throw new CustomException("Failed to save data");

}

return response.toString();

} catch (JsonProcessingException e) {

throw new CustomException("Invalid Json body");

}

}

@ResponseStatus(code = HttpStatus.OK)

@PutMapping(path = "/bid\_creation", consumes = "application/json")

@ApiOperation(value = "CRUD operation on Popbidcreation entity", httpMethod = "PUT")

@ApiResponse(code = 200, message = "Returns a 200 response code if a Popbidcreation entity has been updated successfully")

public String update(@RequestBody @Validated PopbidcreationBody body, HttpServletRequest request) {

hRequest = request;

JsonObject response = new JsonObject();

try {

long id = bidCreationDao.crud(hRequest, body, "UPDATE");

if (id > 0) {

response.addProperty("txnid", id);

response.addProperty("code", 200);

response.addProperty("status", "Success");

response.addProperty("message", "Updated successfully");

} else {

throw new CustomException("Failed to update data");

}

return response.toString();

} catch (JsonProcessingException e) {

throw new CustomException("Invalid Json body");

}

}

@ResponseStatus(code = HttpStatus.OK)

@DeleteMapping(path = "/bid\_creation", consumes = "application/json")

@ApiOperation(value = "CRUD operation on Popbidcreation entity", httpMethod = "DELETE")

@ApiResponse(code = 200, message = "Returns a 200 response code if a Popbidcreation entity has been deleted successfully")

public String dalete(@RequestBody @Validated PopbidcreationBody body, HttpServletRequest request) {

hRequest = request;

JsonObject response = new JsonObject();

try {

long id = bidCreationDao.crud(hRequest, body, "DELETE");

if (id > 0) {

response.addProperty("txnid", id);

response.addProperty("code", 200);

response.addProperty("status", "Success");

response.addProperty("message", "Deleted successfully");

} else {

throw new CustomException("Failed to dalete data");

}

return response.toString();

} catch (JsonProcessingException e) {

throw new CustomException("Invalid Json body");

}

}

@ResponseStatus(code = HttpStatus.OK)

@GetMapping(path = "/bid\_creation/{POPBIDCREATIONID}", produces = "application/json")

@ApiOperation(value = "Retrieve a Popbidcreation entity with a specified 'POPBIDCREATIONID' column value", httpMethod = "GET")

@ApiResponse(code = 200, message = "Returns a 200 response code if a Popbidcreation entity has been retrieved successfully")

public String get(@PathVariable(name = "POPBIDCREATIONID") long popbidcreationid, HttpServletRequest request) {

return bidCreationDao.get(request, popbidcreationid);

}

@Override

public Connection getConnection() {

return utility.getConnection();

}

}

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.PostgreSQLDialect

spring.datasource.url = jdbc:postgresql://localhost:5432/demo

spring.datasource.username = postgres

spring.datasource.password = Acerfaraari@123

spring.jpa.hibernate.ddl-auto = update

spring.jpa.show-sql = true