/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package cta.service;

import cta.Report;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.List;

import javax.ejb.Stateless;

import javax.json.Json;

import javax.json.JsonArray;

import javax.json.JsonArrayBuilder;

import javax.json.JsonObject;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

import javax.persistence.Query;

import javax.persistence.TypedQuery;

import javax.ws.rs.Consumes;

import javax.ws.rs.DELETE;

import javax.ws.rs.GET;

import javax.ws.rs.POST;

import javax.ws.rs.PUT;

import javax.ws.rs.Path;

import javax.ws.rs.PathParam;

import javax.ws.rs.Produces;

import javax.ws.rs.core.MediaType;

/\*\*

\*

\* @author nafiy

\*/

@Stateless

@Path("cta.report")

public class ReportFacadeREST extends AbstractFacade<Report> {

@PersistenceContext(unitName = "CalorieTrackerApplicationPU")

private EntityManager em;

public ReportFacadeREST() {

super(Report.class);

}

@POST

@Override

@Consumes({MediaType.APPLICATION\_XML, MediaType.APPLICATION\_JSON})

public void create(Report entity) {

super.create(entity);

}

@PUT

@Path("{id}")

@Consumes({MediaType.APPLICATION\_XML, MediaType.APPLICATION\_JSON})

public void edit(@PathParam("id") Short id, Report entity) {

super.edit(entity);

}

@DELETE

@Path("{id}")

public void remove(@PathParam("id") Short id) {

super.remove(super.find(id));

}

@GET

@Path("{id}")

@Produces({MediaType.APPLICATION\_XML, MediaType.APPLICATION\_JSON})

public Report find(@PathParam("id") Short id) {

return super.find(id);

}

@GET

@Override

@Produces({MediaType.APPLICATION\_XML, MediaType.APPLICATION\_JSON})

public List<Report> findAll() {

return super.findAll();

}

@GET

@Path("{from}/{to}")

@Produces({MediaType.APPLICATION\_XML, MediaType.APPLICATION\_JSON})

public List<Report> findRange(@PathParam("from") Integer from, @PathParam("to") Integer to) {

return super.findRange(new int[]{from, to});

}

@GET

@Path("count")

@Produces(MediaType.TEXT\_PLAIN)

public String countREST() {

return String.valueOf(super.count());

}

@Override

protected EntityManager getEntityManager() {

return em;

}

@GET

@Path("findByRepUserid/{repUserid}")

@Produces({"application/json"})

public List<Report> findByRepUserid(@PathParam("repUserid") int repUserid) {

Query query = em.createNamedQuery("Report.findByRepUserid");

query.setParameter("repUserid", repUserid);

return query.getResultList();

}

@GET

@Path("findByRepDate/{repDate}")

@Produces({"application/json"})

public List<Report> findByRepDate(@PathParam("repDate") String repDate)throws Exception {

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

Date dt = sdf.parse(repDate);

Query query = em.createNamedQuery("Report.findByRepDate");

query.setParameter("repDate", dt);

return query.getResultList();

}

@GET

@Path("findByRepConsumecal/{repConsumecal}")

@Produces({"application/json"})

public List<Report> findByRepConsumecal(@PathParam("repConsumecal") Double repConsumecal) {

Query query = em.createNamedQuery("Report.findByRepConsumecal");

query.setParameter("repConsumecal", repConsumecal);

return query.getResultList();

}

@GET

@Path("findByRepBurnedcal/{repBurnedcal}")

@Produces({"application/json"})

public List<Report> findByRepBurnedcal(@PathParam("repBurnedcal") Double repBurnedcal) {

Query query = em.createNamedQuery("Report.findByRepBurnedcal");

query.setParameter("repBurnedcal", repBurnedcal);

return query.getResultList();

}

@GET

@Path("findByRepStep/{repStep}")

@Produces({"application/json"})

public List<Report> findByRepStep(@PathParam("repStep") Integer repStep) {

Query query = em.createNamedQuery("Report.findByRepStep");

query.setParameter("repStep", repStep);

return query.getResultList();

}

@GET

@Path("findByRepGoalcal/{repGoalcal}")

@Produces({"application/json"})

public List<Report> findByRepGoalcal(@PathParam("repGoalcal") Double repGoalcal) {

Query query = em.createNamedQuery("Report.findByRepGoalcal");

query.setParameter("repGoalcal", repGoalcal);

return query.getResultList();

}

@GET

@Path("findByRepUserIdAndRepDate/{repUserid}/{repDate}")

@Produces({"application/json"})

public Object findByRepUserIdAndRepDate(@PathParam("repUserid") int repUserid,@PathParam("repDate") String repDate)throws Exception {

JsonArrayBuilder arrayBuilder = Json.createArrayBuilder();

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

Date dt = sdf.parse(repDate);

TypedQuery<Object[]> query = em.createQuery("SELECT r.repConsumecal,r.repBurnedcal,r.repGoalcal FROM Report AS r WHERE r.repUserid.userId = :repUserid AND r.repDate = :repDate",Object[].class);

query.setParameter("repUserid", repUserid);

query.setParameter("repDate", dt);

List<Object[]> queryList = query.getResultList();

double remain = 0;

for (Object[] row : queryList) {

remain = (Double)row[2] - (Double)row[0] + (Double)row[1];

JsonObject userObject = Json.createObjectBuilder().add("total consumed calorie",(Double)row[0]).add("total burned calorie",(Double)row[1]).add("remain calorie",remain).build();

arrayBuilder.add(userObject);

}

JsonArray jArray = arrayBuilder.build();

return jArray;

}

@GET

@Path("findByRepUserIdAndStartTimeAndEndTime/{repUserid}/{repDate1}/{repDate2}")

@Produces({"application/json"})

public Object findByRepUserIdAndStartTimeAndEndTime(@PathParam("repUserid") int repUserid,@PathParam("repDate1") String repDate1,@PathParam("repDate2") String repDate2)throws Exception {

JsonArrayBuilder arrayBuilder = Json.createArrayBuilder();

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

Date dt1 = sdf.parse(repDate1);

Date dt2 = sdf.parse(repDate2);

TypedQuery<Object[]> query = em.createQuery("SELECT r.repConsumecal,r.repBurnedcal,r.repStep FROM Report AS r WHERE r.repUserid.userId = :repUserid AND (r.repDate BETWEEN :repDate1 AND :repDate2)",Object[].class);

query.setParameter("repUserid", repUserid);

query.setParameter("repDate1", dt1);

query.setParameter("repDate2", dt2);

List<Object[]> queryList = query.getResultList();

double tburnedcalorie = 0;

double tconsumedcalorie = 0;

int tstep = 0;

for (Object[] row : queryList) {

tconsumedcalorie = tconsumedcalorie + (Double)row[0];

tburnedcalorie = tburnedcalorie + (Double)row[1];

tstep = tstep + (Integer)row[2];

}

JsonObject userObject = Json.createObjectBuilder().add("total consumed calorie",tconsumedcalorie).add("total burned calorie",tburnedcalorie).add("total step",tstep).build();

arrayBuilder.add(userObject);

JsonArray jArray = arrayBuilder.build();

return jArray;

}

@GET

@Path("findCaloriesByRepUserIdAndStartTimeAndEndTime/{repUserid}/{repDate1}/{repDate2}")

@Produces({"application/json"})

public Object findCaloriesByRepUserIdAndStartTimeAndEndTime(@PathParam("repUserid") int repUserid,@PathParam("repDate1") String repDate1,@PathParam("repDate2") String repDate2)throws Exception {

JsonArrayBuilder arrayBuilder = Json.createArrayBuilder();

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

Date dt1 = sdf.parse(repDate1);

Date dt2 = sdf.parse(repDate2);

TypedQuery<Object[]> query = em.createQuery("SELECT r.repConsumecal,r.repBurnedcal,r.repDate FROM Report AS r WHERE r.repUserid.userId = :repUserid AND (r.repDate BETWEEN :repDate1 AND :repDate2)",Object[].class);

query.setParameter("repUserid", repUserid);

query.setParameter("repDate1", dt1);

query.setParameter("repDate2", dt2);

List<Object[]> queryList = query.getResultList();

for (Object[] row : queryList) {

JsonObject userObject = Json.createObjectBuilder().add("consumed calorie",(Double)row[0]).add("burned calorie",(Double)row[1]).add("date",sdf.format((Date)row[2])).build();

arrayBuilder.add(userObject);

}

JsonArray jArray = arrayBuilder.build();

return jArray;

}

}