/\*

\* 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.Users;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.Date;

import java.util.List;

import javax.ejb.Stateless;

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.users")

public class UsersFacadeREST extends AbstractFacade<Users> {

@PersistenceContext(unitName = "CalorieTrackerApplicationPU")

private EntityManager em;

public UsersFacadeREST() {

super(Users.class);

}

@POST

@Override

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

public void create(Users entity) {

super.create(entity);

}

@PUT

@Path("{id}")

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

public void edit(@PathParam("id") Short id, Users 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 Users find(@PathParam("id") Short id) {

return super.find(id);

}

@GET

@Override

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

public List<Users> findAll() {

return super.findAll();

}

@GET

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

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

public List<Users> 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("findByUserName/{userName}")

@Produces({"application/json"})

public List<Users> findByUserName(@PathParam("userName") String userName) {

Query query = em.createNamedQuery("Users.findByUserName");

query.setParameter("userName", userName);

return query.getResultList();

}

@GET

@Path("findByUserSurname/{userSurname}")

@Produces({"application/json"})

public List<Users> findByUserSurname(@PathParam("userSurname") String userSurname) {

Query query = em.createNamedQuery("Users.findByUserSurname");

query.setParameter("userSurname", userSurname);

return query.getResultList();

}

@GET

@Path("findByUserEmail/{userEmail}")

@Produces({"application/json"})

public List<Users> findByUserEmail(@PathParam("userEmail") String userEmail) {

Query query = em.createNamedQuery("Users.findByUserEmail");

query.setParameter("userEmail", userEmail);

return query.getResultList();

}

@GET

@Path("findByUserDob/{userDob}")

@Produces({"application/json"})

public List<Users> findByUserDob(@PathParam("userDob") String userDob)throws Exception {

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

Date dob = sdf.parse(userDob);

Query query = em.createNamedQuery("Users.findByUserDob");

query.setParameter("userDob", dob);

return query.getResultList();

}

@GET

@Path("findByUserHeight/{userHeight}")

@Produces({"application/json"})

public List<Users> findByUserHeight(@PathParam("userHeight") Double userHeight) {

Query query = em.createNamedQuery("Users.findByUserHeight");

query.setParameter("userHeight", userHeight);

return query.getResultList();

}

@GET

@Path("findByUserWeight/{userWeight}")

@Produces({"application/json"})

public List<Users> findByUserWeight(@PathParam("userWeight") Double userWeight) {

Query query = em.createNamedQuery("Users.findByUserWeight");

query.setParameter("userWeight", userWeight);

return query.getResultList();

}

@GET

@Path("findByUserGender/{userGender}")

@Produces({"application/json"})

public List<Users> findByUserGender(@PathParam("userGender") String userGender) {

Query query = em.createNamedQuery("Users.findByUserGender");

query.setParameter("userGender", userGender);

return query.getResultList();

}

@GET

@Path("findByUserAddress/{userAddress}")

@Produces({"application/json"})

public List<Users> findByUserAddress(@PathParam("userAddress") String userAddress) {

Query query = em.createNamedQuery("Users.findByUserAddress");

query.setParameter("userAddress", userAddress);

return query.getResultList();

}

@GET

@Path("findByUserPostcode/{userPostcode}")

@Produces({"application/json"})

public List<Users> findByUserPostcode(@PathParam("userPostcode") String userPostcode) {

Query query = em.createNamedQuery("Users.findByUserPostcode");

query.setParameter("userPostcode", userPostcode);

return query.getResultList();

}

@GET

@Path("findByUserActivelevel/{userActivelevel}")

@Produces({"application/json"})

public List<Users> findByUserActivelevel(@PathParam("userActivelevel") Integer userActivelevel) {

Query query = em.createNamedQuery("Users.findByUserActivelevel");

query.setParameter("userActivelevel", userActivelevel);

return query.getResultList();

}

@GET

@Path("findByUserSteppermile/{userSteppermile}")

@Produces({"application/json"})

public List<Users> findByUserSteppermile(@PathParam("userSteppermile") Integer userSteppermile) {

Query query = em.createNamedQuery("Users.findByUserSteppermile");

query.setParameter("userSteppermile", userSteppermile);

return query.getResultList();

}

@GET

@Path("findByUserNameAndUserSurname/{userName}/{userSurname}")

@Produces({"application/json"})

public List<Users> findByUserNameAndUserSurname(@PathParam("userName") String userName,@PathParam("userSurname") String userSurname) {

TypedQuery<Users> query = em.createQuery("SELECT u FROM Users u WHERE u.userName = :userName AND u.userSurname = :userSurname", Users.class);

query.setParameter("userName", userName);

query.setParameter("userSurname", userSurname);

return query.getResultList();

}

@GET

@Path("findCalorieBurnedPerStep/{userId}")

@Produces("text/plain")

public double findCalorieBurnedPerStep(@PathParam("userId") Short userId) {

TypedQuery<Object[]> query = em.createQuery("SELECT u.userSteppermile,u.userWeight FROM Users AS u WHERE u.userId = :userId",Object[].class);

query.setParameter("userId", userId);

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

double calorie = 0.0;

for (Object[] row : queryList) {

calorie = (Double) row[1]/((double)(Short)row[0]\*1.6);

}

return calorie;

}

@GET

@Path("findBMR/{userId}")

@Produces("text/plain")

public double findBMR(@PathParam("userId") Short userId) throws Exception {

TypedQuery<Object[]> query = em.createQuery("SELECT u.userWeight,u.userHeight,u.userDob,u.userGender FROM Users AS u WHERE u.userId = :userId",Object[].class);

query.setParameter("userId", userId);

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

double bmr = 0.0;

for (Object[] row : queryList) {

Calendar now = Calendar.getInstance();

Date d = (Date)row[2];

Calendar dob = Calendar.getInstance();

dob.setTime(d);

int age = dob.get(Calendar.YEAR) - now.get(Calendar.YEAR);

if (((String)row[3]).equals("m"))

{

bmr = 13.75 \* (Double)row[0] + 5.003 \* (Double)row[1] - 6.755 \* age + 66.5;

}

else if (((String)row[3]).equals("f"))

{

bmr = 9.563 \* (Double)row[0] + 1.85 \* (Double)row[1] - 4.676 \* age + 655.1 ;

}

}

return bmr;

}

@GET

@Path("findTotalCalorieBurned/{userId}")

@Produces("text/plain")

public double findTotalCalorieBurned(@PathParam("userId") Short userId)throws Exception {

TypedQuery<Short> query = em.createQuery("SELECT u.userActivelevel FROM Users AS u WHERE u.userId = :userId",Short.class);

query.setParameter("userId", userId);

List<Short> queryList = query.getResultList();

double bmr = findBMR(userId);

double total = 0.0;

int act = 1;

for (Short row : queryList)

{

act = row;

switch(act)

{

case 1:total = 1.2 \* bmr;

break;

case 2:total = 1.375 \* bmr;

break;

case 3:total = 1.55 \* bmr;

break;

case 4:total = 1.725 \* bmr;

break;

case 5:total = 1.9 \* bmr;

break;

}

}

return total;

}

}