Atitit jpa doc rest springboot

javaprjx jpa rest.zip

C:\Users\ati\OneDrive\sumdoc atub\javaprjx jpa rest.zip

**package** comx;

**import** java.util.List;

**import** java.util.Map;

**import** javax.persistence.EntityManager;

**import** javax.persistence.EntityManagerFactory;

**import** javax.persistence.Persistence;

**import** javax.persistence.Query;

**import** org.apache.openjpa.enhance.RuntimeUnenhancedClassesModes;

**import** com.alibaba.fastjson.JSON;

**import** com.google.common.collect.Maps;

**import** com.mysql.cj.core.conf.url.ConnectionUrlParser;

**public** **class** JpaTest {

**public** **static** **void** main(String[] args) {

Map properties = Maps.*newLinkedHashMap*();

properties.put("javax.persistence.jdbc.driver", "com.mysql.cj.jdbc.Driver");

properties.put("javax.persistence.jdbc.url",

"jdbc:mysql://localhost:3306/test?user=root&password=&allowMultiQueries=true");

properties.put("javax.persistence.jdbc.user", "root"); // if needed

properties.put("javax.persistence.jdbc.password", ""); // if needed

properties.put("openjpa.RuntimeUnenhancedClasses", RuntimeUnenhancedClassesModes.***SUPPORTED***); // if needed

EntityManagerFactory factory = Persistence.*createEntityManagerFactory*("wmsPersisteUnitName", properties);

EntityManager em = factory.createEntityManager();

Query q = em.createNativeQuery("select 1 as age,'ati' name;select 2;", Map.**class**);

List<Map> li = q.getResultList();

System.***out***.println(JSON.*toJSONString*(li));

**for** (Object object : li) {

System.***out***.println(object);

}

}

C:\Users\Administrator\eclipse-workspace\javaprjx

package db;

import java.util.List;

import java.util.Map;

//frm hibernate-jpa-2.1-api

import javax.persistence.EntityManager;

import javax.persistence.EntityManagerFactory;

import javax.persistence.EntityTransaction;

import javax.persistence.Persistence;

import javax.persistence.Query;

import org.hibernate.SQLQuery;

import org.hibernate.transform.Transformers;

import com.alibaba.fastjson.JSONObject;

import com.google.common.collect.Maps;

public class JpaSaveUpdate {

public static void main(String[] args) {

Map properties = Maps.newLinkedHashMap();

properties.put("javax.persistence.jdbc.driver", "org.sqlite.JDBC");

properties.put("javax.persistence.jdbc.url","jdbc:sqlite:test"+Math.random()+".db");

// properties.put("javax.persistence.jdbc.user", usr.toString()); // if needed

// properties.put("javax.persistence.jdbc.password", pwd.toString()); // if needed

// properties.put("openjpa.RuntimeUnenhancedClasses", RuntimeUnenhancedClassesModes.SUPPORTED); // if needed

// // properties.put("openjpa.RuntimeUnenhancedClasses", "supported"); // if needed

// org.hibernate.dialect.SQLiteDialect

System.out.println(properties);

// Create a new EntityManagerFactory using the System properties.

// The "hellojpa" name will be used to configure based on the

// corresponding name in the META-INF/persistence.xml file

//from hibernate-jpa-2.1-api jar

EntityManagerFactory factory = Persistence.createEntityManagerFactory("HbntUnit",properties);

exeUpdate(factory, "CREATE TABLE sys\_data (jsonfld json )");

exeUpdate(factory, "insert into sys\_data values('{\"age\":88}')");

System.out.println(query(factory,"select json\_extract(jsonfld,'$.age') as age from sys\_data") ); ;

System.out.println("f");

}

private static List<Map> query(EntityManagerFactory factory, String sql) {

EntityManager em = factory.createEntityManager();

Query createNativeQuery = em.createNativeQuery(sql ,Map.class );

// createNativeQuery.unwrap(SQLQuery.class).setResultTransformer(Transformers.ALIAS\_TO\_ENTITY\_MAP);

List<Map> result = createNativeQuery.getResultList();

return result;

}

private static int exeUpdate(EntityManagerFactory factory, String sql) {

try {

EntityManager em = factory.createEntityManager();

EntityTransaction transaction = em.getTransaction();

transaction.begin();

/////sql

// sql = MessageFormat.format(sql, "'" + getUpflag() + "'", "'" + getUpflag() + "'", "'" + getUpflag() + "'");

System.out.println(sql);

Query createNativeQuery = em.createNativeQuery(sql);

int executeUpdate = createNativeQuery.executeUpdate();

System.out.println(executeUpdate);

transaction.commit();

return executeUpdate;

} catch (Exception e) {

e.printStackTrace();

}

return 0;

}

}