Atitit jpa 读写范例

目录

[1. Pom maven 1](#_Toc30401)

[1.1. JpaOpenjpa 1](#_Toc13195)

[1.2. Other 2](#_Toc23190)

[1.3. Custome jpa cfg 4](#_Toc16618)

[2. JpaSave 5](#_Toc24706)

# Pom maven

<!-- https://mvnrepository.com/artifact/org.apache.openjpa/openjpa -->

<dependency>

<groupId>org.apache.openjpa</groupId>

<artifactId>openjpa</artifactId>

<version>3.1.1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/javax.persistence/javax.persistence-api -->

<dependency>

<groupId>javax.persistence</groupId>

<artifactId>javax.persistence-api</artifactId>

<version>2.2</version>

</dependency>

org.eclipse.persistence.jpa.PersistenceProvider

## JpaOpenjpa

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

**public** **class** JpaOpenjpa {

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

// 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

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

// Create a new EntityManager from the EntityManagerFactory. The

// EntityManager is the main object in the persistence API, and is

// used to create, delete, and query objects, as well as access

// the current transaction

EntityManager em = factory.createEntityManager();

// Perform a simple query for all the Message entities

Query q = em.createNativeQuery("select \* from sys\_tab",Map.**class**);

List li = q.getResultList();

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

}

}

## Other

Hb的和openjpa的使用api逗死jpa一样的

Spring data api失败

package com.kok.sport.utils.db;

import java.util.List;

import java.util.Map;

import org.junit.Test;

import org.junit.runner.RunWith;

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

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.core.type.AnnotationMetadata;

import org.springframework.data.repository.config.AnnotationRepositoryConfigurationSource;

import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)

@SpringBootTest

public class JpaTest {

@Autowired

JpaDao JpaDao1;

@Test

public void testAdd() {

// AnnotationMetadata

List<Map> li= JpaDao1.findSQL("1", "2");

System.out.println(li);

System.out.println("F");

}

public static void main(String[] args) {

//AnnotationRepositoryConfigurationSource

}

}

package com.kok.sport.utils.db;

import java.util.List;

import java.util.Map;

import java.util.Optional;

import org.springframework.data.domain.Example;

import org.springframework.data.domain.Page;

import org.springframework.data.domain.Pageable;

import org.springframework.data.domain.Sort;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.query.Param;

@Repository

public interface JpaDao extends JpaRepository<Map , Long>{

//@Query(nativeQuery = true, value = "SELECT \* FROM AUTH\_USER WHERE name = :name1 OR name = :name2 ")

@Query(nativeQuery = true, value = "SELECT 333 ")

List<Map> findSQL(@Param("name1") String name1, @Param("name2") String name2);

}

## Custome jpa cfg

**private** **static** EntityManagerFactory JpaUtil\_getFac() {

String url2;// =

url2=MybatisUtil. *getMysqlUrl*();

System.***out***.println(url2);

String mysqlConnUrl =url2;

ConnectionUrlParser connStringParser = ConnectionUrlParser.*parseConnectionString*(mysqlConnUrl);

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

Object url = mysqlConnUrl;

Object usr = UrlUtil.*parse4Q*(connStringParser.getQuery()).get("user");

Object pwd = UrlUtil.*parse4Q*(connStringParser.getQuery()).get("password");

Map properties = Maps.*newLinkedHashMap*();

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

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

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

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

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

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

**return** factory;

}

# JpaSave

ckage com.kok.sport.utils.db;

import javax.persistence.EntityManager;

import javax.persistence.EntityManagerFactory;

import javax.persistence.EntityTransaction;

public class JpaSave {

public static void main(String[] args) {

JpaEntityAti d = new JpaEntityAti();

d.setName("Angel");

EntityManagerFactory factory = JpaUtil.getFac();

EntityManager em = factory.createEntityManager();

EntityTransaction transaction = em.getTransaction();

transaction.begin();

em.persist(d);

transaction.commit();

System.out.println("f");

}

}