<http://blog.csdn.net/u012702547/article/details/53946440>

版权声明：本文为sang原创文章，转载请注明出处。

目录[(?)[+]](http://blog.csdn.net/u012702547/article/details/53946440)

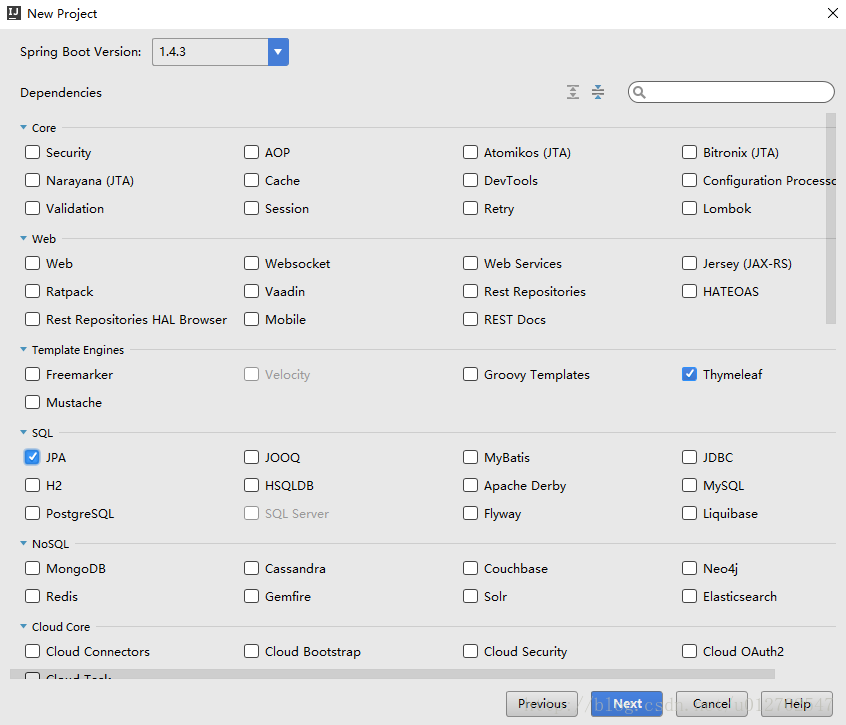
前面关于[**spring**](http://lib.csdn.net/base/javaee) Boot的文章已经介绍了很多了，但是一直都没有涉及到[**数据库**](http://lib.csdn.net/base/mysql)的操作问题，数据库操作当然也是我们在开发中无法回避的问题，那么今天我们就来看看Spring Boot给我们提供了哪些疯狂的方式来解决数据库的操作问题。   
如果小伙伴想要读懂这篇博客，必须要对Spring Boot有一定了解，如果你还不太了解，可以先移步这里[从Spring到Spring Boot](https://github.com/lenve/JavaEETest)，如果已经很了解，那么请忽略。   
OK，废话不多说，让我们愉快的开启今天的数据库操作之旅吧！

# 什么是JPA

一说JavaWeb，很多小伙伴都知道SSH，这个H代表的就是[**hibernate**](http://lib.csdn.net/base/javaee)框架，这个小伙伴们都知道，可是什么又是JPA呢？相信许多刚入门的小伙伴听说过但不是特别清楚，首先JPA的全称叫做[**Java**](http://lib.csdn.net/base/java)Persistence API，JPA是一个基于O/R映射的标准规范，在这个规范中，JPA只定义标准规则，不提供实现，使用者则需要按照规范中定义的方式来使用。目前JPA的主要实现有Hibernate、EclipseLink、OpenJPA等，事实上，由于Hibernate在数据访问解决技术领域的绝对霸主地位，JPA的标准基本是由Hibernate来主导的。虽然做开发的小伙伴不怎么喜欢度娘，不过度娘关于JPA的介绍个人觉得倒是比较清晰，有兴趣的小伙伴可前去了解下。[JPA\_百度百科](http://baike.baidu.com/link?url=ikjN5lL4zdDWDZ89W8e6KOIksEc0Je-xS0yqp2Ap2QO1XeO9AuogE5pATveyVNLK56mBhMWujPMtD1je6F44XK)。另外，Spring框架为我们提供了Spring Data JPA这样一个东东，可以减少我们使用JPA时的代码量。

# 使用流程

## 创建工程并添加相关依赖

在Spring Boot中使用JPA，我们在创建工程的时候需要选择JPA依赖，如下：   
   
其他的步骤和我们创建一个普通的Spring Boot项目是一样的，如果小伙伴不了解如何创建一个Spring Boot项目可以参考这篇文章[初识Spring Boot框架](http://blog.csdn.net/u012702547/article/details/53740047)。   
项目创建成功之后，我这里是使用[**MySQL**](http://lib.csdn.net/base/mysql)做演示，因此还需要添加[**mysql**](http://lib.csdn.net/base/mysql)驱动，在pom.xml文件中添加如下依赖：

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.40</version>

</dependency>

## 配置基本属性

接下来需要我们在application.properties中配置数据源和jpa的基本的相关属性，如下：

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/jpatest

spring.datasource.username=root

spring.datasource.password=123456

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jackson.serialization.indent\_output=true

关于这里的配置我说如下几点：

1.第一行表示驱动的名称，这个和具体的数据库驱动有关，视情况而定，我这里使用了MySql数据库，所以驱动名为com.mysql.jdbc.Driver   
2.第二行表示数据库连接地址，当然也是视情况而定   
3.第三四行表示数据库连接的用户名和密码   
4.第五行则配置了实体类维护数据库表结构的具体行为，update表示当实体类的属性发生变化时，表结构跟着更新，这里我们也可以取值create，这个create表示启动的时候删除上一次生成的表，并根据实体类重新生成表，这个时候之前表中的数据就会被清空；还可以取值create-drop，这个表示启动时根据实体类生成表，但是当sessionFactory关闭的时候表会被删除；validate表示启动时验证实体类和数据表是否一致；none表示啥都不做。   
5.第六行表示hibernate在操作的时候在控制台打印真实的sql语句   
6.第七行表示格式化输出的json字符串

OK，以上就是我们在application.properties中对JPA进行的一个简单配置。

## 定义映射实体类

接下来，定义相应的实体类，在Project启动时，系统会根据实体类创建相应的数据表，我的实体类如下：

@Entity

@NamedQuery(name = "Person.withNameAndAddressNamedQuery",

query = "select p from Person p where p.name=?1 and p.address=?2")

public class Person {

@Id

@GeneratedValue

private Long id;

private String name;

private Integer age;

private String address;

public Person() {

}

public Person(Long id, String name, Integer age, String address) {

this.id = id;

this.name = name;

this.age = age;

this.address = address;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Integer getAge() {

return age;

}

public void setAge(Integer age) {

this.age = age;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

}

首先在实体类上我们使用了@Entity注解，这个表示这是一个和数据库表映射的实体类，在属性id上我们添加了@Id注解，表示该字段是一个id，@GeneratedValue注解则表示该字段自增。@NamedQuery注解表示一个NamedQuery查询，这里一个名称代表一个查询语句，我们一会可以在控制器中直接调用@NamedQuery中的withNameAndAddressNamedQuery方法，该方法代表的查询语句是select p from Person p where p.name=?1 and p.address=?2。

## 定义数据访问接口

OK，做好上面几个步骤之后，接下来我们就可以定义数据访问接口了，我们的数据访问接口需要继承JpaRepository类，我在数据访问接口中一共定义了四个方法，如下：

public interface PersonRepository extends JpaRepository<Person, Long> {

List<Person> findByAddress(String name);

Person findByNameAndAddress(String name, String address);

@Query("select p from Person p where p.name=:name and p.address=:address")

Person withNameAndAddressQuery(@Param("name") String name, @Param("address") String address);

Person withNameAndAddressNamedQuery(String name, String address);

}

关于这个数据访问接口，我说如下几点：

1.当我们继承JpaRepository接口后，我们就自动具备了如下数据访问方法：

List<T> findAll();

List<T> findAll(Sort var1);

List<T> findAll(Iterable<ID> var1);

<S extends T> List<S> save(Iterable<S> var1);

void flush();

<S extends T> S saveAndFlush(S var1);

void deleteInBatch(Iterable<T> var1);

void deleteAllInBatch();

T getOne(ID var1);

<S extends T> List<S> findAll(Example<S> var1);

<S extends T> List<S> findAll(Example<S> var1, Sort var2);

2.我们可以在接口中定义查询方法，可以按照属性名来查询，但是方法的命名方式是固定的，比如第一个方法和第二个方法，第一个方法表示根据一个属性查询，第二个方法表示根据多个属性查询，findBy、And等可以算作是这里的查询关键字了，如果写作其他名称则系统不能识别，类似的关键字还有Like、Or、Is、Equals、Between等，而这里的findBy关键字又可以被find、read、readBy、query、queryBy、get、getBy等来代替。

3.在查询的过程中我们也可以限制查询结果，这里使用的关键字是top、first等，比如查询前10条数据我们可以写作：

List<Person> findFirst10ByName(String name);

4.使用NamedQuery来查询，就是我们直接在实体类上使用@NamedQuery注解来定义查询方法和方法名，一个名称对应一个查询语句，具体可以参考我们上文的实体类   
5.我们也可以向第三个方法那样添加@Query注解，当我调用这个方法的时候使用这个注解中的sql语句进行查询，方法的参数则是注解中的占位符的值。

## 编写测试Controller

数据访问接口都有了，接下来就是一个Controller了，我们写一个简单的Controller，用来[**测试**](http://lib.csdn.net/base/softwaretest)一下上文中的数据访问接口是否正确，如下：

@RestController

public class DataController {

@Autowired

PersonRepository personRepository;

@RequestMapping("/save")

public Person save(String name,String address,Integer age) {

Person person = personRepository.save(new Person(null, name, age, address));

return person;

}

@RequestMapping("/q1")

public List<Person> q1(String address) {

List<Person> people = personRepository.findByAddress(address);

return people;

}

@RequestMapping("/q2")

public Person q2(String name, String address) {

Person people = personRepository.findByNameAndAddress(name, address);

return people;

}

@RequestMapping("/q3")

public Person q3(String name, String address) {

Person person = personRepository.withNameAndAddressQuery(name, address);

return person;

}

@RequestMapping("/q4")

public Person q4(String name, String address) {

Person person = personRepository.withNameAndAddressNamedQuery(name, address);

return person;

}

@RequestMapping("/sort")

public List<Person> sort() {

List<Person> people = personRepository.findAll(new Sort(Sort.Direction.ASC, "age"));

return people;

}

@RequestMapping("/page")

public Page<Person> page(int page,int size){

Page<Person> all = personRepository.findAll(new PageRequest(page, size));

return all;

}

@RequestMapping("/all")

public List<Person> all(){

return personRepository.findAll();

}

}

* 3这里的代码都很简单，我就不再一一进行解释了，值得说的是第36行代码表示根据age对查询结果进行排序然后显示出来，第40行的方法表示一个分页查询，第一个参数表示页数，从0开始计，第二个参数表示每页的数据量。最后在浏览器中分别测试这几个接口就可以了，我这里就不再展示测试页面了，小伙伴们自行测试。

本文源码下载：   
[本文GitHub地址](https://github.com/lenve/JavaEETest/tree/master/Test22-JPA)

更多关于Spring Boot资料请移步[这里](https://github.com/lenve/JavaEETest)

参考资料：   
《JavaEE开发的颠覆者 Spring Boot实战》第八章