链滴

SpringBoot 入门教程 (八) | 使用 Spring Data JPA 访问 Mysql 数据库

作者：JavaFish

原文链接：https://ld246.com/article/1621247208558

来源网站：链滴

前言

如题，今天介绍 Spring Data JPA 的使用。

什么是 Spring Data JPA

在介绍 Spring Data JPA 之前，首先介绍 Hibernate 。 Hibernate 使用 O/R 映射 （Object-Relatio Mapping） 技术实现数据访问， O/R 映射即将领域模型类与数据库的表进行映射，通过程序操作象而实现表数据操作的能力，让数据访问操作无需关注数据库相关技术。

Hibernate 主导了 EJB 3.0 的 JPA 规范， JPA 即 Java Persistence API。JPA 是一个基于 O/R 映射标准协议（目前最新版本是 JPA 2.1）。所谓规范即只定义标准规制（如注解、接口），不提供实现软件提供商可以按照标准规范来实现，而使用者只需按照规范中定义的方式来使用，而不用和软件提商的实现打交道。

JPA 的主要实现由 Hibernate 、 EclipseLink 和 OpenJPA 等完成，我们只要使用 JPA 来开发，无论哪一个开发方式都是一样的。

Spring Data JPA 是 Spring Data 的一个子项目，它通过基于 JPA 的 Repository 极大地减少了 JPA 作为数据访问方案的代码量。

简而言之，JPA 是一种 ORM 规范，但并未提供 ORM 实现，而 Hibernate 是一个 ORM 框架，它供了 ORM 实现。

准备工作

- IDEA
- JDK1.8
- SpringBoot 2.1.3

pom.xml 文件引入的依赖如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
MLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd
maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.1.3.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.nasus</groupId>
    <artifactId>jpa</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>jpa</name>
    <description>jpa Demo project for Spring Boot</description>

    <properties>
```

```
        <java.version>1.8</java.version>
    </properties>

    <dependencies>

        <!-- JPA 依赖 -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <!-- web 依赖 -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <!-- mysql 连接类 -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
        <!-- lombok 依赖 -->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <!-- 单元测试依赖 -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

简单说下，加入 JPA 依赖；mysql 连接类用于连接数据；web 启动类，但凡是 web 应用都需要依赖；lombok 用于简化实体类。不会的看这篇旧文介绍：SpringBoot 实战 (三) | 使用 LomBok

application.yaml 配置文件

```
spring:
# 数据库相关
```

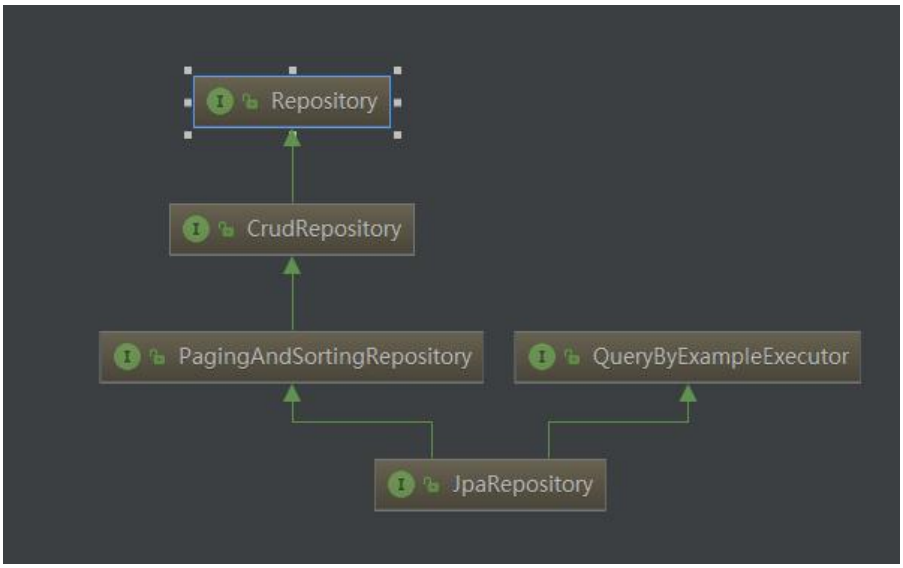
```
  datasource:
    driver-class-name: com.mysql.jdbc.Driver
    url: jdbc:mysql://127.0.0.1:3306/test?useUnicode=true&characterEncoding=utf8&serverTi
ezone=UTC&useSSL=true
    username: root
    password: 123456
# JPA 相关
  jpa:
    hibernate:
      ddl-auto: update   #ddl-auto:设为 create 表示每次都重新建表
    show-sql: true
```

repository (dao) 层

```
package com.nasus.jpa.repository;

import com.nasus.jpa.entity.Student;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

/**
 * Project Name:springboot_jpa_demo <br/>
 * Package Name:com.nasus.jpa.repository <br/>
 * Date:2019/2/19 21:37 <br/>
 * <b>Description:</b> TODO: 描述该类的作用 <br/>
 * @author <a href="turodog@foxmail.com">nasus</a><br/>
 */
@Repository
public interface StudentRepository extends JpaRepository<Student,Integer>, CrudRepository
Student, Integer> {
}
```



从上图，可以看出 JpaRepository 继承于 PangingAndSortingRepository 继承于 CrudRepository 。

CrudRepository 提供基本的增删改查PagingAndSortingRepository 提供分页和排序方法；JpaRep

sitory 提供 JPA 需要的方法。在使用的时候，可以根据具体需要选中继承哪个接口。

使用这些接口的好处有：

1. 继承这些接口，可以使Spring找到自定义的数据库操作接口，并生成代理类，后续可以注入到Sprin容器中；

2. 可以不写相关的sql操作，由代理类生成

service 层

```
package com.nasus.jpa.service;

import com.nasus.jpa.entity.Student;
import java.util.List;

/**
 * Project Name:springboot_jpa_demo <br/>
 * Package Name:com.nasus.jpa.service <br/>
 * Date:2019/2/19 21:41 <br/>
 * <b>Description:</b> TODO: 描述该类的作用 <br/>
 * @author <a href="turodog@foxmail.com">nasus</a><br/>
 */
public interface StudentService {

    Student save(Student student);

    Student findStudentById(Integer id);

    void delete(Integer id);

    void updateStudent(Student student);

    List<Student> findStudentList();
}
```

实现类：

```
package com.nasus.jpa.service.impl;

import com.nasus.jpa.entity.Student;
import com.nasus.jpa.repository.StudentRepository;
import com.nasus.jpa.service.StudentService;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

/**
 * Project Name:springboot_jpa_demo <br/>
 * Package Name:com.nasus.jpa.service.impl <br/>
 * Date:2019/2/19 21:43 <br/>
 * <b>Description:</b> TODO: 描述该类的作用 <br/>
 * @author <a href="turodog@foxmail.com">nasus</a><br/>
 */
```

```java
@Service
public class StudentServiceImpl implements StudentService {

    @Autowired
    private StudentRepository studentRepository;

    /**
     * 保存学生信息
     * @param student
     * @return
     */
    @Override
    public Student save(Student student) {
        return studentRepository.save(student);
    }

    /**
     * 根据 Id 查询学生信息
     * @param id
     * @return
     */
    @Override
    public Student findStudentById(Integer id) {
        return studentRepository.findById(id).get();
    }

    /**
     * 删除学生信息
     * @param id
     */
    @Override
    public void delete(Integer id) {
        Student student = this.findStudentById(id);
        studentRepository.delete(student);
    }

    /**
     * 更新学生信息
     * @param student
     */
    @Override
    public void updateStudent(Student student) {
        studentRepository.save(student);
    }

    /**
     * 查询学生信息列表
     * @return
     */
    @Override
    public List<Student> findStudentList() {
        return studentRepository.findAll();
    }
}
```

controller 层构建 restful API

```
package com.nasus.jpa.controller;

import com.nasus.jpa.entity.Student;
import com.nasus.jpa.service.StudentService;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * Project Name:springboot_jpa_demo <br/>
 * Package Name:com.nasus.jpa.controller <br/>
 * Date:2019/2/19 21:55 <br/>
 * <b>Description:</b> TODO: 描述该类的作用 <br/>
 * @author <a href="turodog@foxmail.com">nasus</a><br/>
 */
@RestController
@RequestMapping("/student")
public class StudentController {

    @Autowired
    private StudentService studentService;

    @PostMapping("/save")
    public Student saveStudent(@RequestBody Student student){
        return studentService.save(student);
    }

    @GetMapping("/{id}")
    public Student findStudentById(@PathVariable("id") Integer id){
        return studentService.findStudentById(id);
    }

    @GetMapping("/list")
    public List<Student> findStudentList(){
        return studentService.findStudentList();
    }

    @DeleteMapping("/{id}")
    public void deleteStudentById(@PathVariable("id") Integer id){
        studentService.delete(id);
    }

    @PutMapping("/update")
    public void updateStudent(@RequestBody Student student){
        studentService.updateStudent(student);
```

```
    }
}
```

测试结果

其他接口已通过 postman 测试，无问题。

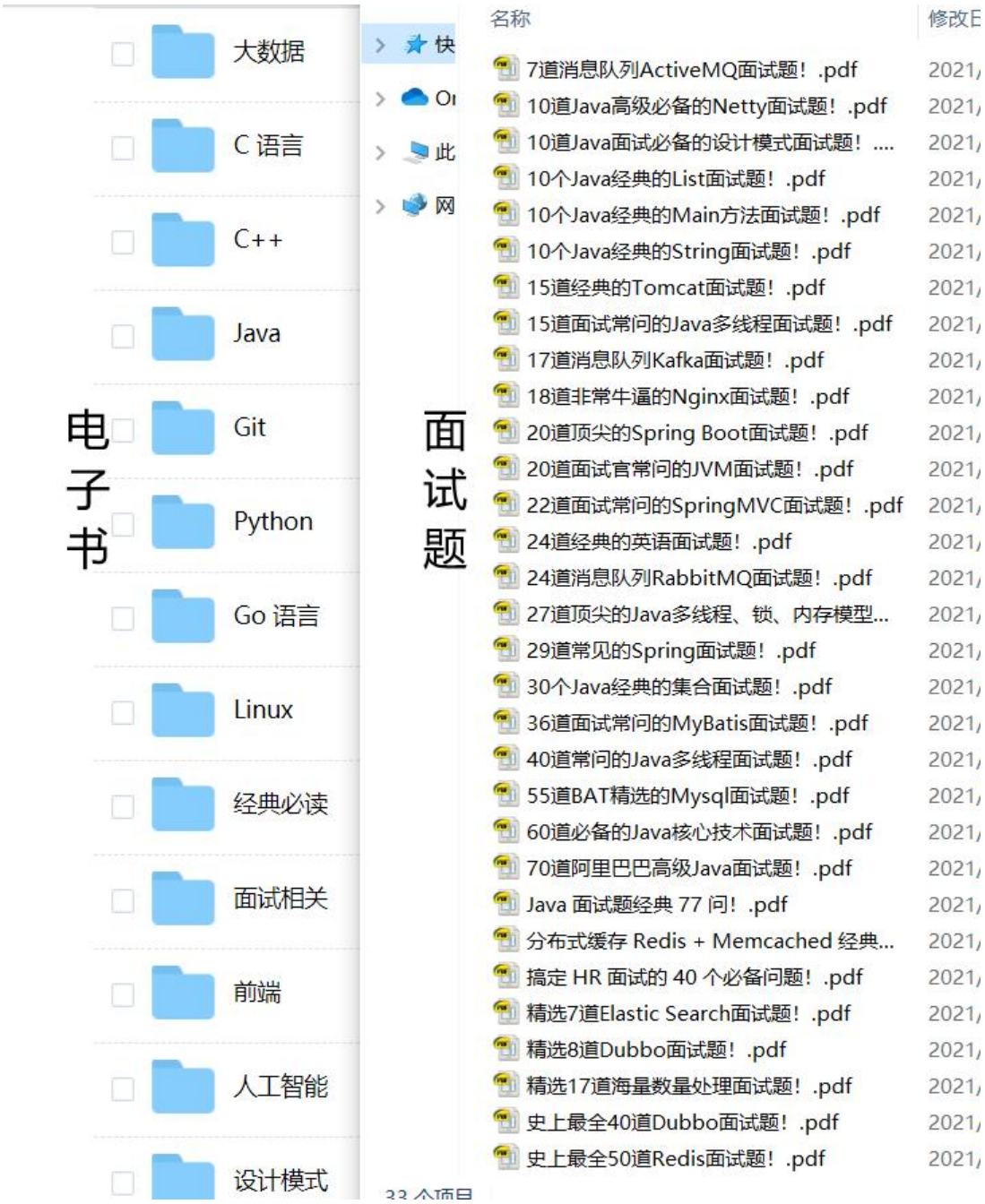
源码下载：https://github.com/turoDog/Demo/tree/master/springboot_jpa_demo

后语

以上为 SpringBoot 使用 Spring Data JPA 访问 Mysql 数据库的教程。

初次见面，也不知道送你们啥。干脆就送**几百本电子书**和**2021最新面试资料**吧。微信搜索**JavaFish**复**电子书**送你 1000+ 本编程电子书；回复**面试**获取 50 套大厂面试题；回复**1024**送你一套完整的 jav 视频教程。

面试题都是有答案的，如下所示：有需要的就来拿吧，**绝对免费，无套路获取。**



原文链接：SpringBoot 入门教程 (八) | 使用 Spring Data JPA 访问 Mysql 数据库

JavaFish

微信扫描二维码，关注我的公众号

原文链接：SpringBoot 入门教程 (八) | 使用 Spring Data JPA 访问 Mysql 数据库