spring-boot使用springDataJPA完成CRUD

笔记本：	spring_boot		
创建时间：	2018/9/14/周五 19:43	**更新时间：**	2018/9/14/周五 19:52
作者：	1634896520@qq.com		
URL：	about:blank		

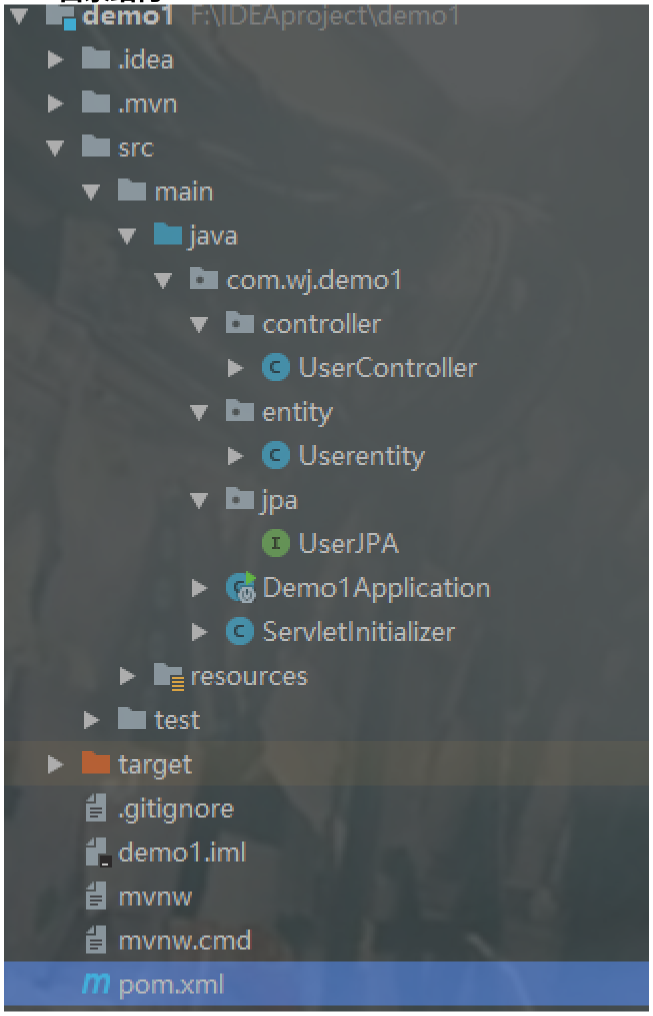
spring-boot使用springDataJPA完成CRUD

一、所用技术

maven管理jar包、spring-boot使用springJPA完成CRUD操作

二、代码demo

目录结构：



```
demo1 F:\IDEAproject\demo1
  .idea
  .mvn
  src
    main
      java
        com.wj.demo1
          controller
            UserController
          entity
            Userentity
          jpa
            UserJPA
          Demo1Application
          ServletInitializer
      resources
    test
  target
  .gitignore
  demo1.iml
  mvnw
  mvnw.cmd
  pom.xml
```

1. pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>

<groupId>com.wj</groupId>
<artifactId>demo1</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>war</packaging>
```

```
<name>demo1</name>
<description>Demo project for Spring Boot</description>

<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>2.0.5.RELEASE</version>
<relativePath/> <!-- lookup parent from repository -->
</parent>

<properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
<java.version>1.8</java.version>
</properties>

<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>

<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<scope>runtime</scope>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-tomcat</artifactId>
<scope>provided</scope>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
</dependencies>

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


</project>
```

2.UserController.java

```
package com.wj.demo1.controller;

import com.wj.demo1.entity.Userentity;
import com.wj.demo1.jpa.UserJPA;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
@RequestMapping("/user")
public class UserController {

@Autowired
private UserJPA userJPA;

@RequestMapping("/list")
public List<Userentity> list(){
return userJPA.findAll();
}

@RequestMapping("/save")
public Userentity save(Userentity userentity){
return userJPA.save(userentity);
}
}
```

3.UserEntity.java

```
package com.wj.demo1.entity;

import javax.persistence.*;
import java.io.Serializable;

@Entity
@Table(name = "t_user")
public class Userentity implements Serializable {
@Id
@GeneratedValue
@Column(name = "t_id")
protected long id;

@Column(name="t_name")
private String name;

@Column(name="t_age")
private int age;

@Column(name="t_address")
private String 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 int getAge() {
return age;
}

public void setAge(int age) {
this.age = age;
}

public String getAddress() {
return address;
}

public void setAddress(String address) {
this.address = address;
}
}
```

4.UserJPA.java(JPA重头戏)

```
package com.wj.demo1.jpa;

import com.wj.demo1.entity.Userentity;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.JpaSpecificationExecutor;

import java.io.Serializable;

/*
JpaRepository接口（SpringDataJPA提供的简单数据操作接口）
、JpaSpecificationExecutor（SpringDataJPA提供的复杂查询接口）
、Serializable（序列化接口）。

SpringDataJPA内部使用了类代理的方式让继承了它接口的子接口都以spring管理的Bean的形式存在，
也就是说我们可以直接使用@Autowired注解在spring管理bean使用
*/
public interface UserJPA extends
JpaRepository<Userentity,Long>,
JpaSpecificationExecutor<Userentity>,
Serializable {
}
```

5、Demo1Aplication.java

```
package com.wj.demo1;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Demo1Application {

public static void main(String[] args) {
SpringApplication.run(Demo1Application.class, args);
}
}
```

6、ServletInitializer.java

```
package com.wj.demo1;

import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

public class ServletInitializer extends SpringBootServletInitializer {

@Override
protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
return application.sources(Demo1Application.class);
}

}
```

7.application.yml

```
spring:
datasource:
url: jdbc:mysql://127.0.0.1:3306/test?characterEncoding=utf8
driver-class-name: com.mysql.jdbc.Driver
username: xxxxx
password: ***************
jpa:
database: MySQL
show-sql: true
hibernate:
naming strategy: org.hibernate.cfg.ImprovedNamingStrategy
```