Spring Boot 整合 Spring Data JPA

Spring Data JPA 是 Spring Data 大家族的一员

JPA 和 Spring Data JPA 的关系

JPA （Java Persistence API）Java 持久层规范，定义了一系列 ORM 接口，它本身是不能直接使用，接口必须实现才能使用，Hibernate 框架就是一个实现了 JPA 规范的框架。

Spring Data JPA 是 Spring 框架提供的对 JPA 规范的抽象，通过约定的命名规范完成持久层接口的编写，**在不需要实现接口的情况下，就可以完成对数据库的操作。**

简单理解，通过 Spring Data JPA 只需要定义接口而不需要实现，就能完成 CRUD 操作。

Spring Data JPA 本身并不是一个具体的实现，它只是一个抽象层，底层还是需要 Hibernate 这样的 JPA 来提供支持。

Spring Data JPA 和 Spring JdbcTemplate 的关系

Spring JdbcTemplate 是 Spring 框架提供的一套操作数据库的模版，Spring Data JPA 是 JPA 的抽象。

1、pom.xml

```
<!-- Spring Boot集成 Spring Data JPA -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

2、实体类，完成实体类与数据表的映射

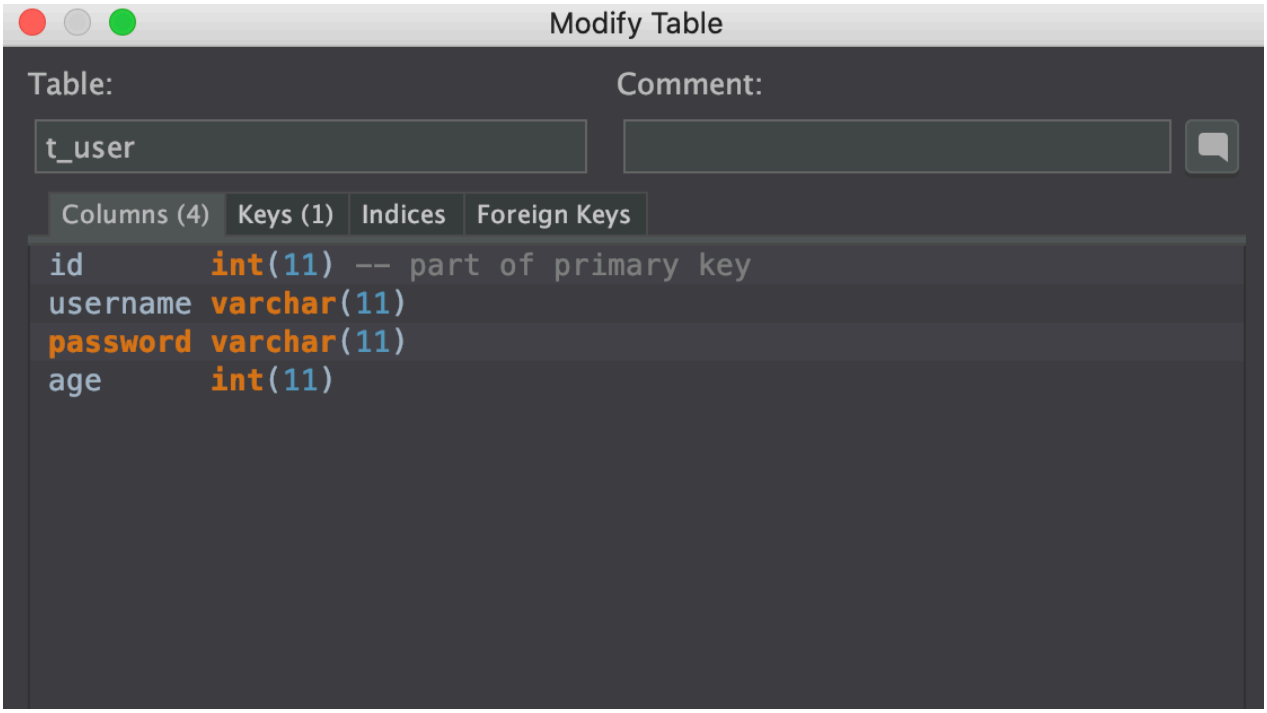
```
package com.southwind.jpa.entity;

import lombok.Data;

import javax.persistence.*;

@Data
@Entity(name = "t_user")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    @Column
    private String username;
    @Column
    private String password;
    @Column
```

```
    private Integer age;
}
```



- @Entity 将实体类与数据表进行映射
- @Id 将实体类中的成员变量与数据表的主键进行映射，一般都是 id
- @GeneratedValue 表示自动生成主键，strategy 为主键选择生成策略
- @Column 将实体类中的成员变量与数据表的普通字段进行映射

3、创建 UserRepository

```
package com.southwind.jpa.repository;

import com.southwind.entity.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User,Integer> {
}
```

4、创建 Handler

```
package com.southwind.controller.jpa;

import com.southwind.jpa.entity.User;
import com.southwind.jpa.repository.JpaUserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController("/jpaHandler")
@RequestMapping("/jpauser")
```

```
public class UserHandler {

    @Autowired
    private JpaUserRepository userRepository;

    @GetMapping("/findAll")
    public List<User> findAll(){
        return userRepository.findAll();
    }

    @GetMapping("/findById/{id}")
    public User findById(@PathVariable("id") Integer id){
        return userRepository.findById(id).get();
    }

    @PostMapping("/save")
    public void save(@RequestBody User user){
        userRepository.save(user);
    }

    @PutMapping("/update")
    public void update(@RequestBody User user){
        userRepository.save(user);
    }

    @DeleteMapping("/deleteById/{id}")
    public void deleteById(@PathVariable("id") Integer id){
        userRepository.deleteById(id);
    }
}
```

5、application.yml

```
spring:
  datasource:
    url: jdbc:mysql://localhost:3306/test?
useUnicode=true&characterEncoding=UTF-8
    driver-class-name: com.mysql.cj.jdbc.Driver
    username: root
    password: root
  jpa:
    show-sql: true
    properties:
      hibernate:
        format_sql: true
```

6、在继承 JpaRepsitory 的基础上，开发者也可以自定义方法。

```
@GetMapping("/findByUserName/{username}")
public User findByUserName(@PathVariable("username") String username){
    return userRepository.findByUsername(username);
}
```

Spring Boot 整合 Spring Security

1、创建 Maven 工程，pom.xml

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

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

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

2、Handler

```
package com.southwind.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class SecurityHandler {

    @GetMapping("/index")
    public String index(){
        return "index";
    }
}
```

3、HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <p>index</p>
    <form method="post" action="/logout">
        <input type="submit" value="退出"/>
    </form>
</body>
</html>
```

4、application.yml

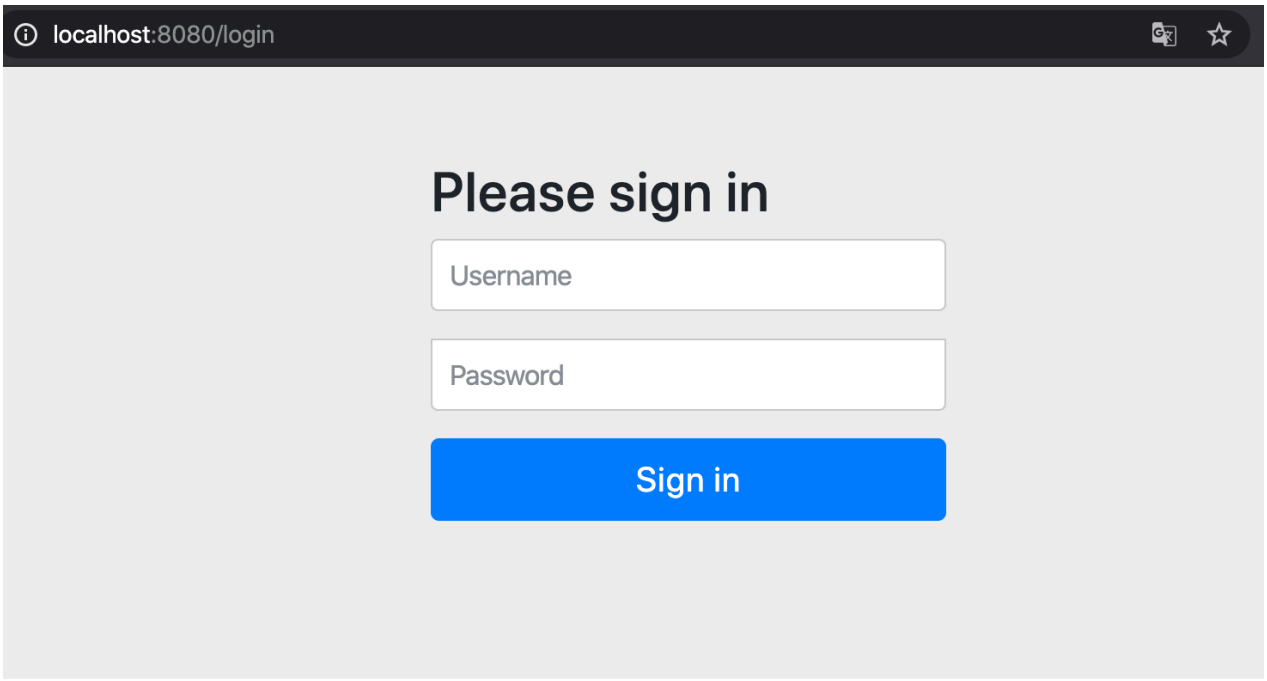
```
spring:
  thymeleaf:
    prefix: classpath:/templates/
    suffix: .html
```

5、Application

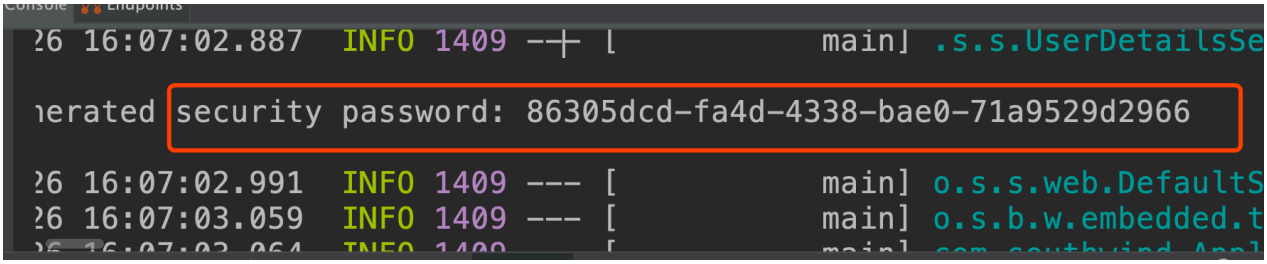
```
package com.southwind;

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

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



输入用户名、密码才可以进行访问，默认的用户名是 user，密码是启动 Spring Security 自动生成的随机密码。



自定义用户密码

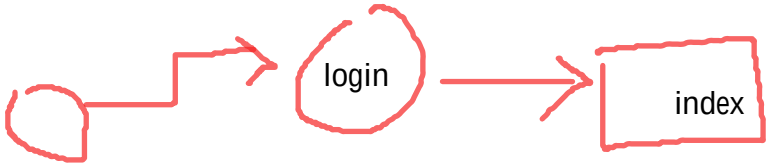
spring security有一个默认登陆的页面也有默认的用户和密码

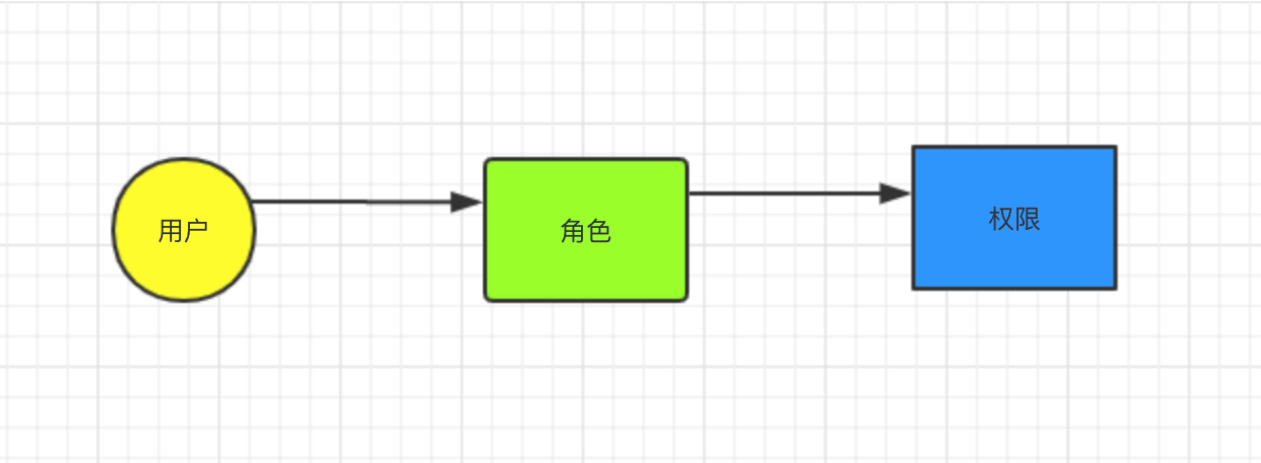
```yaml
spring:
  thymeleaf:
    prefix: classpath:/templates/
    suffix: .html
  security:
    user:
      name: admin
      password: 123123
```

权限管理

一般用这个框架做权限管理的开发，实际开发当中我们不会将权限和用户绑定而是将权限赋给角色，然后再给用户添加角色

springsecurity会对我们所有的请求进行权限验证





定义两个资源

- index.html
- admin.html

定义两个角色

- ADMIN 访问 index.html 和 admin.html
- USER 访问 index.html

1、创建 SecurityConfig 类　　在这个类中完成权限的管理

```
package com.southwind.config;

import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.Authent
icationManagerBuilder;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecu
rity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityCo
nfigurerAdapter;

@Configuration  配置类注解
@EnableWebSecurity      开启WebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    /**
     * 角色和资源的关系                    重写两个方法
     * @param http
     * @throws Exception
     */
    @Override
    protected void configure(HttpSecurity http) throws Exception {
```

```
        http.authorizeRequests()
                .antMatchers("/admin").hasRole("ADMIN")
                .antMatchers("/index").access("hasRole('ADMIN') or
hasRole('USER')")
                .anyRequest().authenticated()
                .and()
                .formLogin()
                .loginPage("/login")
                .permitAll()
                .and()
                .logout()
                .permitAll()
                .and()
                .csrf()
                .disable();
    }

    /**
     * 用户和角色的关系
     * @param auth
     * @throws Exception
     */
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws
Exception {
        auth.inMemoryAuthentication().passwordEncoder(new MyPasswordEncoder())
                .withUser("user").password(new MyPasswordEncoder()
                .encode("000")).roles("USER")
                .and()
                .withUser("admin").password(new MyPasswordEncoder()
                .encode("123")).roles("ADMIN","USER");
    }
}
```

访问admin需要admin这样一个角色，访问index需要admin或者user角色

user 绑定user角色

admin绑定admin和user两个角色

2、自定义 MyPassowrdEncoder

5.x版本需要提供password实例否则会抛异常，就是对密码的一个转码

```
package com.southwind.config;

import org.springframework.security.crypto.password.PasswordEncoder;

public class MyPasswordEncoder implements PasswordEncoder {
    @Override
    public String encode(CharSequence charSequence) {
        return charSequence.toString();
    }

    @Override
    public boolean matches(CharSequence charSequence, String s) {
        return s.equals(charSequence.toString());
```

```
    }
}
```

3、Handler

```
package com.southwind.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class SecurityHandler {

    @GetMapping("/index")
    public String index(){
        return "index";
    }

    @GetMapping("/admin")
    public String admin(){
        return "admin";
    }

    @GetMapping("/login")
    public String login(){
        return "login";
    }
}
```

4、login.html

```
<!DOCTYPE html>
<html lang="en">
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <p th:if="${param.error}">
        用户名或密码错误
    </p>
    <form th:action="@{/login}" method="post">
        用户名：<input type="text" name="username"/><br/>
        密码：<input type="password" name="password"/><br/>
        <input type="submit" value="登录"/>
    </form>
</body>
```

```
</html>
```

5、index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <p>欢迎回来</p>
    <form method="post" action="/logout">
        <input type="submit" value="退出"/>
    </form>
</body>
</html>
```

6、admin.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <p>后台管理系统</p>
    <form method="post" action="/logout">
        <input type="submit" value="退出"/>
    </form>
</body>
</html>
```