day02 - 后台项目框架结构搭建

一、SpringBoot技术快速入门

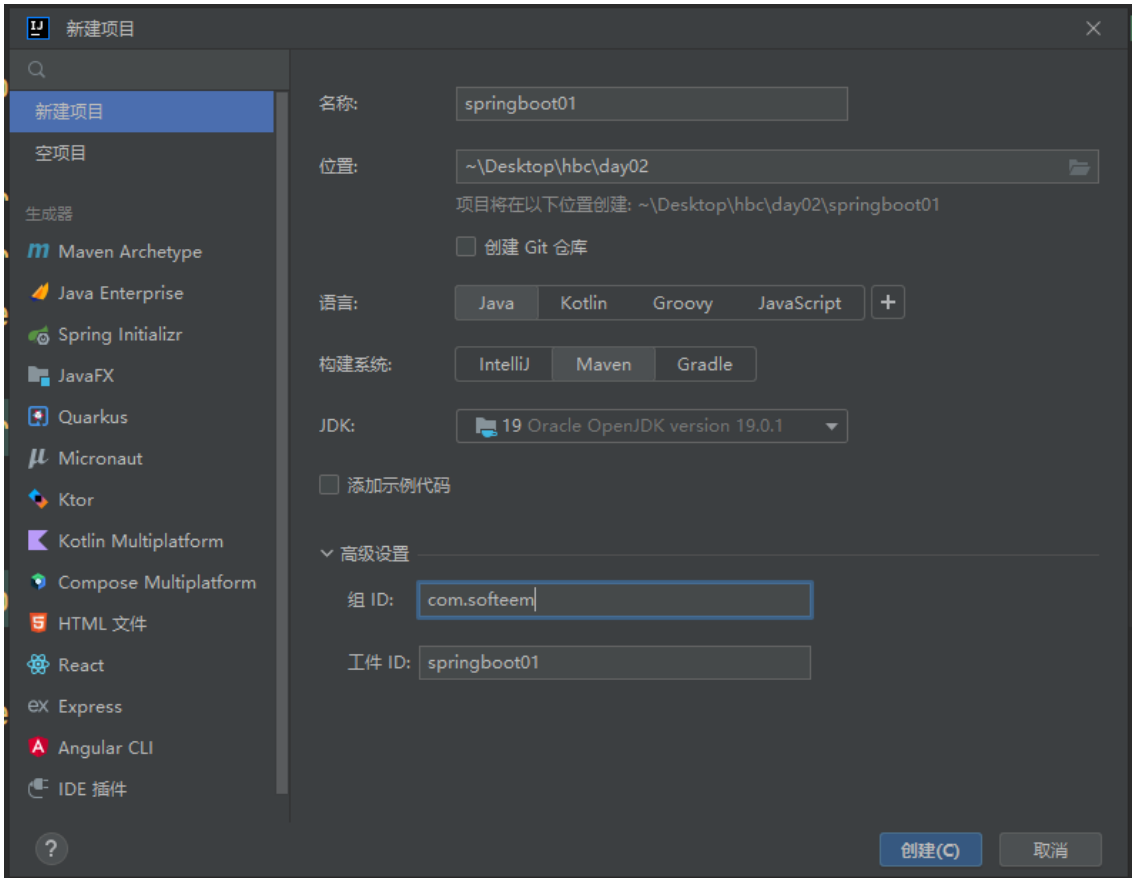
> SpringBoot是由Spring官方出品的一套用于快速搭建Spring开发环境的脚手架项目，真正的实现了开箱即用，零配置，一键启动的特点

SpringBoot前置技术与知识体系：

- 熟悉Java开发环境
- 熟悉Maven
- 了解SSM框架

1.1. 创建第一个SpringBoot应用

1. 创建Maven项目



2. 添加Springboot父工程到`pom.xml`

```
<!--    添加springboot父工程：核对springboot版本（springboot3.0要求jdk必须17+）-->
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.2.7</version>
</parent>
```

3. 添加springboot-web依赖（SpringMVC）

```
<!--依赖管理-->
<dependencies>
    <!--springboot-web依赖-->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
</dependencies>
```

4. 创建启动器类

```
@SpringBootApplication
public class MyApplication {

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

> Springboot支持一键启动，即：无需额外配置Tomcat服务器（内嵌tomcat）;运行springboot项目只需要一个简单的`main`方法即可

5. 创建控制器类以及对外提供的访问接口方法

```
@RestController
public class IndexController {

    @RequestMapping("/")
    public String index(){
        return "这是第一个SpringBoot程序,这里是首页!";
    }

}
```

6. 运行启动器类，打开浏览器，地址栏中输入访问地址：`http://127.0.0.1:8080/`

1.2. springboot配置文件

> 由于springboot已经按照约定优于配置策略将一些常规的框架配置默认做了设置，因此，大多数时候无需额外的配置；但是如果，对某些特定的需求，例如：修改服务器的默认端口，数据库连接信息等；springboot提供了一个统一的配置入口：
>
> `resources/application.yml`

```yaml
# 修改服务器默认端口号
server:
  port: 8081    # 建议1023~49151之间      端口取值范围0~65535
```

1.3. springboot项目中整合静态网页

> springboot项目默认允许静态页面的存在,通常在`resources`目录下提供了一个`static`目录，对该目录中的所有资源springBoot当做静态资源处理（不做拦截，允许直接访问），该目录中通常包含以下类型的文件：
>
> - `html`（静态网页）
> - `css`（样式文件）
> - `js`（脚本文件）
> - `img/audio/video`(媒体资源文件)

启动服务器，输入访问地址：`http://127.0.0.1:8081/index.html`

1.4. SpringBoot接受请求参数

> SpringBoot是一种服务端技术，通常是为客户端提供数据访问接口，因此接受客户端提交的数据也是一种常规能力

```
@RequestMapping("/login")
public String login(String username,String password){
    if(Objects.equals("admin",username) && Objects.equals("123456",password)){
        return "登录成功";
    }
    return "账号或密码错误";
}
```

> 其中：
>
> `username`和`password`是来自客户端提交的数据，在控制器方法中执行完相关逻辑操作之后，再通过`return`将结果反馈到客户端；客户端得到结果后再根据结果执行页面端的交互跳转

1.5. 案例实现用户登录流程

1. 在`resources/static`目录下创建静态页面

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <h1>这里是首页</h1>
    <hr>
    <form id="loginForm">
        <input type="text" name="username" placeholder="请输入用户名"><br>
        <input type="password" name="password" placeholder="请输入密码"><br>
        <button type="button">登录</button>
    </form>
    <!--引入jQuery库-->
    <script src="./js/jquery-3.7.1.min.js"></script>
    <script>
        $(function(){
            //选中按钮并绑定点击事件
            $('button').on('click',function(){
                //获取表单数据并序列化为查询参数
                let data = $('#loginForm').serialize();
                //发送请求到服务端
                $.get('/login',data,function(resp){
                    if(resp.code === 1){
                        location.href='main.html'
                    }else{
                        alert(resp.msg)
                    }
                })
            })
        })
    </script>
</body>
```

```
</html>
```

2. 修改后台接口代码（控制器）

```
@RequestMapping("/login")
public Map<String,Object> login(String username, String password){
    //创建Map集合对象，用于传递丰富的数据到前端
    Map<String,Object> map = new HashMap<>();
    if(!Objects.equals("admin",username)){
        map.put("code",-1);
        map.put("msg","用户不存在");
    }else{
        if(!Objects.equals("123456",password)){
            map.put("code",0);
            map.put("msg","密码错误");
        }else{
            map.put("code",1);
            map.put("msg","登录成功");
        }
    }
    return map;
}
```

3. 前端发送请求到后台

二、SpringBoot整合MybatisPlus

> MybatisPlus是一个基于Mybatis增强的持久层框架，MyBatis-Plus 是一个 MyBatis 的增强工具，在 MyBatis 的基础上只做增强不做改变，为简化开发、提高效率而生。

1. 引入相关的依赖

```xml
<!--mybatisplus-springboot依赖-->
<dependency>
    <groupId>com.baomidou</groupId>
    <artifactId>mybatis-plus-spring-boot3-starter</artifactId>
    <version>3.5.7</version>
</dependency>

<!--mysql驱动包-->
<dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
</dependency>
```

2. 配置数据源

```yaml
#配置数据源
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://127.0.0.1:3306/hbc
    username: root
    password: 123456
```

3. 主启动类中配置映射器所在的包地址（扫描dao层）

```java
//配置映射器所在的包地址（Spring容器会自动扫描并注册，包地址通常为数据访问层，即dao/mapper）
@MapperScan("com.softeem.springbootdemo.dao")
@SpringBootApplication
public class MyApplication {

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

4. 创建对应数据表的实体类

```java
@Data
public class User {

    private Integer id; //int - integer
    private String username;
    private String password;
    private LocalDateTime regtime;
    private String phone;
    private String img;
    private LocalDate birth;
    private Integer status;
    private String salt;

}
```

> 由于使用了`lombok`自动生成模板代码，因此，pom.xml文件中需要引入`lombok`的依赖

```
<!--lombok插件，用于自动生成模板代码
（setter/getter,toString,equals&hashCode）-->
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
</dependency>
```

5. 创建映射器接口（DAO）

```
@Mapper
public interface UserDAO extends BaseMapper<User> {


}
```

> 由于大多时候dao层都是编写的一些CRUD方法，mybatisplus对于常规的增删改查操作已经做了定义和实现，因此，自定义的DAO接口（也叫Mapper）只需要从BaseMapper接口继承即可，除此之外，无需任何代码！

6. 在控制层调用数据访问层实现

```
@RestController
public class IndexController {

    @Autowired
    private UserDAO userDAO;


    @PostMapping("/reg")
    public Result reg(String username,String password){
        Result r = new Result();
        User user = new User();
        user.setUsername(username);
        user.setPassword(password);
        int i = userDAO.insert(user);
        if(i > 0){
            r.setCode(1);
            r.setMsg("注册成功");
            r.setData(user);
        }else{
            r.setCode(-1);
            r.setMsg("注册失败");
        }
        return r;
    }
}
```