自动编译依赖

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <optional>true</optional>  
</dependency>

配置文件添加

# 热部署生效  
spring.devtools.restart.enabled=true  
  
# 设置重启目录  
spring.devtools.restart.additional-paths=src/main/java  
  
spring.servlet.multipart.max-file-size=10MB  
spring.servlet.multipart.max-request-size=10MB

spring.datasource.driver-class-name=com.mysql.jdbc.Driver  
spring.datasource.type=com.alibaba.druid.pool.DruidDataSource  
spring.datasource.url=jdbc:mysql://localhost:3306/mydb?useSSL=false  
spring.datasource.username=root  
spring.datasource.password=1234  
mybatis-plus.configuration.log-impl=org.apache.ibatis.logging.stdout.StdOutImpl

其余操作看编程技巧文档------h

Poml

<?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/4.0.0 https://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>2.5.5</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.example</groupId>  
 <artifactId>h</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>h</name>  
 <description>Demo project for Spring Boot</description>  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>com.baomidou</groupId>  
 <artifactId>mybatis-plus-boot-starter</artifactId>  
 <version>3.4.2</version>  
 </dependency>  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>5.1.45</version>  
 </dependency>  
 <dependency>  
 <groupId>com.alibaba</groupId>  
 <artifactId>druid</artifactId>  
 <version>1.1.20</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>io.springfox</groupId>  
 <artifactId>springfox-swagger2</artifactId>  
 <version>2.9.2</version>  
 </dependency>  
 <dependency>  
 <groupId>io.springfox</groupId>  
 <artifactId>springfox-swagger-ui</artifactId>  
 <version>2.9.2</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <optional>true</optional>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

FileUploadController类代码负责上传文件

package com.example.h.controller;  
  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.web.multipart.MultipartFile;  
  
import javax.servlet.http.HttpServletRequest;  
import java.io.File;  
import java.io.IOException;  
  
@RestController  
public class FileUploadController {  
 @RequestMapping  
 public String up(String nickname, MultipartFile photo, HttpServletRequest request) throws IOException {  
 System.*out*.println(nickname);  
// 获得图片的原始名称  
 System.*out*.println(photo.getOriginalFilename());  
// 获取文件类型  
 System.*out*.println(photo.getContentType());  
 System.*out*.println(System.*getProperty*("user.dir"));  
 String path=request.getServletContext().getRealPath("/upload/");  
 System.*out*.println(path);  
  
 saveFile(photo,path);  
 return "上传成功！";  
 }  
  
 private void saveFile(MultipartFile photo, String path) throws IOException {  
// 判断存储的目录是否存在，如果不存在就创建  
 File dir=new File(path);  
 if(!dir.exists()){  
// 创建目录  
 dir.mkdir();  
 }  
 File file=new File(path+photo.getOriginalFilename());  
 photo.transferTo(file);  
 }  
}

拦截器类LoginInterceptor模板

package com.example.h.interceptor;  
  
import org.springframework.web.servlet.HandlerInterceptor;  
  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
  
public class LoginInterceptor implements HandlerInterceptor {  
 @Override  
 public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {  
 System.*out*.println("Logininterceptor!");  
 return false;  
 }  
}

WebConfig 模板类

package com.example.h.config;  
  
import com.example.h.interceptor.LoginInterceptor;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;  
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  
@Configuration  
public class WebConfig implements WebMvcConfigurer {  
 @Override  
 public void addInterceptors(InterceptorRegistry registry) {  
 registry.addInterceptor(new LoginInterceptor());  
 }  
}

Swagger2Config类的模板

package com.example.h.config;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import springfox.documentation.RequestHandler;  
import springfox.documentation.builders.ApiInfoBuilder;  
import springfox.documentation.builders.PathSelectors;  
import springfox.documentation.builders.RequestHandlerSelectors;  
import springfox.documentation.service.ApiInfo;  
import springfox.documentation.service.Documentation;  
import springfox.documentation.spi.DocumentationType;  
import springfox.documentation.spring.web.plugins.Docket;  
import springfox.documentation.swagger2.annotations.EnableSwagger2;  
//查看途径 http://127.0.0.1:8080/swagger-ui.html  
@Configuration//告诉spring这是一个容器  
@EnableSwagger2//启用Swagger2功能  
public class Swagger2Config {  
 @Bean  
 public Docket createRestApi(){  
 return new Docket(DocumentationType.*SWAGGER\_2*)  
 .apiInfo(apiInfo())  
 .select()  
 .apis(RequestHandlerSelectors.*basePackage*("com"))  
 .paths(PathSelectors.*any*()).build();  
 }  
  
 private ApiInfo apiInfo() {  
 return new ApiInfoBuilder()  
 .title("演示项目api")//标题  
 .description("描述")//描述  
 .build();  
 }  
}

启动类加

@MapperScan("com.ysu.mpdemo.mapper")