# 学习springboot的最基本的用法

## 搭建无ide的springboot的开发环境

## 搭建IntelliJ IDEA的springboot的开发环境

## 增加Controller类

@RestController  
class Controller{

}

## 增加get方法

### 无请求参数的get

<http://localhost:8080/hello1/>

hello1地址后面有其他内容，否则就会找不到页面，包404错误

@GetMapping("hello1")  
public String hello1(){  
 return "hello1()";  
}

### 参数直接在路径中

<http://localhost:8080/hello2/123>

Hello2地址后面必须要有其他内容，否则就会找不到页面，包404错误

http://localhost:8080/hello2/123/123也会报404错误

@GetMapping("hello2/{name}")  
public String hello2(@PathVariable("name") String name){  
 return "hello2() 获取到的name是：" + name;  
}

### 参数跟在 ? 号后面

<http://localhost:8080/hello3?name=123>

@GetMapping("hello3")  
public String hello3(@RequestParam("name") String name){  
 return "hello3() 获取到的name是：" + name;  
}

### 参数跟在 ? 号后面，且**required = false** 标注参数是非必须的

@GetMapping("hello4")  
public String hello4(@RequestParam(name = "name",required = false) String name){  
 return "hello4() 获取到的name是：" + name;  
}

则有下面几种场景：

#### 正常场景



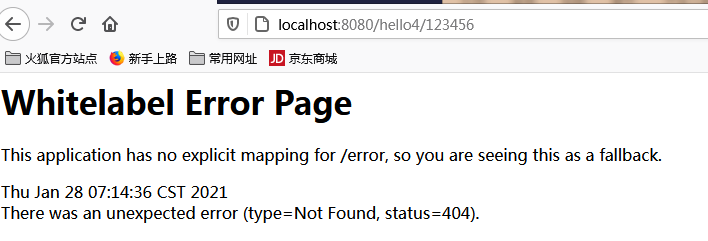
#### 指定名词不指定值



#### 不指定参数



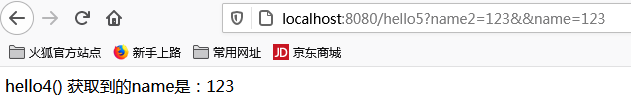
#### 不指定参数，而是跟斜杠



### 参数跟在 ? 号后面，且指定个默认值

@GetMapping("hello5")  
public String hello5(@RequestParam(name = "name",defaultValue = "xxx") String name){  
 return "hello4() 获取到的name是：" + name;  
}

#### 正常场景



#### 指定参数，但是缺少需要的参数



#### 指定参数但不指定值



#### 指定参数但不指定等号



#### 不指定参数



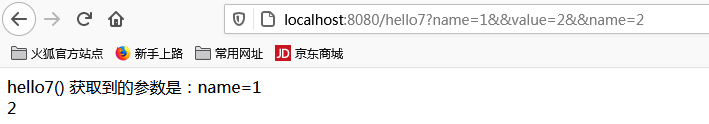
### 使用 map 来接收参数

@GetMapping("hello6")  
public String hello6(@RequestParam Map<String, Object> params){  
 return "hello6() 获取到的参数是：name=" + params.get("name") + ", vamue=" + params.get("value");  
}



### 接收一个数组

@GetMapping("hello7")  
public String hello7(@RequestParam("name") String[] names){  
 String result = "";  
 for(String name:names){  
 result += name + "<br>";  
 }  
 return "hello7() 获取到的参数是：name=" + result;  
}



### 定义类来接收参数

## 增加swagger，生成接口文档

### 在pom文件中：**导入依赖**

<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>

增加后如下图：

### **编写Swagger2的配置类**

**添加如下的类**

@Configuration  
@EnableSwagger2  
public class SwaggerConfig {  
 @Bean  
 public Docket createRestApi() {  
 return new Docket(DocumentationType.*SWAGGER\_2*)  
 .pathMapping("/")  
 .select()  
 .apis(RequestHandlerSelectors.*basePackage*("com.example.demo"))  
 .paths(PathSelectors.*any*())  
 .build().apiInfo(new ApiInfoBuilder()  
 .title("SpringBoot整合Swagger")  
 .description("SpringBoot整合Swagger，详细信息......")  
 .version("9.0")  
 //.contact(new Contact("啊啊啊啊","blog.csdn.net","aaa@gmail.com"))  
 .license("The Apache License")  
 .licenseUrl("http://www.baidu.com")  
 .build());  
 }  
}

### **配置某个Controller**

**在controller类前加注解:**

@Api(tags = "用户接口")  
@RestController  
@EnableSwagger2  
class Controller{

### **配置某个接口方法**

@GetMapping("hello2/{name}")  
@ApiOperation("第二接口")  
@ApiImplicitParams({  
 @ApiImplicitParam(name = "name", value = "用户名", defaultValue = "李四")  
})

### **测试**

在浏览器输入地址:http://localhost:8080/swagger-ui.html#/

### **使用新的主题**

**在pom增加依赖**

<dependency>  
 <groupId>com.github.caspar-chen</groupId>  
 <artifactId>swagger-ui-layer</artifactId>  
 <version>1.1.3</version>  
</dependency>

**在浏览器输入新的地址**

<http://localhost:8080/docs.html>

调试

## 增加post方法

## 增加delete方法

## 增加update方法

## 增加数据库访问

## 增加数据库连接池的访问方法

## 代码打包

# 自己动手写“最简版的springboot，”最基本的原理）

# 自己动手写“简化版的spring boot”，了解spring boot的大致结构

# 学习spring boot的源代码

# spring boot微服务部署

# spring boot反向代理