@RequestMapping

1. **修饰**
2. 修饰方法.

**映射请求**，直接在方法上，无需再写serlvet

**@RequestMapping**("/helloworld")

**public** String hello() {

**return** "success";

}

该方法就可以响应helloworld的请求了，要求这个类必须在IOC容器中，然后会**转发**到解析器解析过后的页面

**返回值可以通过视图解析器，解析成实际的物理视图**

视图解析器会把返回值解析成

**前缀+返回值+后缀**

前缀后缀在配置文件中配置

1. 修饰类

修饰类时，URL时基于web应用的，类中方法的URL是基于类的

如，

@RequestMapping("/hello")

@Controller

**public** **class** SpringMVCTest {

**private** **static** **final** String ***SUCCESS***="success";

@RequestMapping("/world")

**public** String world() {

System.***out***.println("hello world!");

**return** ***SUCCESS***;

}

}

World方法的请求就变成了/hello/world

1. **请求方式**

Method属性，很常用，指定接受的请求方式

@RequestMapping(value = "/testMethod", method =

RequestMethod.***POST***)

**public** String testMothod() {

**return** ***SUCCESS***;

}

1. **请求参数，请求头（了解）**

请求参数中必须含有username，不能含有age，sex不能等于男

@RequestMapping(value="/testParan",params= {"username","!age","sex!='男'"},headers= {""})

**public** String testParams() {

**return** ***SUCCESS***;

}

请求头类似

1. **路径Ant通配符**

3种匹配符

* ?：匹配文件名中的一个字符
* \*：匹配文件名中的一个或多个字符
* \*\*：匹配多层路径

@RequestMapping(value="/testAntPath/\*/p")

了解

1. **PathVariable注解**

映射url绑定的占位符，可以映射url中的占位符到目标方法的参数中

@RequestMapping(value = "testPathVariable/{id}")

**public** String testPathVariable(@PathVariable Integer id) {

System.***out***.println("id：" + id);

**return** ***SUCCESS***;

}