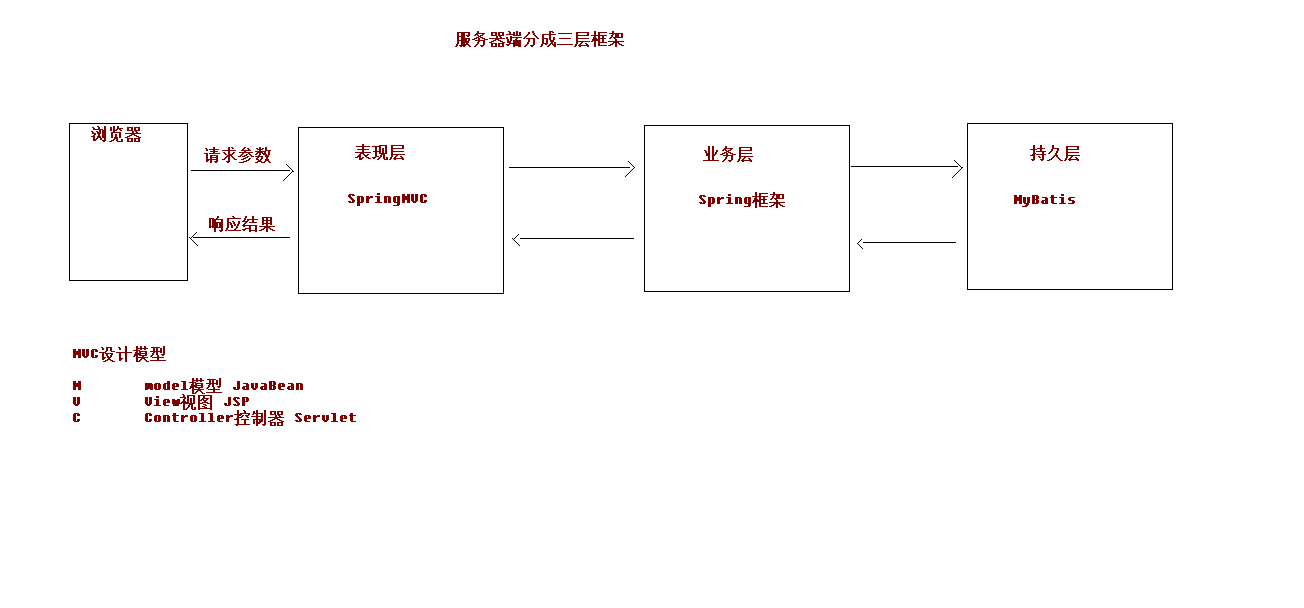
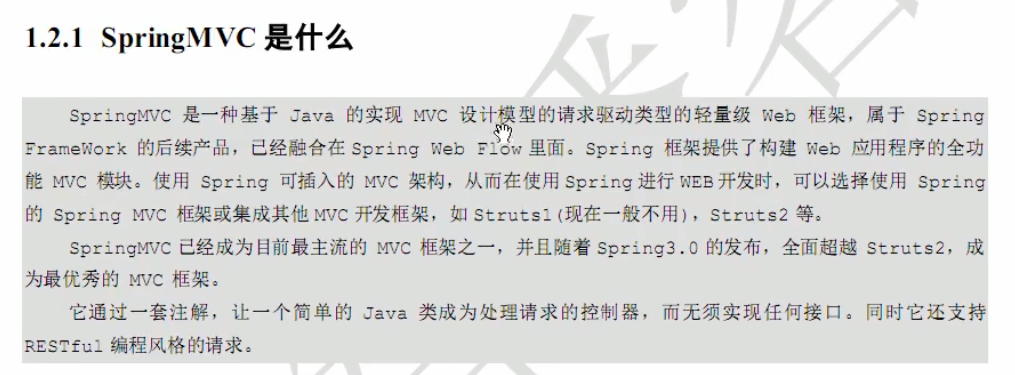
老样子，先来看看大体架构：

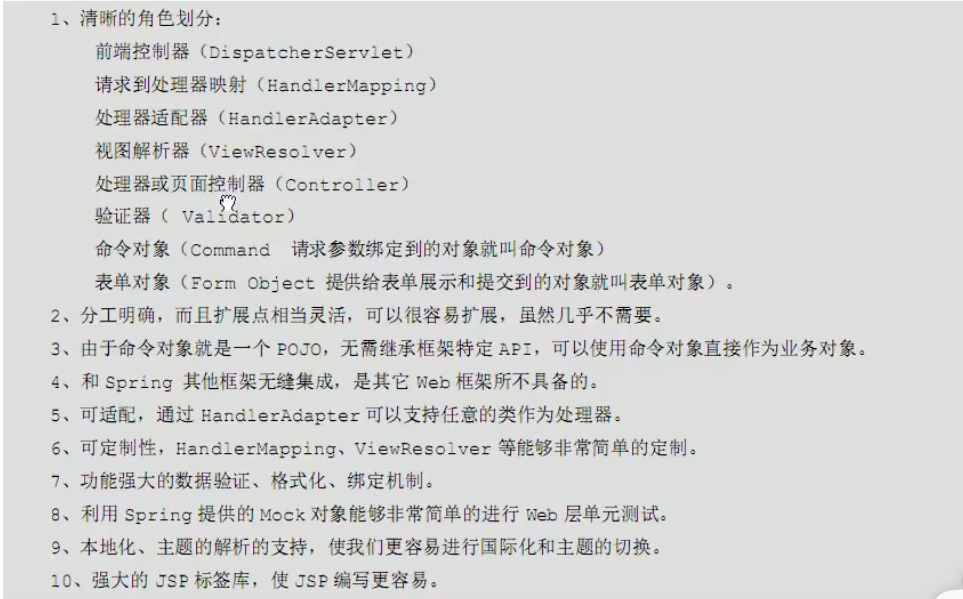


SpringMVC框架本身：



也就是说前端不用mvc框架也可以，spring都会帮你整合（比如vue）。

SpringMVC的优势在于：



当然了，springMVC也支持rest风格！

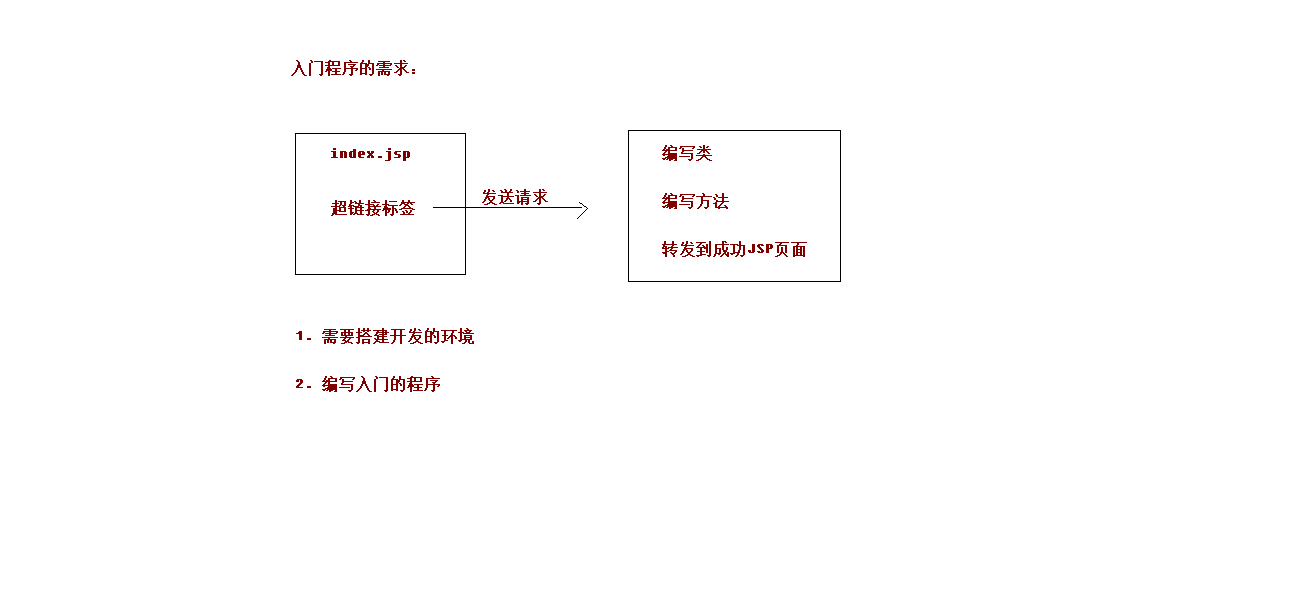
#第一个示例程序

接下来我们编写第一个springMVC程序。

该程序的功能是：

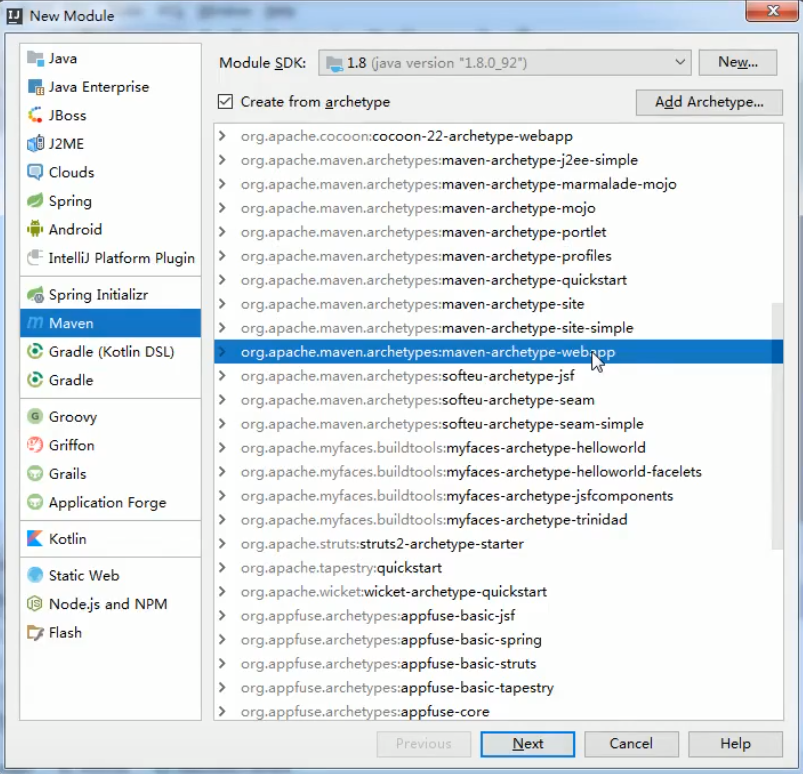
有一个页面，上面有一个超链接可以点击；

前端点击了这个链接，就会调用后端一个特定类的特定函数；该函数的功能是：跳转到一个特定jsp页面上！

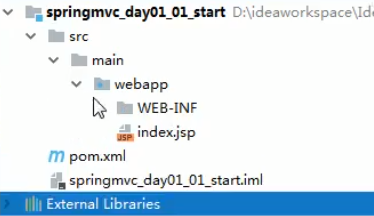


搭建环境：

新建项目选maven项目，然后选webapp开发！

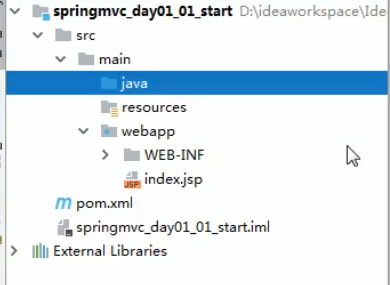


项目初始文件结构：



多了webapp包；多了.iml配置文件。

目录结构补全：



坑：没法新建配置文件or类or包：

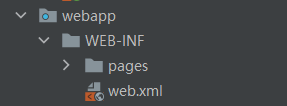
右键java，设置为root文件夹；

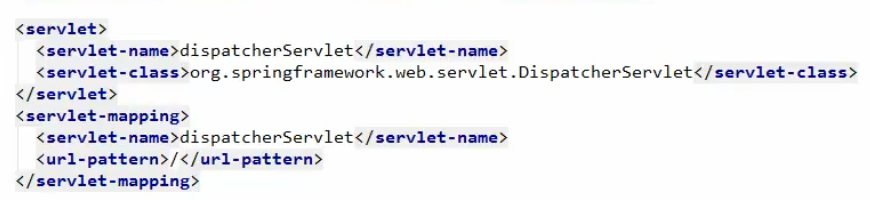
右键resources，设置为sources文件夹。

然后是常规的pom中配置工作，此处省略；

接下来，想要使用springMVC第一步：加入【前端控制器】！（其实就是一个servlet）

在webapp.WEB-INF.web.xml中配置





都是固定的！

第二步：在resources中新建.xml项目配置文件。

第三步：配置tomcat服务器！

如何安装：

<https://www.runoob.com/servlet/servlet-environment-setup.html>

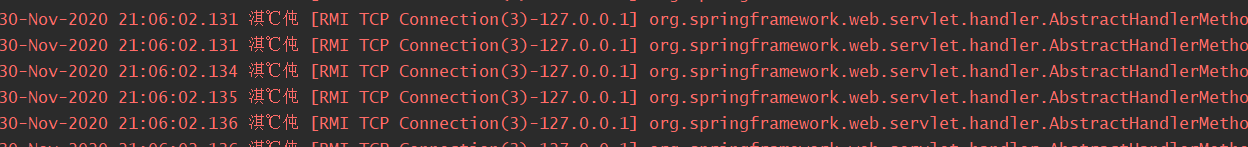
如何在idea中配置：

<https://www.jianshu.com/p/91a2d0a1e45a>

坑：运行报错Address localhost:8080 is already in use.更改tomcat的默认端口就好了。

<https://blog.csdn.net/Jay_1989/article/details/52870760>

坑：运行乱码：



解决：<https://blog.csdn.net/B2345012/article/details/88365722>

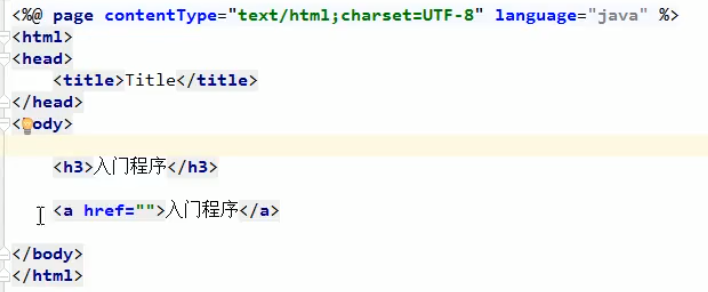
环境搭建搞定！

接下来我们开写入门程序！

第一步：进入index.jsp。默认的index.jsp没写头部编码2，咱给他加上一行：

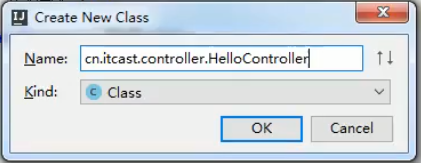


然后写一个简单的超链接：



Href里写啥先留个悬念。

第二步：到java里编写对应的类来处理点击超链接的请求



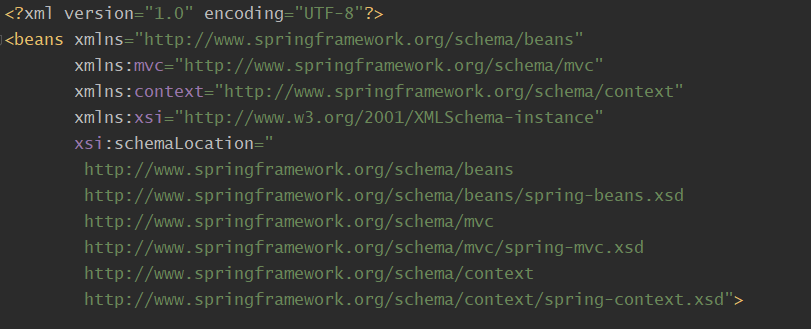
文件结构如上。写一个简单的hello：



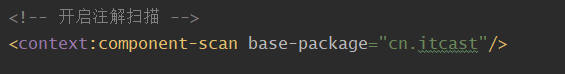
接下来，如何才能让点击那个超链接和这个函数关联起来呢？

用spring中讲到过的ioc！（使用注解方法）

在项目配置文件springmvc.xml中，咱们先导入必要的坐标：



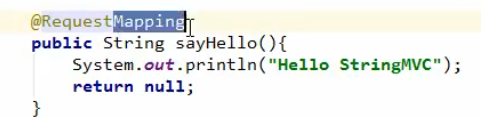
然后开始注解扫描：



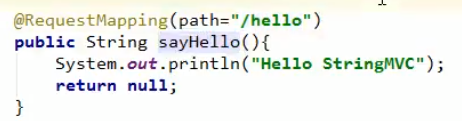
这样我们就可以给hellocontroller类加注解了：



然后再给想要调用的方法加注解：



Request为请求，mapping为映射，正如其名！该注解还需要一个参数，确定请求该方法的路径：



不过这个路径并不是指url中的路径，而是指前端发出请求的路径。

所以前端页面的href我们应该填上：

第三步：让前端控制器servlet导入我们的项目配置文件springmvc.xml

在web.xml中：

Servlet有固定的接受配置文件参数，名为contextConfigLocation。如下：



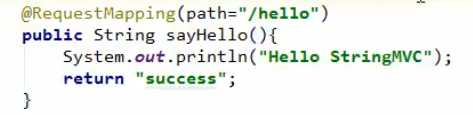
到此为止，我们就实现了：点击超链接，对应的方法sayhello（）会执行！

接下来，我们要实现页面跳转！

这里有一个规则：

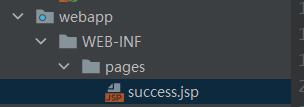
如果我们让sayhello（）返回一个字符串，这个字符串就【默认】是新页面的名字！（需要配置一个视图解析器）

例如：



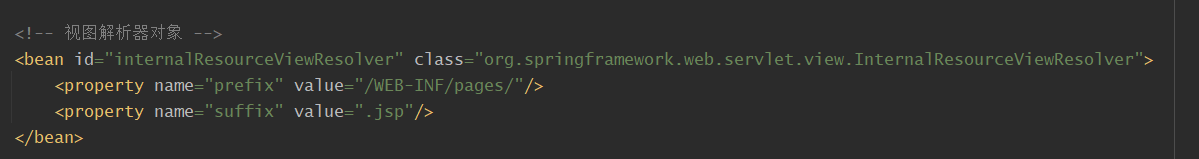
新的跳转页面就是success.jsp！

于是我们写一个名为success的页面：

（常用文件结构）

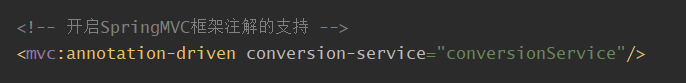
接下来我们配置一下【视图解析器】，有了它，程序就会根据返回的页面名字“success”寻找名为success的页面。

在项目配置文件springmvc.xml中：



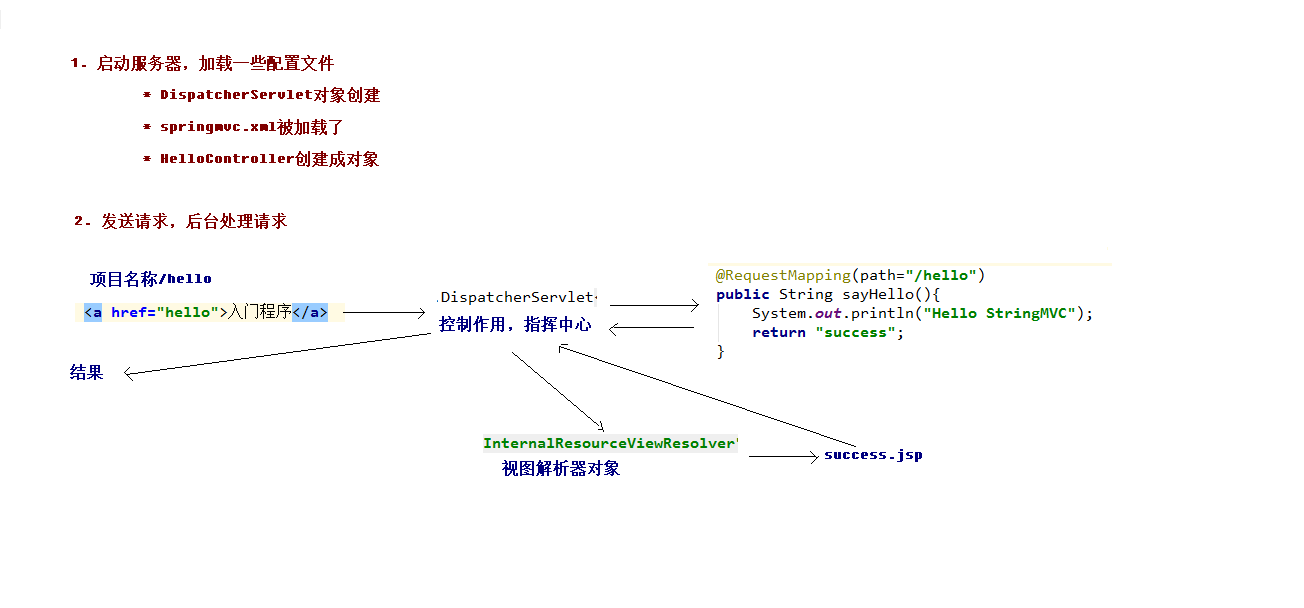
Prefix是跳转页面所在的目录；suffix是跳转页面的后缀名。

最后一步：打开springmvc框架对注解的支持！

就一句：

该看p164了。

解析来我们对上述例子做一个简单的过程回顾：



Servlet到底是干嘛的：

前端发出请求后，由后端的哪个函数来接呢？这个过程就是servlet处理的！

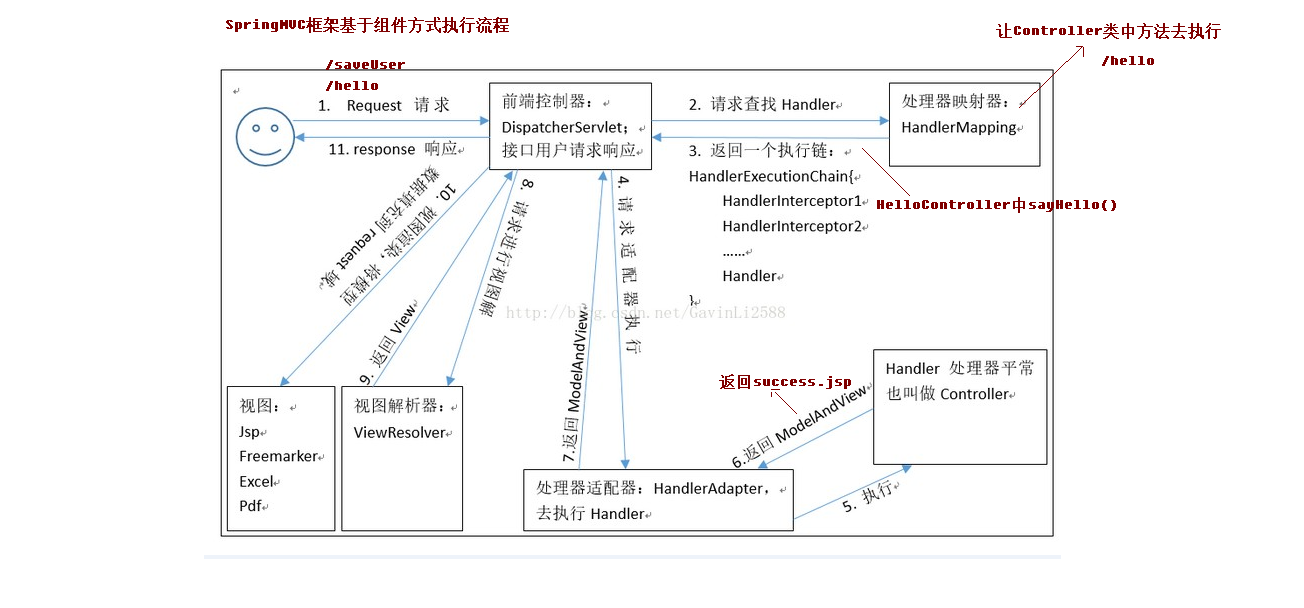
它是一个【指挥中心】：

首先它收到前端的请求，然后根据注解配置、找到对应的后端函数，将其调用；

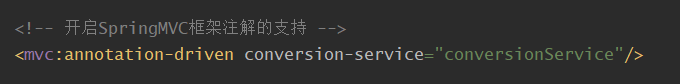
后端函数返回一个值给它，它在根据视图解析器的配置（也是注解）、找到要返回的页面hello.jsp；

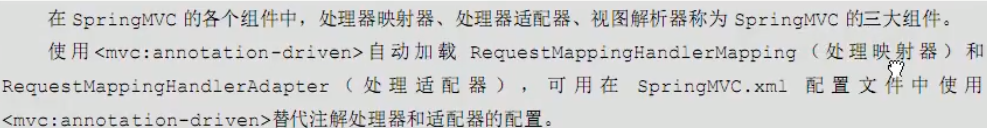
然后把要返回的页面返回给前端进行跳转！

具体而言：



这时候你可能会问：这么多“器”我都没配置过啊，怎么能使用呢？

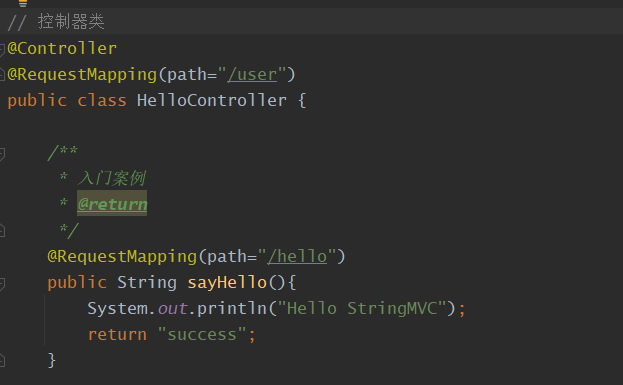
事实上我们配置了：

就相当于配置了处理器映射器和处理器适配器！

#RequestMapping注解

基础的用法就是加在方法前面（如上例）；

但其实也可以加在类前面，like：

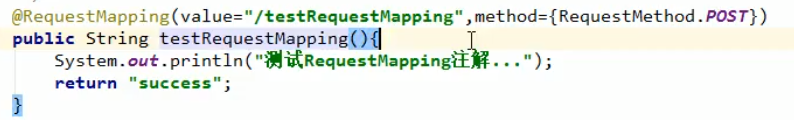


这样的话前端请求时href应该是：user/hello！

RequestMapping中的【属性】：

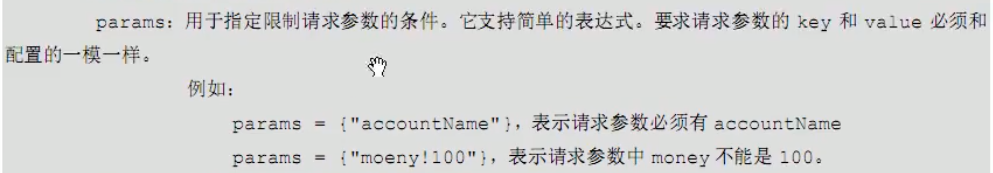
首先是【path】（也可以写成【value】 完全相同！）：如果你的requestMapping只有这一个属性，可以直接省略写为

【method】属性：用于规定调用该方法的请求类型！

例如：

加上这个属性，以后只有POST请求可以调用该函数！（上例的超链接就不能调用该函数了，因为超链接的请求固定是GET类型！）

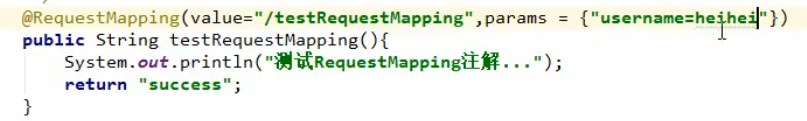
【params】属性：



比如上例的超链接请求肯定是不行了，因为那个请求没有传任何参数！

请求如何传参数：就可以了！

当然如果你把params属性改成：



上述超链接也不行了，虽然传了username属性、但是值不对！

【headers】属性：

举例：

请求中必须包含Accept请求头才能访问！

请求头相关知识后面会讲。

如何看一个请求的请求头？

浏览器内按f12，选择“网络”选项，就可以看请求的各种信息！（其实postman也可以）

该看168了！