1.发起请求request到前端控制器DisppatcherServlet
2.前端控制器根据url请求处理映射器HandlerMapping
查找Handler(XML配置/注解@RequestMapping声明映射关系)
3.处理映射器HandlerMapping向前端控制器
返回HandlerExecutionChain{HandlerIntercepter, Handler}
4.前端控制器调用处理适配器HandlerAdapter去执行各种各样的Handler(@Controller声明是Handler类)
5.处理适配器执行Handler
6.Handler执行完给处理适配器返回ModelAndView
7.处理适配器向前端控制器返回ModelAndView
8.前端控制器请求试图解析器进行视图解析
9.视图解析器向前端控制器返回各种各样的view(jsp, Freemarker等)
10.前端控制器进行视图渲染
11.前端控制器相应用户相应response

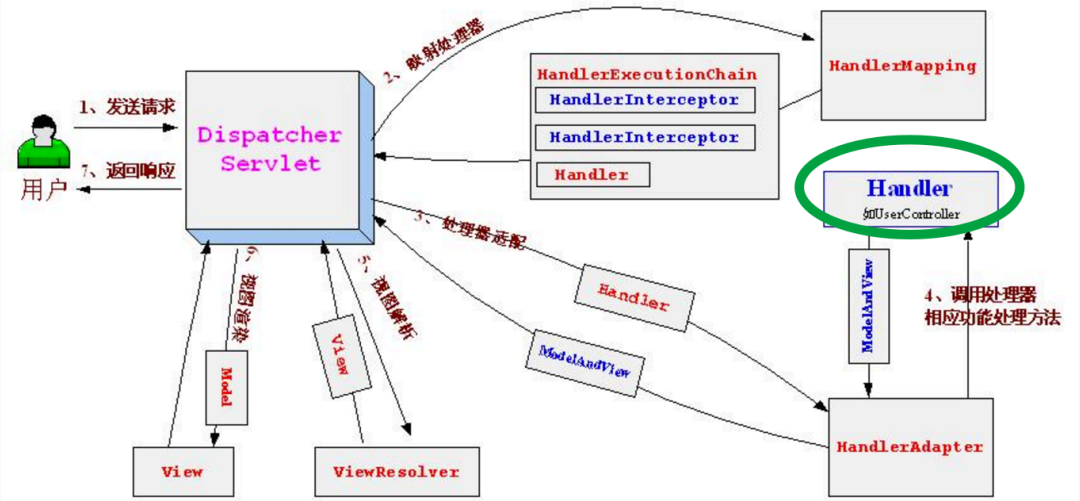
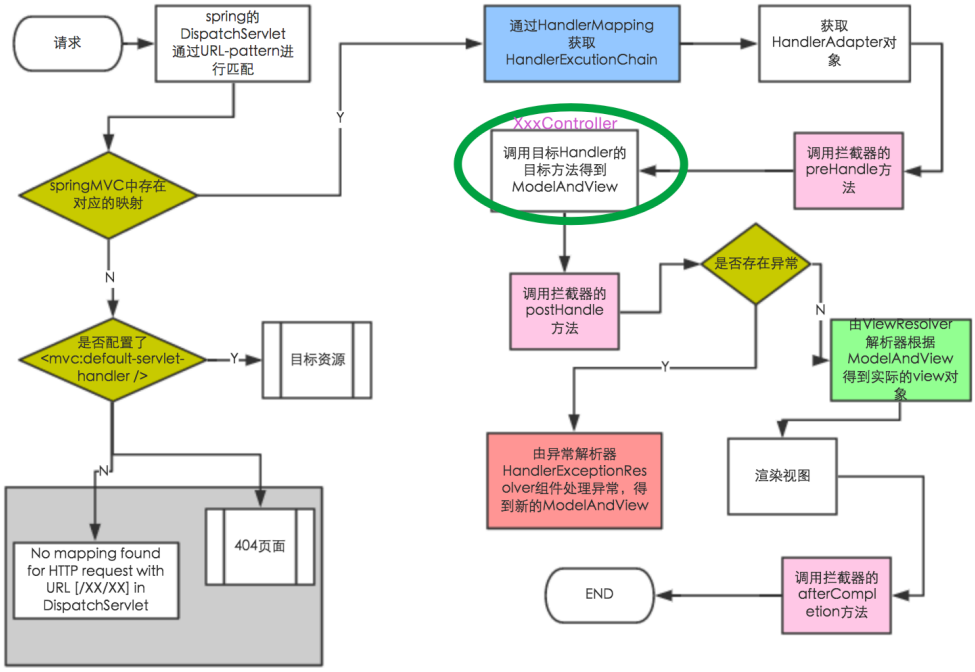


图2-2



SpringMVC的简单使用

SpringMVC的控制转发

SpringMVC是基于DispatcherServlet的MVC框架，DispatcherServlet的继承关系为：

```
HttpServlet <-- HttpServletBean <-- FrameworkServlet <--
DispatcherServlet
```

每一个请求最先访问的都是DispatcherServlet，DispatcherServlet负责转发Request请求给相应的Handler，Handler处理以后再返回相应的视图(View)或模型(Model)或都不返回。

在使用注解的SpringMVC中，处理器Handler是基于@Controller和@RequestMapping这两个注解的，@Controller声明一个处理器类，@RequestMapping声明对应请求的映射关系，这样就可以提供一个非常灵活的匹配和处理方式。

web配置

要想使用SpringMVC，就得在web.xml文件中像配置普通servlet那样对DispatcherServlet进行配置：

```
<servlet>
    <servlet-name>web</servlet-name>
    <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath*:spring.xml</param-value>
    </init-param>
</servlet>
<servlet-mapping>
    <servlet-name>web</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
```

上面的servlet配置中直接通过初始化参数设置了contextConfigLocation，这样就会去指定的位置加载spring配置；如果设置的话，则SpringMVC会自动到/WEB-INF目录下寻找一个叫[servlet-name]-servlet.xml的配置文件，像上面的例子就会找/WEB-INF/web-servlet.xml进行加载。

- classpath*:spring.xml与classpath:spring.xml的区别

classpath:spring.xml表示仅加载classpath目录下的spring.xml
classpath*:spring.xml表示加载classpath目录及其子目录下，还有jar包中所有名为的spring.xml的文件

类控制器

类控制器是真正做事情的Handler，web配置好了之后，来看看处理器类是怎么写的：

```java
//LoginController.java
@Controller
@RequestMapping("/admin")
public class LoginController {
    @RequestMapping(value = "/login", method = RequestMethod.GET)
    @ResponseBody
    public String login() {
        return "login success";
    }
}
```

上面例子中请求的URL后面的路径为：`/admin/login`，即方法上的`@RequestMapping`注解是在类的注解基础上的，如果类上没有`@RequestMapping`注解，则方法上注解的路径就是绝对路径了。

另外注解`@ResponseBody`表示直接返回结果，否则，返回的字符串会被当成一个模板文件(如jsp)，具体内容后续文章再说。

spring的配置

通过web配置，可以把请求转发到我们定义的类控制器中处理，前提是web项目能够找到我们定义的类控制器，这就需要在spring配置文件中来指定。

这里的spring配置跟之前的差不多，无非就是让Spring能够找到我们用`@Controller`注解的Bean，另外还需要添加`<mvc:annotation-driven />`来支持SpringMVC注解

```xml
<!-- 支持SpringMVC注解 -->
<mvc:annotation-driven />
<!-- 扫描 LoginController 所在的包 -->
<context:component-scan base-package="com.test.springMVC"/>
```

SpringMVC拦截器两种方法：

配置文件加<mvc:interceptors>

1. implements HandlerInterceptor
2. extends HandlerInterceptorAdapter

   preHandle (controller之前执行)

   postHandle(controller执行完毕, ModelAndView返回之前执行)

   afterCompletion(最后执行)