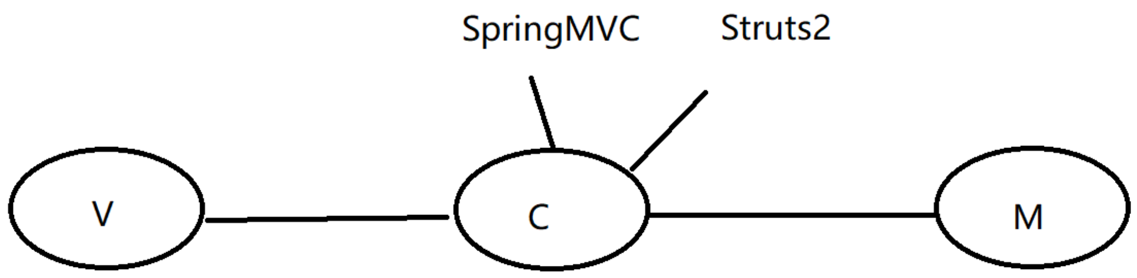
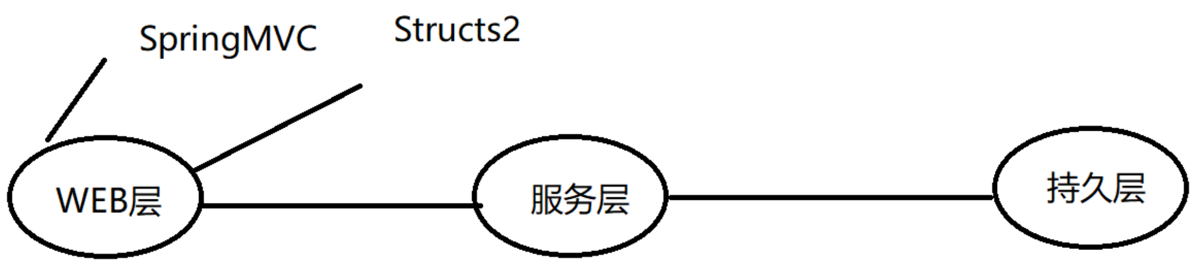
* 学习目标
  * 能够掌握MyBatisPlus的开发
  * 能够理解SpringMVC的概述
    * MVC:C
    * WEB三层架构：Web层
  * 能够掌握SpringMVC的HelloWorld开发
    * 添加依赖
    * 实现Controller---handleRequest---ModelAndView
    * springmvc.xml--bean
    * web.xml---DispatcherServlet
    * 视图解析器
  * 能够掌握SpringMVC的执行流程
    * request--DispatcherServlet--HandlerMapping---Handler---HandlerAdapter---ModelAndView
    * ViewResovler---View---Model---Response
  * 能够了解在SpringMVC中使用Servlet的形式进行开发
    * 不用xml配置和@WebServlet，不在tomcat容器，在SpringMVC容器

----

* 能够掌握MyBatisPlus的开发
  * 添加依赖
  * MyBatisSqlSessionFactoryBean
  * 实体：@TableName,@TableField
  * BaseMapper<实体>
    * CRUD，分页功能
* 能够理解SpringMVC的概述
  * Spring web mvc和Struts2都属于Web层（控制层）的框架





* 能够掌握SpringMVC的HelloWorld开发

```
* 案例一：implements Controller接口
* 添加依赖
<dependency>
   <groupId>junit</groupId>
   <artifactId>junit</artifactId>
   <version>4.12</version>
   <scope>test</scope>
</dependency>
<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-webmvc</artifactId>
   <version>5.2.2.RELEASE</version>
</dependency>
<dependency>
   <groupId>javax.servlet</groupId>
   <artifactId>javax.servlet-api</artifactId>
   <version>3.1.0</version>
   <scope>provided</scope>
</dependency>

* 编写控制器
```

```java
public class HelloController implements Controller {
    @Override
    public ModelAndView handleRequest(HttpServletRequest httpServletRequest,
        HttpServletResponse httpServletResponse) throws Exception {
        ModelAndView modelAndView=new ModelAndView();
        modelAndView.addObject("name","xiaohei");
        modelAndView.setViewName("hello.jsp");
        return modelAndView;
    }
}
```

* 在webapp下新建hello.jsp文件

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" isELIgnored="fal
<html>
<head>
    <title>Hello SpringMVC</title>
</head>
<body>
    <h1>HelloWorld ${name}</h1>
    <h1>123</h1>
    <h1>345</h1>
</body>
</html>
```

* springmvc.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-context.xsd
       http://www.springframework.org/schema/aop http://www.springframework.or
       <bean name="/hello.action" class="com.lg.controller.HelloController"/>
</beans>
```

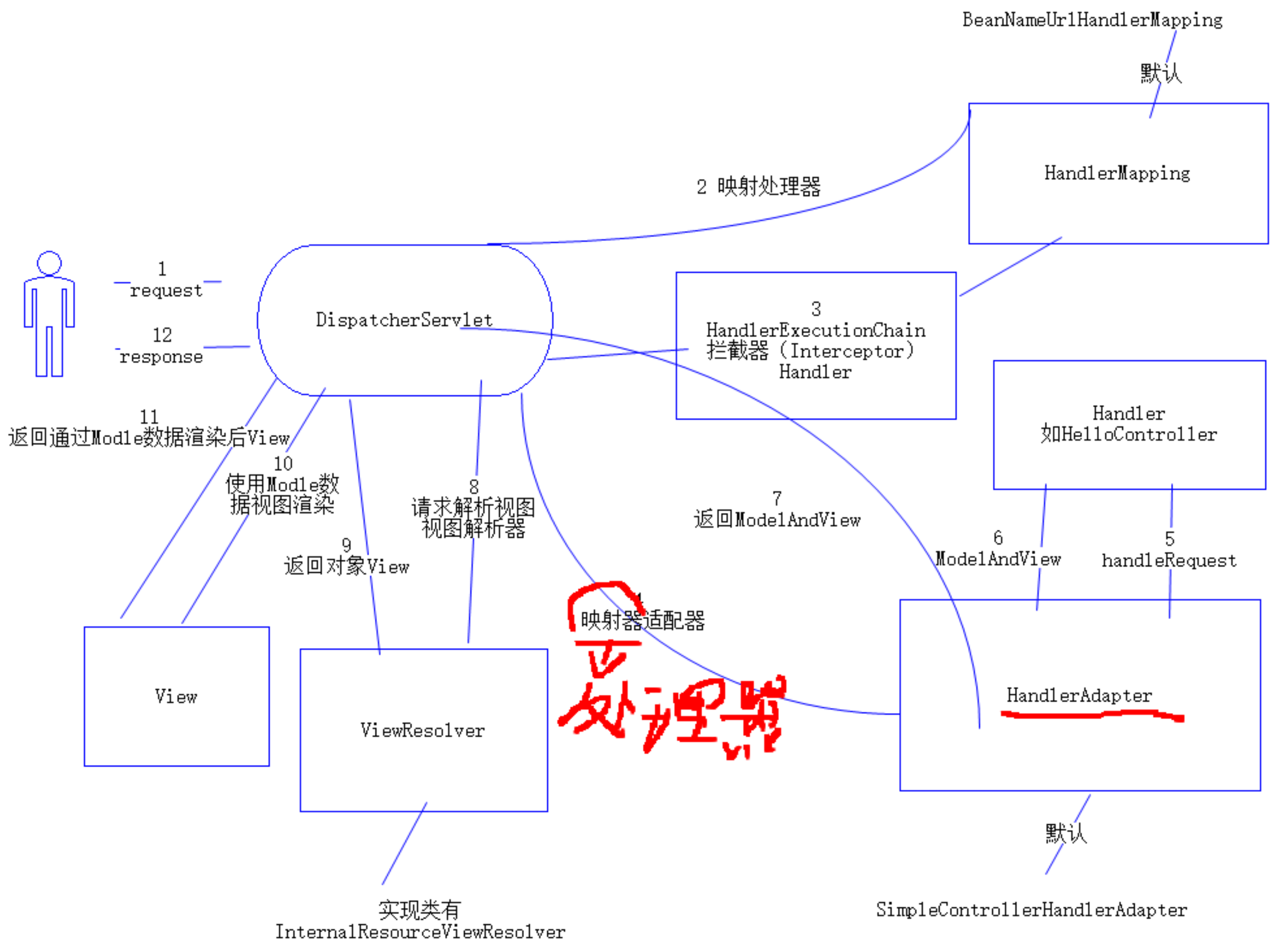
* web.xml

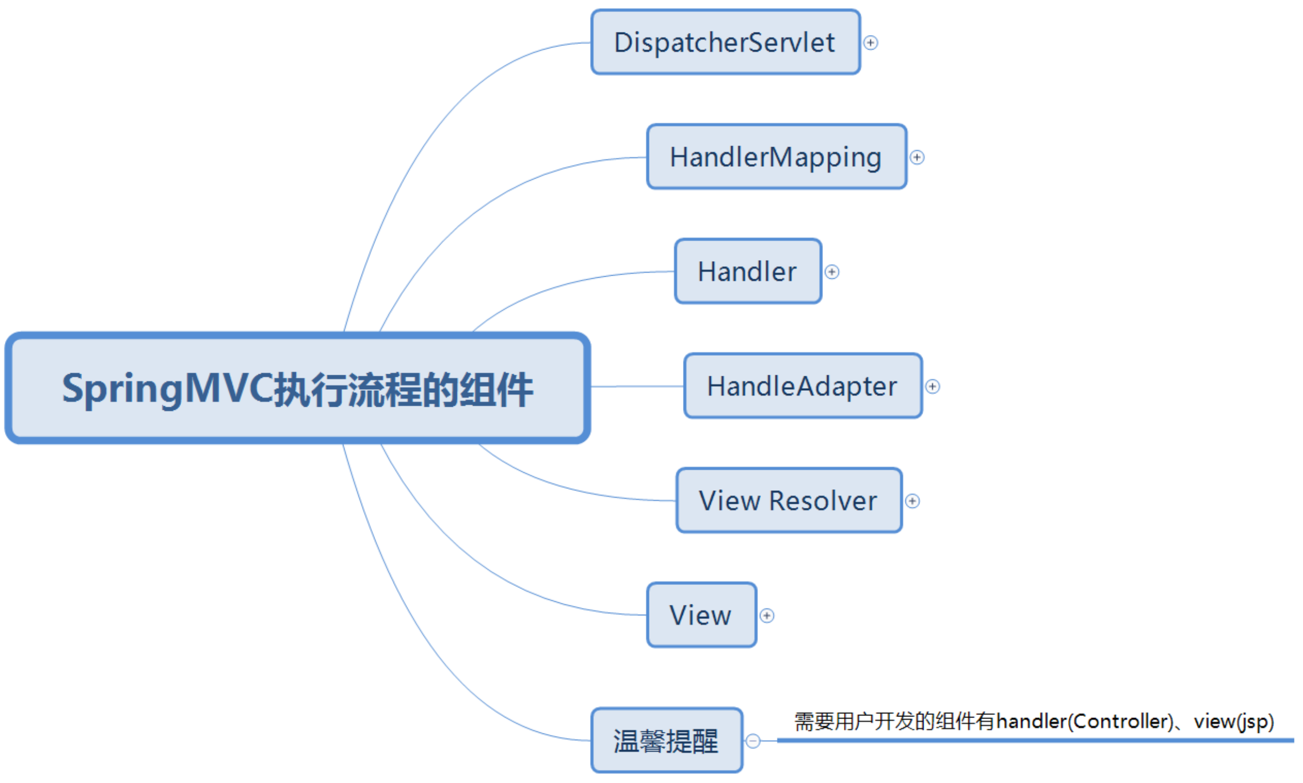
```xml
<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd" >

```

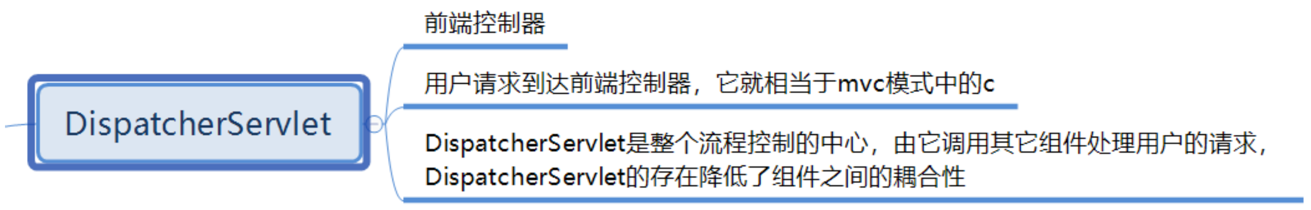
```
<web-app>
  <display-name>亮哥教育</display-name>
  <servlet>
    <servlet-name>springmvc</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-c
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:springmvc.xml</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>springmvc</servlet-name>
    <url-pattern>*.action</url-pattern>
  </servlet-mapping>
</web-app>

* 访问测试效果
  * http://localhost:8080/lgspringmvc/hello.action

* 案例二：（配置视图解析器-防止外界访问jsp页面）
  * 把hello.jsp放到/WEB-INF/jsps/目录下
  * 在springmvc.xml配置视图解析器
  <bean class="org.springframework.web.servlet.view.InternalResourceViewResolve
        <property name="prefix" value="/WEB-INF/jsps/"/>
        <property name="suffix" value=".jsp"/>
  </bean>
   * 修改HelloController代码
    * modelAndView.setViewName("hello");
   * 访问测试

* 案例三：通过注解的配置
@Controller
public class HelloController2 {
    @RequestMapping("/test1.action")
    public ModelAndView test1(){
        ModelAndView modelAndView=new ModelAndView();
        modelAndView.addObject("name","xiaobai666");
        modelAndView.setViewName("hello");
        return modelAndView;
    }
}
* 在springmvc.xml配置
 * <context:component-scan base-package="com.lg"/>
* 访问测试
```

* 能够掌握SpringMVC的执行流程

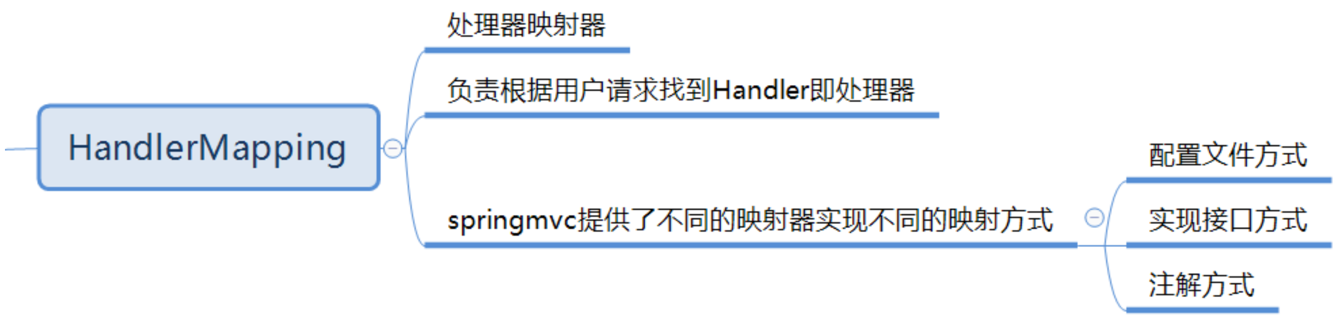
```
* 用户发送请求至前端控制器DispatcherServlet
* DispatcherServlet收到请求URL调用HandlerMapping处理器映射器（默认是BeanNameUrlHan
* 处理器映射器根据请求url找到具体的处理器，生成处理器对象及处理器拦截器(如果有则生成)-
* DispatcherServlet通过HandlerAdapter处理器适配器调用处理器（默认是SimpleControlle
* 执行处理器(Controller，也叫后端控制器)。
* Controller执行完成返回ModelAndView
* HandlerAdapter将controller执行结果ModelAndView返回给DispatcherServlet
* DispatcherServlet将ModelAndView传给ViewReslover视图解析器
* ViewReslover解析后返回具体View
* DispatcherServlet对View进行渲染视图（即将模型数据填充至视图中）。
* DispatcherServlet响应用户
```


BeanNameUrlHandlerMapping
默认
HandlerMapping
2 映射处理器
1 request
12 response
DispatcherServlet
3 HandlerExecutionChain 拦截器（Interceptor） Handler
Handler 如HelloController
11 返回通过Modle数据渲染后View
10 使用Modle数据视图渲染
8 请求解析视图 视图解析器
7 返回ModelAndView
6 ModelAndView
5 handleRequest
9 返回对象View
4 映射器适配器
处理器
View
ViewResolver
HandlerAdapter
默认
实现类有 InternalResourceViewResolver
SimpleControllerHandlerAdapter

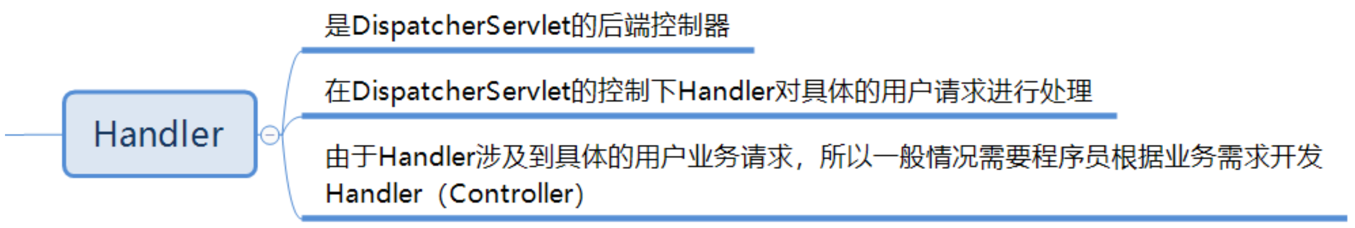

SpringMVC执行流程的组件
DispatcherServlet
HandlerMapping
Handler
HandleAdapter
View Resolver
View
温馨提醒
需要用户开发的组件有handler(Controller)、view(jsp)



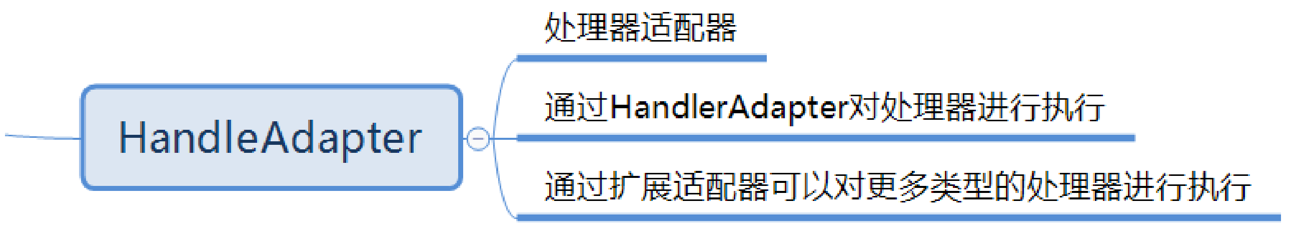
- DispatcherServlet
  - 前端控制器
  - 用户请求到达前端控制器，它就相当于mvc模式中的c
  - DispatcherServlet是整个流程控制的中心，由它调用其它组件处理用户的请求，DispatcherServlet的存在降低了组件之间的耦合性



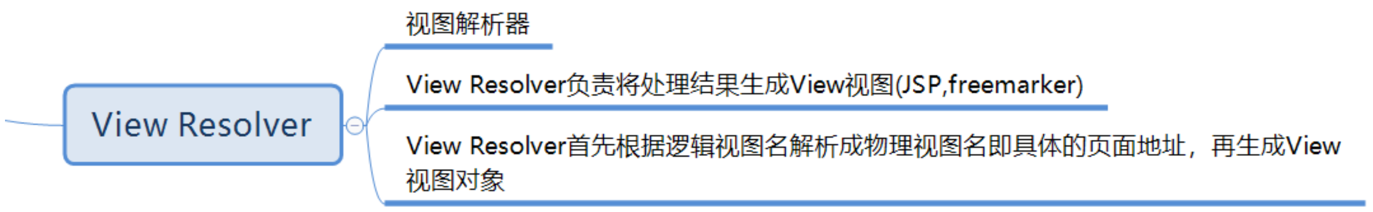
- HandlerMapping
  - 处理器映射器
  - 负责根据用户请求找到Handler即处理器
  - springmvc提供了不同的映射器实现不同的映射方式
    - 配置文件方式
    - 实现接口方式
    - 注解方式



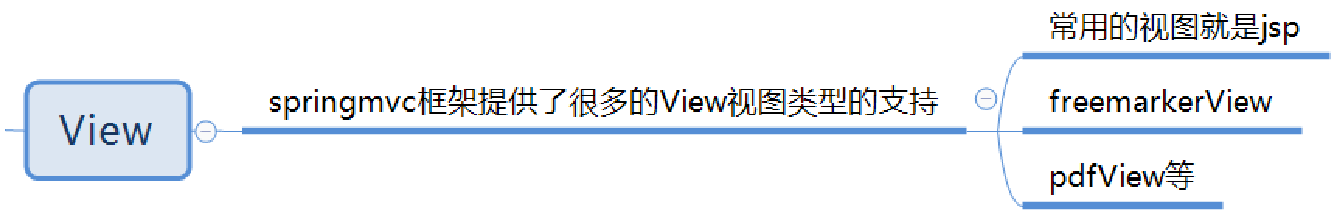
- Handler
  - 是DispatcherServlet的后端控制器
  - 在DispatcherServlet的控制下Handler对具体的用户请求进行处理
  - 由于Handler涉及到具体的用户业务请求，所以一般情况需要程序员根据业务需求开发Handler（Controller）



- HandleAdapter
  - 处理器适配器
  - 通过HandlerAdapter对处理器进行执行
  - 通过扩展适配器可以对更多类型的处理器进行执行



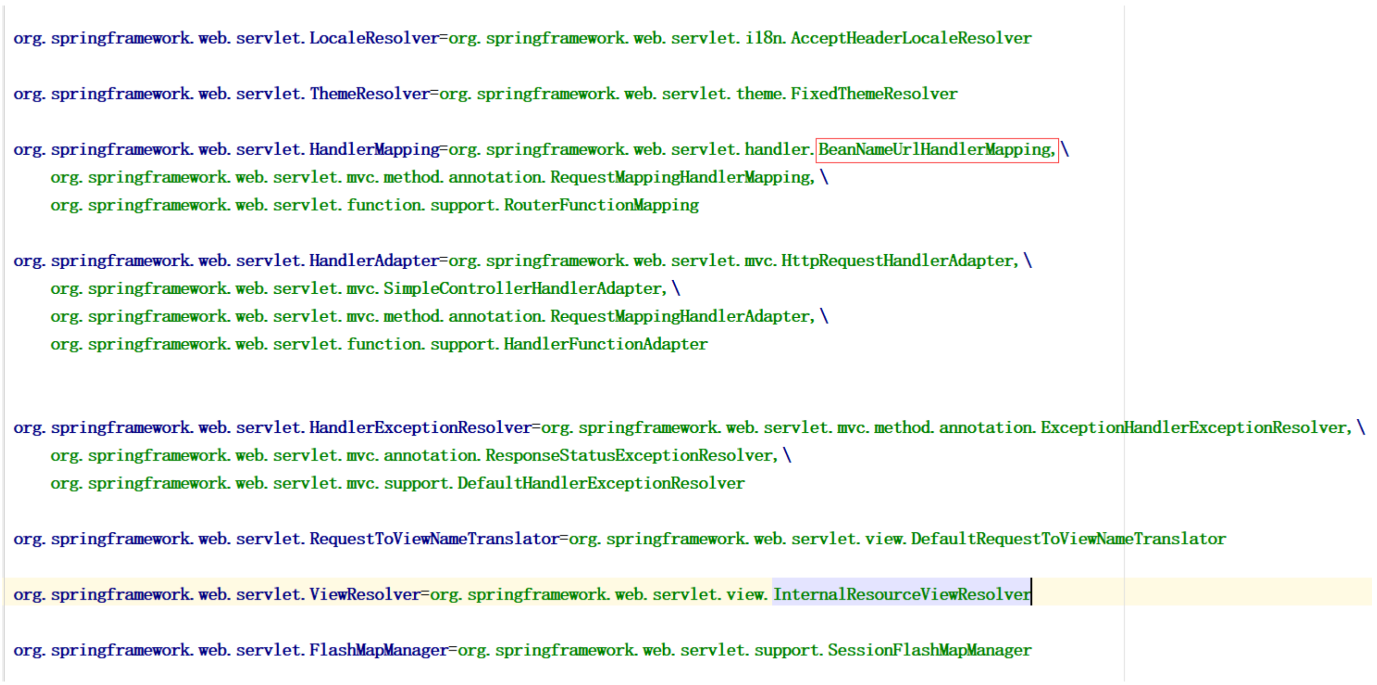
- View Resolver
  - 视图解析器
  - View Resolver负责将处理结果生成View视图(JSP,freemarker)
  - View Resolver首先根据逻辑视图名解析成物理视图名即具体的页面地址，再生成View视图对象



- View
  - springmvc框架提供了很多的View视图类型的支持
    - 常用的视图就是jsp
    - freemarkerView
    - pdfView等

* 查看框架默认加载的组件

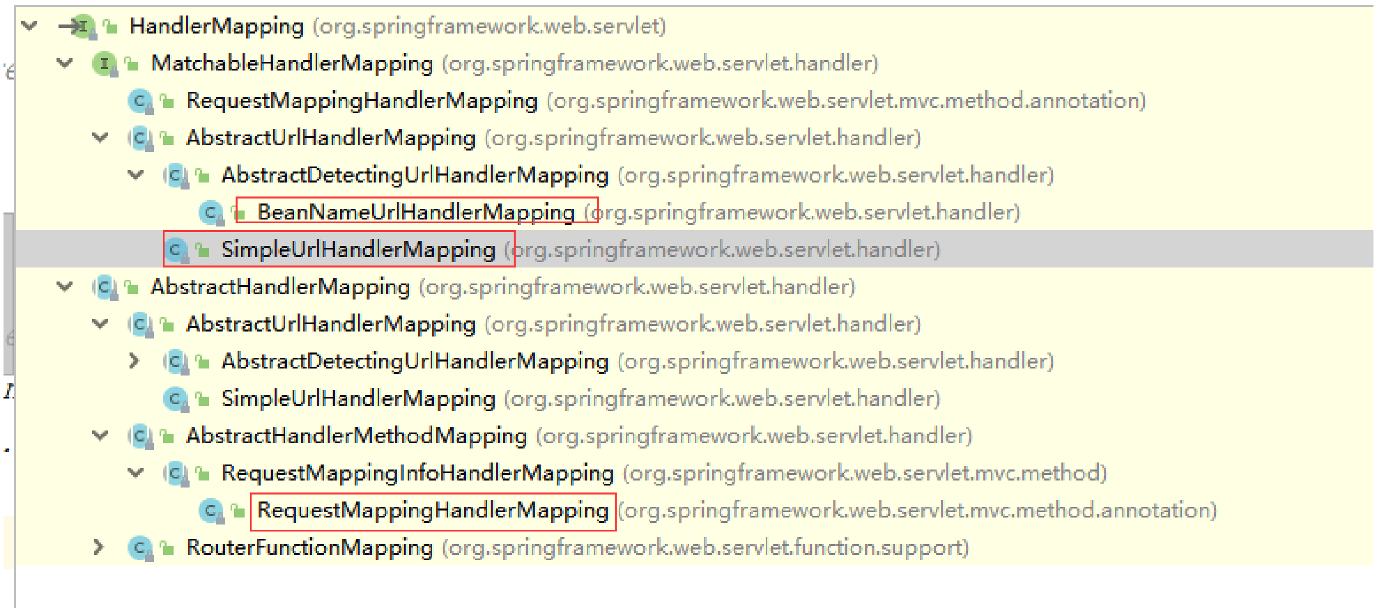
* 搜索DispatcherServlet.properties



```
org.springframework.web.servlet.LocaleResolver=org.springframework.web.servlet.i18n.AcceptHeaderLocaleResolver

org.springframework.web.servlet.ThemeResolver=org.springframework.web.servlet.theme.FixedThemeResolver

org.springframework.web.servlet.HandlerMapping=org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping,\
	org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping,\
	org.springframework.web.servlet.function.support.RouterFunctionMapping

org.springframework.web.servlet.HandlerAdapter=org.springframework.web.servlet.mvc.HttpRequestHandlerAdapter,\
	org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter,\
	org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter,\
	org.springframework.web.servlet.function.support.HandlerFunctionAdapter


org.springframework.web.servlet.HandlerExceptionResolver=org.springframework.web.servlet.mvc.method.annotation.ExceptionHandlerExceptionResolver,\
	org.springframework.web.servlet.mvc.annotation.ResponseStatusExceptionResolver,\
	org.springframework.web.servlet.mvc.support.DefaultHandlerExceptionResolver

org.springframework.web.servlet.RequestToViewNameTranslator=org.springframework.web.servlet.view.DefaultRequestToViewNameTranslator

org.springframework.web.servlet.ViewResolver=org.springframework.web.servlet.view.InternalResourceViewResolver

org.springframework.web.servlet.FlashMapManager=org.springframework.web.servlet.support.SessionFlashMapManager
```

```
* 测试案例一
 * 在springmvc.xml
  <bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMap
  <bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMap
 * 会覆盖默认的HandlerMapping和HandlerAdapter
  * 访问：http://localhost:8080/lgspringmvc/test1.action和之前一样
  * 访问：http://localhost:8080/lgspringmvc/hello.action会出现404

* 测试案例二
 * 在springmvc.xml继续添加
  <bean class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMappin
  <bean class="org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapt
 * 访问：http://localhost:8080/lgspringmvc/hello.action和之前一样

* 案例三：
 * 在springmvc.xml
  <bean id="mController" name="/hello.action" class="com.lg.controller.HelloCon
  <bean class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping"
        <property name="mappings">
            <props>
                <prop key="/a.action">mController</prop>
                <prop key="/b.action">mController</prop>
            </props>
        </property>
    </bean>
   * 访问：http://localhost:8080/lgspringmvc/a.action或b.action
```

* HandlerMapping继承结构



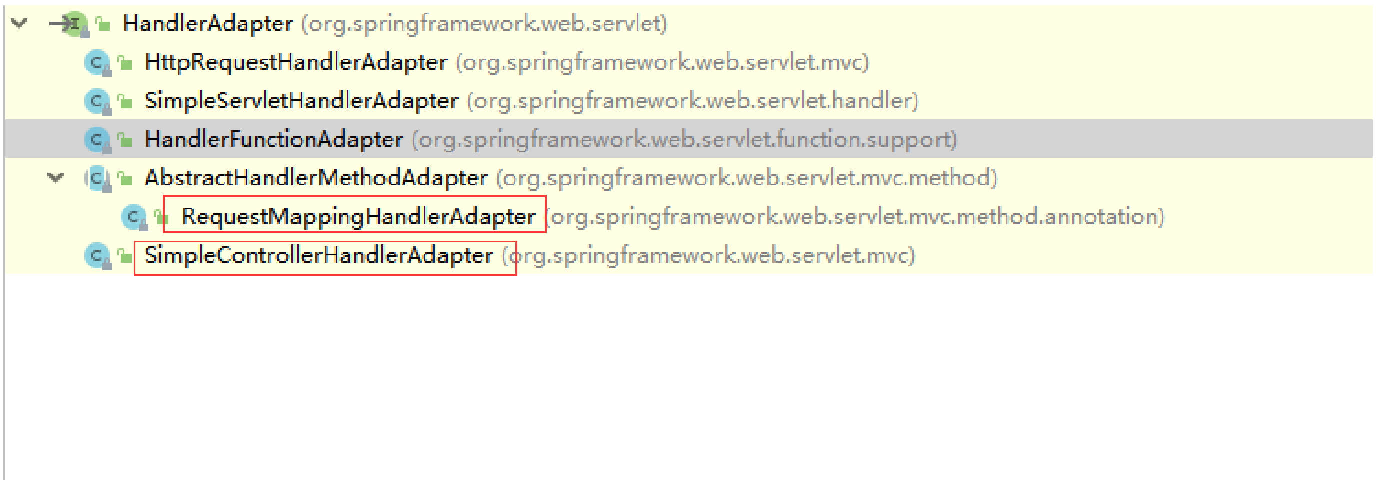
```
HandlerMapping (org.springframework.web.servlet)
  MatchableHandlerMapping (org.springframework.web.servlet.handler)
    RequestMappingHandlerMapping (org.springframework.web.servlet.mvc.method.annotation)
    AbstractUrlHandlerMapping (org.springframework.web.servlet.handler)
      AbstractDetectingUrlHandlerMapping (org.springframework.web.servlet.handler)
        BeanNameUrlHandlerMapping (org.springframework.web.servlet.handler)
      SimpleUrlHandlerMapping (org.springframework.web.servlet.handler)
  AbstractHandlerMapping (org.springframework.web.servlet.handler)
    AbstractUrlHandlerMapping (org.springframework.web.servlet.handler)
      AbstractDetectingUrlHandlerMapping (org.springframework.web.servlet.handler)
      SimpleUrlHandlerMapping (org.springframework.web.servlet.handler)
    AbstractHandlerMethodMapping (org.springframework.web.servlet.handler)
      RequestMappingInfoHandlerMapping (org.springframework.web.servlet.mvc.method)
        RequestMappingHandlerMapping (org.springframework.web.servlet.mvc.method.annotation)
    RouterFunctionMapping (org.springframework.web.servlet.function.support)
```

* HandlerMapping接口核心方法

* 返回此请求的处理程序和所有拦截器

* HandlerExecutionChain getHandler(HttpServletRequest request) throws Exception;

* HandlerAdapter继承结构



```
HandlerAdapter (org.springframework.web.servlet)
  HttpRequestHandlerAdapter (org.springframework.web.servlet.mvc)
  SimpleServletHandlerAdapter (org.springframework.web.servlet.handler)
  HandlerFunctionAdapter (org.springframework.web.servlet.function.support)
  AbstractHandlerMethodAdapter (org.springframework.web.servlet.mvc.method)
    RequestMappingHandlerAdapter (org.springframework.web.servlet.mvc.method.annotation)
  SimpleControllerHandlerAdapter (org.springframework.web.servlet.mvc)
```

* 核心方法：

* 使用给定的处理程序来处理此请求

* ModelAndView handle(HttpServletRequest request,

  HttpServletResponse response, Object handler) throws Exception;

* 查看SimpleControllerHandlerAdapter的handle方法源码
  * 在这里发现： ((Controller) handler).handleRequest(request, response);
  * 这个就是我们实现的接口

* 能够了解在SpringMVC中使用Servlet的形式进行开发
  * 不用在web.xml或者@WebServlet里面配置，Servlet不在tomcat容器里，而是在SpringMVC容器里

```
* 代码
public class HelloServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) thro
        resp.getWriter().write("HelloServlet");
    }
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) thr
        doGet(req,resp);
    }
}

* 配置
 <bean name="/hello2.action" class="com.lg.controller.HelloServlet"/>
 <bean class="org.springframework.web.servlet.handler.SimpleServletHandlerAdapt
* 访问: http://localhost:8080/lgspringmvc/hello2.action
```