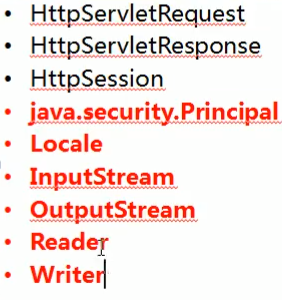
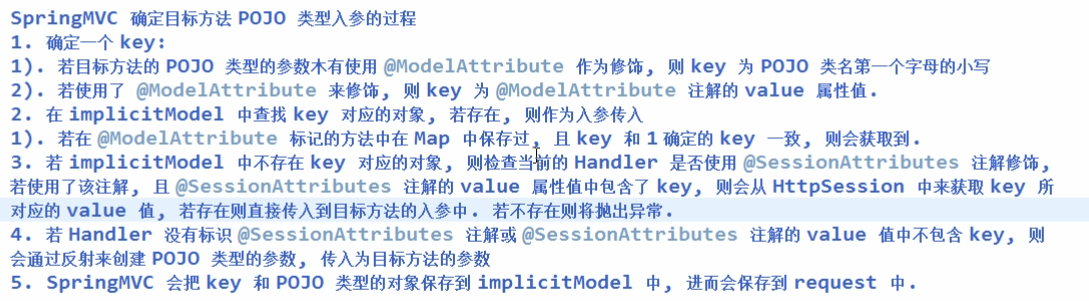
### SpringMVC

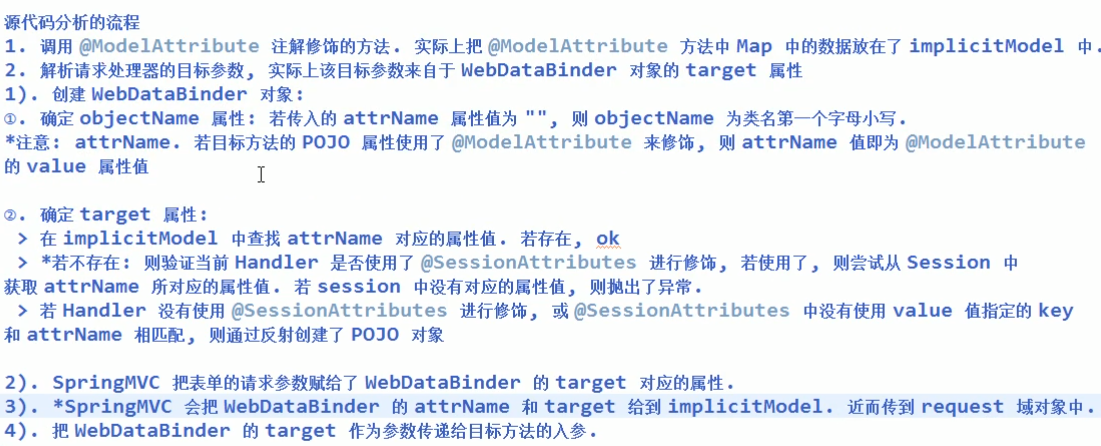
1. 基础步骤
   1. 加入jar包
   2. 在web.xml中配置DispatcherServlet
   3. 加入SpringMVC的配置文件
   4. 编写处理请求的处理器，并标识为处理器
   5. 编写视图
2. RequestMapping除了修饰方法还可以修饰类
   1. 类定义处：提供初步请求映射信息，相对于WEB应用的根目录
   2. 方法处：提供进一步的细分映射信息，相对于类定义处的url。若类定义处未标注RequestMapping，则方法标记处的URL相对于web 的根目录
3. 请求体 
4. REST



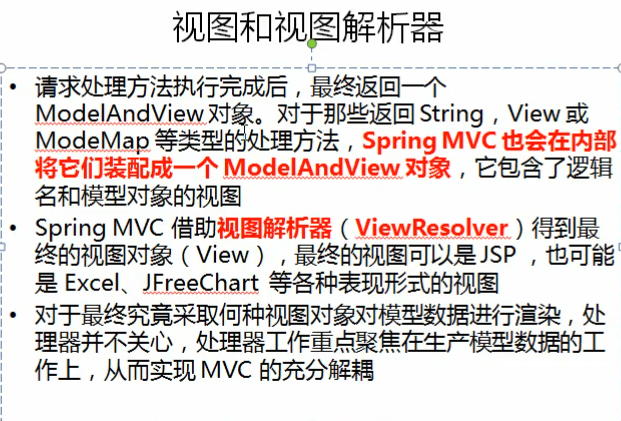
1. Rest风格的URL
   1. 以CURD为例
   2. 新增：/order POST
   3. 修改：/order/1 PUT
   4. 获取：/order/1 GET
   5. 删除：/order/1 DELETE
2. 如果需要发送DELETE和PUT请求需要添加HiddentHttpMethodFilter过滤器
3. 执行顺序：请求--》控制器方法--》按照配置文件配置的过滤器顺序
4. Handler方法可以支持哪些原生ServletAP 
5. @ModelAttribute 注解修饰方法流程：
   1. 从数据库中取出对象，把对象放入到map中，键为：user
   2. SpringMVC从Map中取出User对象，并把表单的请求参数赋给该User对象的对应属性
   3. SpringMVC把上述对象传入目标方法的参数
   4. 注意：在@ModelAttribute修饰的方法中，放入到map时的键需要和目标方法入参类型的第一个字母小写的字符串一致

源码分析





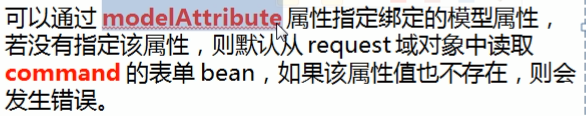
1. 视图和视图解析器：每一个请求都会创建一个新的视图



1. i18n必须要经过SpringMVC的控制器请求方法
2. 如果是Rest风格的URL想要使用重定向则需要使用全路径

|  |
| --- |
| *//Rest* @RequestMapping(value = "testMethodDELETE/{id}", method = RequestMethod.*DELETE*) public String testMethodDELETE(@PathVariable("id") Integer id) {  System.*out*.println("testMethodDELETE::" + id);  return "redirect:/views/hello/hello.jsp"; } |
| Spring标签获取 <spring:message code="i18n.username"/> <spring:message code="i18n.password"/> <br> I18n就需要，最好使用fmt标签获取，因为如果是redirect请求使用Spring标签获取会报错 使用fmt标签获取 <fmt:message key="i18n.username"></fmt:message> <fmt:message key="i18n.password"></fmt:message> |

1. 使用SpringMVC的表单回显时



1. 处理静态资源



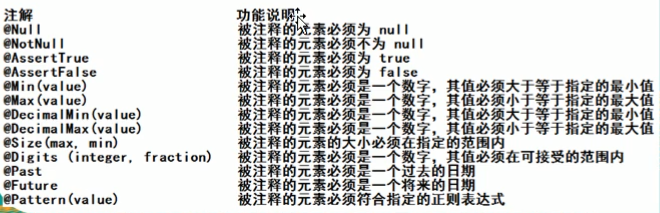
1. 类型转换器

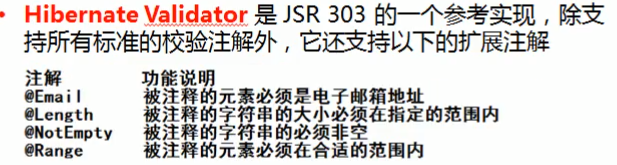


1. 关于mvc：annotation-driven的信息



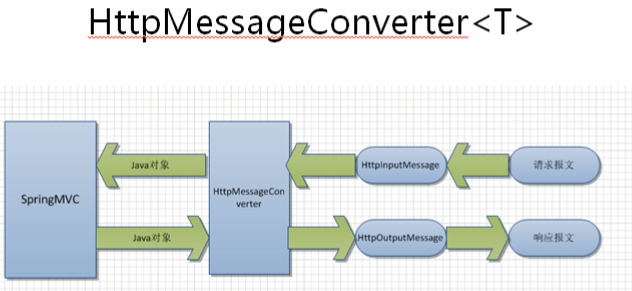
1. JSR 303，数据校验







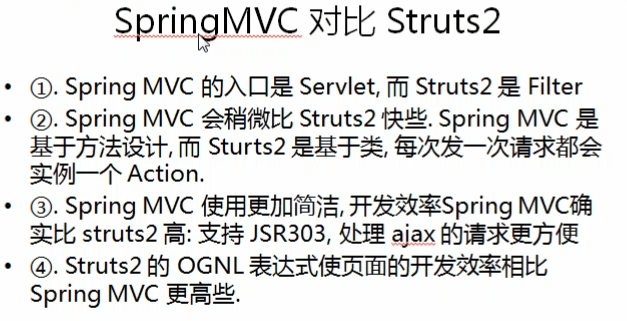
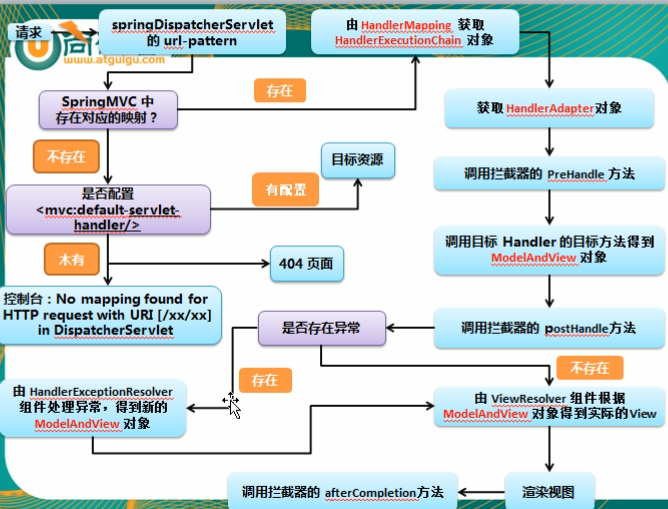
1. JSON包装实现原理



1. 如果使用getResourceAsStream获取不到资源则需要在项目的out目录下添加
2. 异常处理



1. SpringMVC运行流程 和 SpringMVC对比Struts2



1. Spring和SpringMVC





