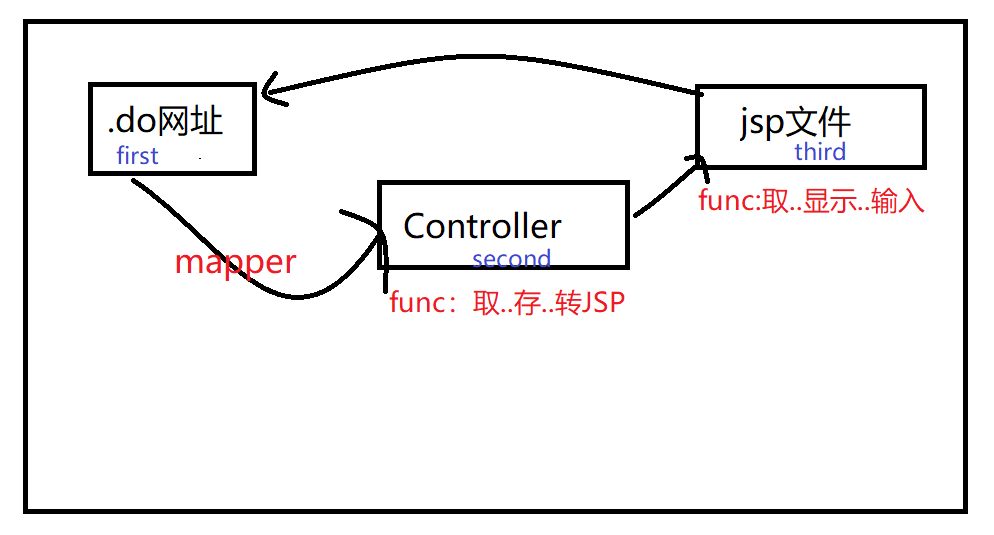
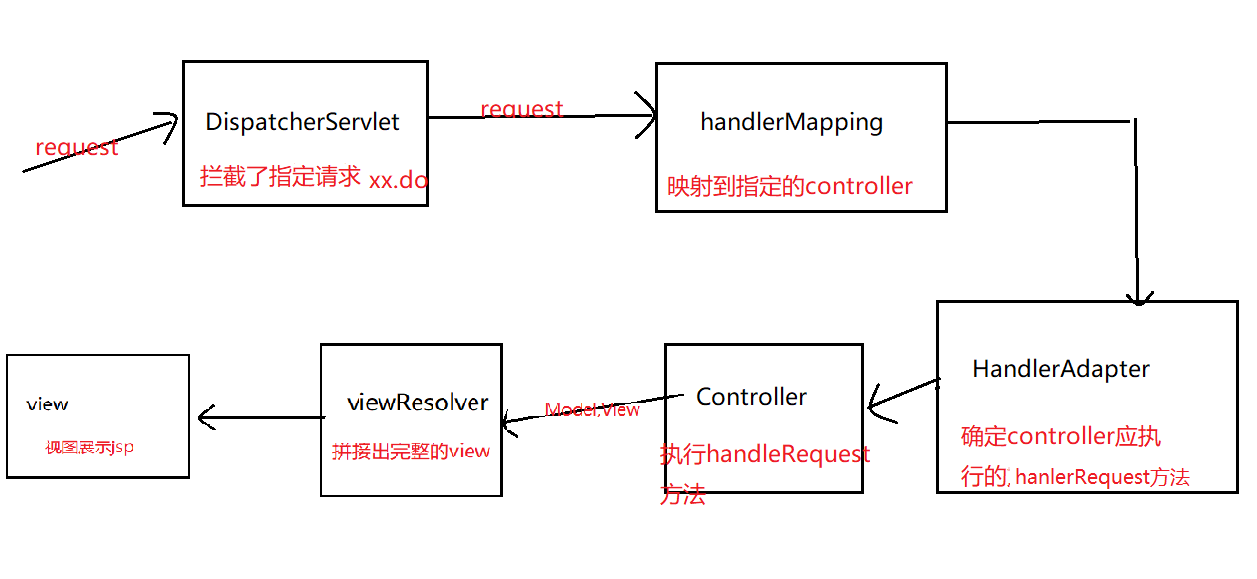
总体流程：



注意：

DispatcherServlet·拦截到一个请求，将这个请求使用HanlerMapping映射到Controller上，再与HandlerAdapter对照确定Controller将执行的handleRequest方法，这个handleRequest的方法或会返回一个ModelAndView，如果没有，那么就要自己完成转发工作以及将数据setAttribute。如果会返回出ModelAndView，那么通过setView,addObject的modelAndView会存起来，model可以被EL表达式拿出(${})，而view会再进一步交给ViewResolver，视图解析器会将view拼接上前缀，后缀，然后转到这个拼接完的url去。



一、步骤：

1. 导入jar包：

aopalliance-1.0.0.jar

aspectjweaver-1.8.10.jar

commons-beanutils-1.8.0.jar

commons-logging-1.2.jar

javax.servlet.jsp.jstl-api-1.2.2.jar

jstl-impl-1.2.jar

spring-aop-4.3.0.RELEASE.jar

spring-beans-4.3.0.RELEASE.jar

spring-context-4.3.0.RELEASE.jar

spring-core-4.3.0.RELEASE.jar

spring-expression-4.3.0.RELEASE.jar

spring-web-4.3.0.RELEASE.jar

spring-webmvc-4.3.0.RELEASE.jar

1. Web.xml 配置DisapatcherServlet , CharacterEncodingFilter。

注意：DispatcherServlet都有一个作用域受限的WebApplicationContext,它继承自根ApplicationContext,在默认情况下，WebApplicationContext会在/WEB-INF目录中查找名为{servletName}-servlet.xml的配置文件，但是可以在<servlet-class>下重新配置<init-param>的contextConfigLocation的value来修改要去查找的配置文件的名称。

1. 配置

方式1：使用xml配置：

1. 配置hanlerMapping
   1. BeanNameUrlHandlerMapping

Controller的name属性即为访问路径，例如：“/register.do”

* 1. SimpleUrlHandlerMapping

访问路径在mappings中的properties中作为key，而Controller的id属性将作为key对应的value,当访问路径时，就map到controller的id上。

* 1. ControllerClassNameMapping

类名作为访问路径,可设置propety: caseSensitive, 大小写敏感

1. 配置hanlerAdapter
   1. SimpleControllerHandlerAdapter：

自定义的Controller实现Controller接口，重写handleRequest方法，返回值为ModelAndView

将数据存入(addObject),设置view(setViewName)，这个viewname将继续传递到ViewResolver中，被拼接为一个完整的页面路径。

* 1. HttpRequestHandlerAdapter

自定义的Controller实现HttpRequestHandler接口，重写handleRequest,返回值为void，自己存数据(xxx.setAttribute),自己转页面(转发/重定向)

1. 配置自定义的Controller

依照上面的设置属性(name / id)，而这个类也继承接口(HttpRequestHandler / Controller)。

1. 配置viewResolver

设置property prefix前缀, suffix 后缀的value

例如：…prefix “/WEB-INF/views/”

…suffix “.jsp”

方式2：使用注解配置：

1. 配置注解扫描：<context:comphonent-scan >配置base-package包
2. 配置RequestMappingHandlerMapping
3. 配置RequsetMappingHandlerAdapter
4. 配置viewResolver

设置property prefix前缀, suffix 后缀的value

例如：…prefix “/WEB-INF/views/”

…suffix “.jsp”

1. 如果是注解配置的话，创建一个类什么都不需要继承，加上注解@Controller，再使用@RequestMapping注解这个类可以给这个类设置一个根路径，那么在访问时就要加上这个根路径再带上这个类下的其他方法的@RequestMapping的路径作为访问路径。

对应的获取表单参数也可以有多种：

1. 多值获取参数，包括用来存参数的Request,都可以在参数中获取
2. @RequestMapping("/registerAnno1.do")
3. **public** String collectionParamAndReturnPage(HttpServletRequest request, String username, String password, String[] hobbies) **throws** InvocationTargetException, IllegalAccessException {
4. System.out.println("username:" + username);
5. System.out.println("password:"+ password);
6. System.out.println("hobbies:"+ hobbies);
8. HashMap<String,Object> hashMap = **new** HashMap();
9. hashMap.put("username",username);
10. hashMap.put("password",password);
11. hashMap.put("hobbies",hobbies);
12. User user = **new** User();
13. BeanUtils.populate(user,hashMap);
14. request.setAttribute("username",username);
15. request.setAttribute("password",password);
16. request.setAttribute("hobbiew",hobbies);
17. request.setAttribute("user",user);
18. **return** "user/userListAnnoJsp";
19. }

其中的jsp输入为 username , password , hobbies

2. 直接聚合成一个model，在参数中获取到这个model

1. @RequestMapping(value = "/registerAnno2.do" )
2. **public** String collectionParamAndReturnPage(HttpServletRequest request, User o) {
3. request.setAttribute("user",o);
4. System.out.println("ooooo"+o.getClass());
5. **return** "user/userListAnnoJsp";
6. }

其中的jsp输入为 username , password , hobbies

3. 再聚合成一个modelPOJO对象，同样在参数中获取

1. @RequestMapping(value = "/registerAnno3.do" )
2. **public** String collectionParamAndReturnPage(HttpServletRequest request, UserVoPOJO userVoPOJO) {
3. request.setAttribute("userVoPOJO", userVoPOJO);
4. **return** "user/userListAnnoJsp";
5. }

其中的jsp输入为 user.username , user.password , user.hobbies

而UserPOJO包含了user成员变量名称。这个要对应上。

1. 可以聚合一个modelList对象，不管是list/数组皆可
   1. @RequestMapping(value = "/registerAnno5.do" )
   2. **public** String collectionParamAndReturnPage(HttpServletRequest request, UserVoList userVoList) {
   3. request.setAttribute("userVoList", userVoList);
   4. **return** "user/userListAnnoJsp";
   5. }

其中的jsp输入为的hobbies[0]~~~~hobbies[n]

在输出显示上：

1. <!-- model存下来的string对应上，**var**用于遍历 -->
2. <c:forEach items="${userList}" **var**="user">
3. 用户名：${user.username}, 密码：${user.password}, 爱好：${user.hobbies}
4. </c:forEach>

重定向，转发：

这两种方式是用来转访问路径的，不是用来转到/WEB-INF/xxx.jsp的，也就是转到网址的路径上。可以设置参数注解：RequestParam，其中的required代表必须，value与参数name要对应上，说明在转发/重定向时要带上这个参数才能完成。

1. @RequestMapping("/forward.do")
2. **public** String forward(@RequestParam(required = **true**,value = "uid",defaultValue = "20") **int** uid){
3. **return** "forward:/user/userList.do";
4. }
6. @RequestMapping("/redirect.do")
7. **public** String redirect(){
8. **return** "redirect:/user/userList.do";
9. }