课程安排：

Springmvc框架原理（掌握）

前端控制器、处理器映射器、处理器适配器、视图解析器

注解的处理器映射器、处理器适配器（掌握）

Springmvc注解开发：

常用的注解学习

参数绑定（简单类型、pojo、集合类型）

自定义参数绑定（掌握）

Springmvc和struts2区别

数据回显

上传图片

Json数据交互

RESTful支持

拦截器

一、springmvc框架

第一步：发起请求到前端控制器（DispatcherServlet）

第二步：前端控制器请求HandlerMapping 查找Handler

可以根据xml配置、注解进行查找

第三步：处理器映射器HandlerMapping 向前端控制器返回Handler

第四步：前端控制器调用处理器适配器去执行Handler

第五步：处理器适配器去执行Hanler

第六步：Handler执行完成给适配器返回ModelAndView

第七步：处理器适配器向前端控制器返回ModelAndView

ModelAndView 是springmvc框架的一个底层对象，包括Model和View

第八步：前端控制器请求视图解析器去进行视图解析

根据逻辑视图名解析成真正的视图（jsp）

第九步：视图解析器向前端控制器返回view

第十步：前端控制器进行视图渲染

视图渲染将模型数据（在ModelAndView对象中）填充到request域

第十一步：前端控制器向用户响应结果。

组件：

1. 前端控制器 DispatcherServlet

作用:接受请求，响应结果，相当于转发器

1. 处理器映射器HandlerMapping

作用：根据请求的url 查找Handler

1. 处理器适配器（HandlerAdapter）

作用：按照特定规则（HandlerAdapter要求的规则）去执行Handler

1. 视图解析器view resolver

作用：进行视图解析，根据逻辑视图名解析成真正的视图

1. 视图（需开发）
2. 处理器Hadler（需开发）
3. springmvc和mybatis整合

第一步：整合dao层

Mybatis和spring整合，通过spring管理mapper接口。

使用mapper的扫描器自动扫描mapper接口在spring中进行注册。

第二步：整合service层

通过spring管理service接口。

使用配置方式将service接口配置在spring配置文件中。

事务分层。

第三步：无需整合springmvc，因为springmvc是spring的模块。

1. 注解详解

@RequestMapper

1. url映射

定义controller方法对应的url，进行处理器映射使用。

1. 窄化请求映射
2. 限制http请求方法
3. controller 方法的返回值
4. 参数绑定
5. spring参数绑定

从客户端请求key/value数据，经过参数绑定，将key/value数据绑定到controller方法的形参上。Springmvc中，接收页面提交的数据是通过方法形参来接收。而不是在controller类定义成员变量接收！！！

1. 默认支持的类型

2.1HTTPServletRequest

2.2HTTPServletResponse

2.3HttpSession

2.4Model/ModelMap

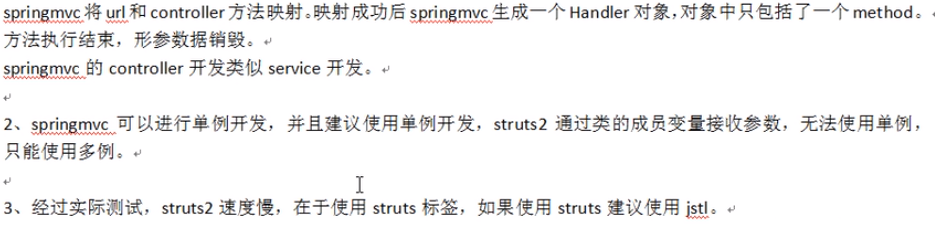
1. 简单类型
2. Pojo绑定

页面中input的name和controller的pojo形参中的属性名称一致，将页面中数据绑定到pojo。

1. springmvc和strut2的区别

5.1

springmvc基于方法开发的，struts2基于类开发的。



复习：

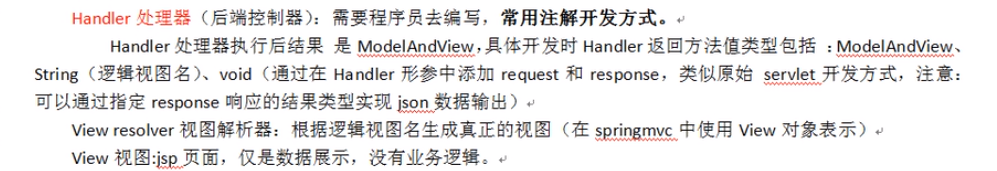
Springmvc框架：

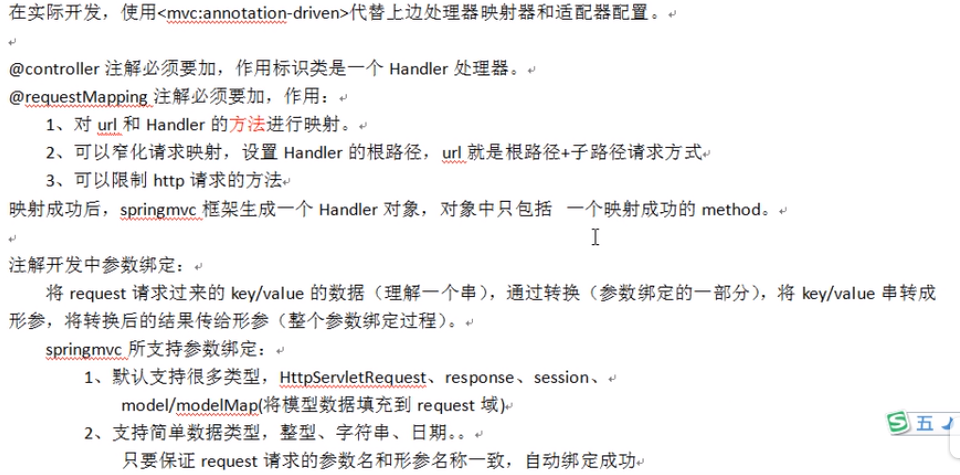
DispatcherServlet前端控制器：接收request，进行response

HandlerMapper处理器映射器：根据url查找Handler。（可以通过xml配置方式，注解方式）

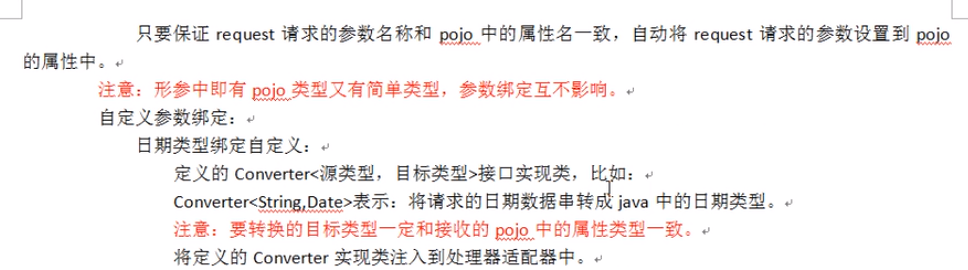
HandlerAdapter处理器适配器：根据特定规则去执行Handler，编写Handler时需要按照HandlerAdapter的要求去编写。

Handler处理器（后端控制器）：需要程序员去编写，常用注解开发方式。





支持Pojo类型



1. 高级知识
2. 包装类型pojo参数绑定
3. 集合类型绑定
   1. 数组绑定
   2. List绑定
   3. Map绑定
4. 服务器校验
5. 分组校验
6. 数据回显

5.1



