@Controller

这个也没什么说头的，就是只要是个控制器类，就给我加上，表明这是个控制器类吗，不然你让配置文件怎么找

@RequetMappig

value属性：标明映射路径

写法：value＝“ /... “

若只有该属性也可直接写为“ /... “

method属性：标明方法处理哪些HTTP请求方式

写法：method=RequestMethod.POST

...

conumes属性：标明处理请求的提交内容类型

写法：consumes="application/json"

表示该方法仅仅处理“application / json “的请求

produces属性：标明返回的内容类型

写法：produces="appliacation/json"

表示该方法仅仅处理"application/json"的请求,

返回的内容类型为application/json

（比consumes更强制？）

params属性：指定request中必须包含某些参数值时才让该方法处理

写法：params="myParam=myValue"

表示该方法仅仅处理名为“myParam “,值为“myValue “的请求

headers属性：指定request中必须包含某些指定的header值,才让该方法处理

写法：headers="Referer=http://www.jlb.com"

表示该方法仅仅处理request的header中包含指定“Referer “请求头

和对应值为"http://www.jlb.com"的请求

注意：一个控制器类中，@ModelAttribute修饰的方法总是会先于映射方法执行

Model,ModelMap,ModelAndView的区别

对于传入为Map型时，Model，ModelMap没什么区别，ModelMap只是指定了只存Map型，

而ModelAndView除了存入Model,还要存入View

接入门笔记那个实例来讲讲这三者的具体用法

Model

```
@ModelAttribute
public void userModel(User user,Model model){
    User u=new User();
    u.setName(user.getName());
    u.setAge(user.getAge());
    u.setPwd(user.getPwd());
    model.addAttribute("user", u);
}

@RequestMapping(value="save")
public String save(Model model){
    User user=(User)model.asMap().get("user");
    user.setName("小叶子");
    return "detail";
}
```

首先呢，@ModelAttribute修饰的方法会先运行，我们看到的是该方法将获取的用户信息放入model，而我们的映射方法除了跳转页面，还干了一件事，就是获取该model，并修改了一个属性，具体结果是不论姓名设置什么，最后都是“小叶子”，其它两项属性根据设置而定

ModelMap

```
@ModelAttribute
public void userModel(User user,ModelMap modelMap){
    User u=new User();
    u.setName(user.getName());
    u.setAge(user.getAge());
    u.setPwd(user.getPwd());
    modelMap.addAttribute("user", u);
}

@RequestMapping(value="save")
public String save(ModelMap modelMap){
    User user=(User)modelMap.get("user");
    user.setName("小叶子");
    return "detail";
}
```

和Model对比一下确实没什么太大的区别吧，可以看映射方法我们获取ModelMap时不需要asMap()这个方法了，就是因为ModelMap已经指定了存的是Map型

ModelAndView

```
@ModelAttribute
public void userModel(User user,ModelAndView mv){
    User u=new User();
    u.setName(user.getName());
    u.setAge(user.getAge());
    u.setPwd(user.getPwd());
    mv.addObject("user", u);
}

@RequestMapping(value="save")
public ModelAndView save(ModelAndView mv){
    User user=(User)mv.getModel().get("user");
    user.setName("小叶子");
    mv.setViewName("detail");
    return mv;
}
```

ModelAndView我们可以看出明显的不一样了，它存了两样东西，一样就是model，还有一样就view的name，图中可以清楚地看出来

@RequestParam

在SpringMVC中，控制器层获取参数的方式主要有两种，一种是通过request.getParameter(“... “),另一种就是通过该注解实现

一.基本使用，获取提交的参数

@RequestParam Object obj

spring会自动根据参数名字封装进入，我们可以直接拿这个参数名来用

（前端标明的name必须与obj一致）

二.各种异常情况处理

1.对传入参数指定参数名

@RequestParam（value＝“sa“）Object obj

（前端标明的name必须与sa一致 ，obj随意起 ）

2.required可以要求前端参数是否一定要传

默认为true

3.对2的说法也有特例 ，若参数是int型 ，并且required＝false，

这时不传参也会报错，因为会赋值null给int

//个人理解，不要求前端传参但也应该设好默认值，配合defaultValue使用更佳

@PathVariable

可以非常方便地获得请求URL中的动态参数

写法：value=“/{id}“

@PathVariable Integer id

下面的定义的integer型的id直接获取地址中的id

@RequestHeader

用于将请求的头信息区的数据映射功能处理方法的参数上

写法：@RequestHeader(“User-Agent“) String userAgent

@RequestHeader(value="Accept") String[] accepts

自动将请求头“User-Agent“的值赋给userAgent，并将Accept请求头的值赋给accepts 参数上

@CookieValue

写法：@CookieValue(value="JSESSIONID", defaultValue＝“ “)String sessionId

自动将JSESSIONID中的值设置到sessionId上

@SessionAttributes

允许我们有选择地指定Model中的哪些属性需要转存到HttpSession对象当中(级别为Session而已，前端获取时本来只能通过request，这个设定之后还可以通过session)

一般写法：紧接着@Controller下一行@SessionAttributes（“user“），user就是指定的model属性名，这样设定以后若该控制器传输了一个名为user的model，前端就可以从session中获取 。

@ModelAttribute(重中之重！！！！！！！！！！！！ 几种用法好好区别)

该注解只支持一个属性value，类型为String，表示绑定的属性名称

慢慢来

一.注释一个方法的参数

1.从Form表单或URL参数中获取

@ModelAttribute User user 等价于 @RequestParam(....)... 不过这个可绑定默认值

(实际上，该处不注解依旧能获取，注意User类一定要有无参构造函数)

2.从model中获取

有一个方法被注解 @ModelAttribute(“user“)，该方法会先于映射路径方法执行并返回一个User对象，其实就是生成一个键为“user”，值为User对象的model，映射路径方法的参数User user会被注解@ModelAttribute(“user“)与上一个方法对应，意指该user的属性就是上一个方法的返回值的属性，其实就是获取键为“user”的值

二.@ModelAttribute（value＝“ “）注释返回具体类的方法

简单地说，value的值其实就是指定了model的键，而返回具体类的方法最后返回的对象就是指定了model的值，就是说这个方法执行完最后就已经生成了一个model，这里就和第一点第2条对应了

三.@ModelAttribute注释void返回值的方法

其实就是第二点的勤劳版本，这里必须要自己将值存入model，不像第二点是自动的，不过这里不仅是自己要勤劳点，而且还多了几个选项，既然自己存，Model、ModelMap、ModelAndView都可以，具体示例在上面专门对比这三者的时候就已经罗列了

四.@ModelAttribute注释返回具体类的方法

额，这个嘛，就是第二点的更偷懒版本了（有没有想打人的冲动，手动捂眼），这个就更厉害了，连model的键都省了自己指定的功夫了，全根据返回类型自动指定model的键，比如User，自动生成一个“user”的键，比如Book，就自动生成一个“book”的键

五.@ModelAttribute和@RequestMapping同时注释一个方法

当然这里这两个注解都是带value的，这个有点小绕头，不过也不会怎么用吧，没事用这种处理方式干嘛（手动笑哭），简单地说下吧，@RequestMapping的value值不是我们获取到的映射路径嘛，这个方法的返回值（其它类型倒无所谓，String不能误解，不是我们跳转的视图名称），视图名称就是@RequestMapping的value值，所以要这么用，映射路径名称要和返回视图名称相同才行，那么继续说@ModelAttribute，这里@ModelAttribute的value依旧是指定了model的键，值就是方法的返回值