5、数据输出

笔记本： spring
创建时间： 2022/4/4 17:24
作者： 雷丰阳

数据输出：如何将数据带给页面；

```
/**
 * SpringMVC除过在方法上传入原生的request和session外还能怎么样把数据带给页面
 *
 * 1）、可以在方法处传入Map、或者Model或者ModelMap。
 *       给这些参数里面保存的所有数据都会放在请求域中。可以在页面获取
 *    关系：
 *       Map, Model, ModelMap：最终都是BindingAwareModelMap在工作；
 *       相当于给BindingAwareModelMap中保存的东西都会被放在请求域中；
 *
 *       Map(interface(jdk))
Model(interface(spring))
 *           ||                          //
 *           ||                         //
 *           \/                        //
 *       ModelMap(clas)               //
 *                    \\             //
 *                     \\           //
 *                    ExtendedModelMap
 *                           ||
 *                           \/
 *                    BindingAwareModelMap
 *
 * 2）、方法的返回值可以变为ModelAndView类型；
 *           既包含视图信息（页面地址）也包含模型数据（给页面带的数据）；
```

```
 *          而且数据是放在请求域中;
 *          request、session、application;
 *
 *
 * @author lfy
 *
 */
@Controller
public class OutputController {

    @RequestMapping("/handle01")
    public String handle01(Map<String,
Object> map){
        map.put("msg","你好");
        System.out.println("map的类
型: "+map.getClass());
        return "success";
    }

    /**
     * Model: 一个接口
     * @param model
     * @return
     */
    @RequestMapping("/handle02")
    public String handle02(Model model){
        model.addAttribute("msg","你好坏! ");
        System.out.println("model的类
型: "+model.getClass());
        return "success";
    }

    @RequestMapping("/handle03")
    public String handle03(ModelMap modelMap)
{
        modelMap.addAttribute("msg","你好
棒! ");
        System.out.println("modelmap的类
型: "+modelMap.getClass());
```

```
        return "success";
    }

    /**
     * 返回值是ModelAndView;可以为页面携带数据
     *@return
     */
    @RequestMapping("/handle04")
    public ModelAndView handle04(){
        //之前的返回值我们就叫视图名；视图名视图解析器是会帮我们最终拼串得到页面的真实地址;
        //ModelAndView mv = new ModelAndView("success");
        ModelAndView mv =new ModelAndView();
        mv.setViewName("success");
        mv.addObject("msg","你好哦！");
        return mv;

    }
}
```

ModelAttribute:
使用场景:
1）页面:

2）、dao：全字段更新。没带的字段会在数据库中更新为null;

```
/**
 * 测试ModelAttribute注解;
 * 使用场景：书城的图书修改为例;
 * 1）页面端;
 *      显示要修改的图书的信息，图书的所有字段都在
 * 2）servlet收到修改请求，调用dao;
 *      String sql="update bs_book set title=?,
```

```
 *                    author=?,price=?,
 *
                     sales=?,stock=?,img_path=?
 *                  where id=?";
 * 3）实际场景?
 *       并不是全字段修改；只会修改部分字段，以修改用户信息为例；
 *       username  password  address;
 *       1）、不修改的字段可以在页面进行展示但是不要提供修改输入框；
 *       2）、为了简单，Controller直接在参数位置来写Book对象
 *       3）、SpringMVC为我们自动封装book；（没有带的值是null）
 *       4）、如果接下来调用了一个全字段更新的dao操作；会将其他的字段可能变为null；
 *           sql = "update bs_book set"
 *           if(book.getBookName()){
 *               sql +="bookName=?,"
 *           }
 *           if(book.getPrice()){
 *               sql +="price=?"
 *           }
 *
 * 4）、如何能保证全字段更新的时候，只更新了页面携带的数据；
 *       1）、修改dao；代价大?
 *       2）、Book对象是如何封装的?
 *           1）、SpringMVC创建一个book对象，每个属性都有默认值，bookName就是null；
 *               1、让SpringMVC别创建book对象，直接从数据库中先取出一个id=100的book对象的信息
 *               2、Book [id=100, bookName=西游记, author=张三, stock=12, sales=32, price=98.98]
 *
 *           2）、将请求中所有与book对应的属性一一设置过来；
```

```
 *                  3、使用刚才从数据库取出的book对象，给它 的里面设置值；（请求参数带了哪些值就覆盖之前的值）
 *                  4、带了的字段就改为携带的值，没带的字段就保持之前的值
 *              3）、调用全字段更新就有问题；
 *                  5、将之前从数据库中查到的对象，并且封装了请求参数的对象。进行保存；
 *
 * @author lfy
 */
@Controller
public class ModelAttributeTestController {

    private Object o1;
    private Object o2;

    private Object b1;
    private Object b2;

    //bookDao.update(book);
    //Book [id=100, bookName=null, author=张三, stock=12, sales=32, price=98.98]
    /**
     *      String sql="update bs_book set bookName=?,
                        author=?,price=?,
                        sales=?,stock=?,img_path=?
                    where id=?";
     */
    /**
     * 可以告诉SpringMVC不要new这个book了我刚才保存了一个book；
     * 哪个就是从数据库中查询出来的；用我这个book?@ModelAttribute("haha")
     *
     *
     * 同都是BindingAwareModelMap
     * @param book
```

```
     * @return
     */
    @RequestMapping("/updateBook")
    public String
updateBook(@ModelAttribute("haha")Book
book,Map<String, Object> model){
        o2 = model;
        b2  = book;
        Object haha = model.get("haha");
        //System.out.println("传入的
model: "+model.getClass());
        System.out.println("o1==o2?"+(o1 ==
o2));
        System.out.println("b1==b2?"+(b1 ==
b2)+"-->"+(b2 == haha));

        System.out.println("页面要提交过来的图
书信息: "+book);
        return "success";
    }

    /**
     * 1）、SpringMVC要封装请求参数的Book对象不
应该是自己new出来的。
     *       而应该是【从数据库中】拿到的准备好的
对象
     * 2）、再来使用这个对象封装请求参数
     *
     * @ModelAttribute:
     *       参数：取出刚才保存的数据
     *       方法位置：这个方法就会提前于目标方法
先运行;
     *           1)我们可以在这里提前查出数据库
中图书的信息
     *           2)将这个图书信息保存起来（方便
下一个方法还能使用）
     *
     * 参数的map: BindingAwareModelMap
```

```
     */
    @ModelAttribute
    public void
hahaMyModelAttribute(Map<String, Object>
map){

        Book book =new Book(100,"西游记","吴
承恩", 98, 10, 98.98);
        System.out.println("数据库中查到的图书
信息是："+book);
        map.put("haha", book);
        b1= book;
        o1= map;
        System.out.println("modelAttribute方
法...查询了图书并给你保存起来了...他用的map的类
型："+map.getClass());
    }
}
```
