

## 1.返回值分类

### 1.1.字符串

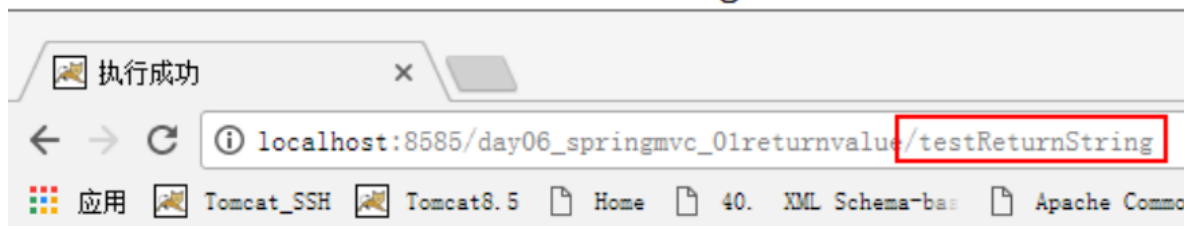
controller 方法返回字符串可以指定逻辑视图名，通过视图解析器解析为物理视图地址。

- Controller代码:

```
1 //指定逻辑视图名，经过视图解析器解析为 jsp 物理路径: /WEB-INF/pages/success.jsp
2 @RequestMapping("/testReturnString")
3 public String testReturnString() {
4     System.out.println("AccountController的testReturnString 方法执行了。。。。");
5     return "success";
6 }
```

- 运行结果:

AccountController的testReturnString 方法执行了。。。。



执行成功!

[https://blog.csdn.net/BruceLiu\\_code](https://blog.csdn.net/BruceLiu_code)

### 1.2.void

我们知道 Servlet 原始 API 可以作为控制器中方法的参数:

```
1 @RequestMapping("/testReturnVoid")
2 public void testReturnVoid(HttpServletRequest request, HttpServletResponse response)
3     throws Exception {
4 }
```

在 controller 方法形参上可以定义 request 和 response，使用 request 或 response 指定响应结果： 1、使用 request 转向页面，如下：

```
1 request.getRequestDispatcher("/WEB-INF/pages/success.jsp").forward(request, response);
```

2、也可以通过 response 页面重定向：

```
1 response.sendRedirect("testRetrunString");
```

3、也可以通过 response 指定响应结果，例如响应 json 数据：

```
1 response.setCharacterEncoding("utf-8");
2 response.setContentType("application/json;charset=utf-8");
3 response.getWriter().write("json 串");
```

### 1.3.ModelAndView

ModelAndView 是 SpringMVC 为我们提供的一个对象，该对象也可以用作控制器方法的返回值。该对象中有两个方法：

```
/**
 * Add an attribute to the model.
 * @param attributeName name of the object to add to the model
 * @param attributeValue object to add to the model (never {@code null})
 * @see ModelMap#addAttribute(String, Object)
 * @see #getModelMap() 添加模型到该对象中，通过源码分析可以看出，和昨天我们讲的请求参数封装中用到的对象是同一个
 */
public ModelAndView addObject(String attributeName, Object attributeValue) {
    getModelMap().addAttribute(attributeName, attributeValue);
    return this; 我们在页面上可以直接用el表达式获取。 获取方式: ${attributeName}
}
/**
 * Set a view name for this ModelAndView, to be resolved by the
 * DispatcherServlet via a ViewResolver. Will override any
 * pre-existing view name or View.
 */ 用于设置逻辑视图名称，视图解析器会根据名称前往指定的视图。
public void setViewName(@Nullable String viewName) {
    this.view = viewName;
}
```

• 示例代码：

```
1 /**
2  * 返回 ModelAndView
3  * @return
4  */
5  @RequestMapping("/testReturnModelAndView")
6  public ModelAndView testReturnModelAndView() {
7      ModelAndView mv = new ModelAndView();
8      mv.addObject("username", "张三");
9      mv.setViewName("success");
10     return mv;
11 }
```

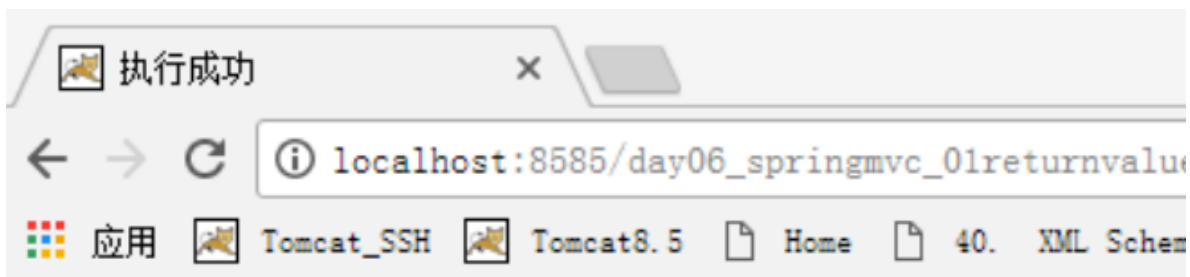
• 响应的 jsp 代码：

```

1  <%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
3  "http://www.w3.org/TR/html4/loose.dtd"> <html> <head> <meta http-
   equiv="Content-Type" content="text/html; charset=UTF-8"> <title>执
   行成功</title>
4  </head>
5  <body>
6      执行成功!
7      ${requestScope.username}
8  </body>
9  </html>

```

- 输出结果:



执行成功！张三

[https://blog.csdn.net/BruceLiu\\_code](https://blog.csdn.net/BruceLiu_code)

注意：我们在页面上获取使用的是 requestScope.username 取的，所以返回 ModelAndView 类型时，浏览器跳转只能是请求转发。

## 1.4.转发和重定向

### 1.4.1 forward 转发

controller 方法在提供了 String 类型的返回值之后，默认就是请求转发。我们也可以写成：

```

1  /**
2   * 转发
3   * @return
4   */
5  @RequestMapping("/testForward")
6  public String testForward() {
7      System.out.println("AccountController的testForward方法执行
   了。。。。");
8      return "forward:/WEB-INF/pages/success.jsp";
9  }

```

需要注意的是，如果用了 forward：则路径必须写成实际视图 url，不能写逻辑视图。它相当

于“`request.getRequestDispatcher("url").forward(request, response)`”。使用请求转发，既可以转发到 jsp，也可以转发到其他的控制器方法。

### 1.4.2 Redirect 重定向

contrller 方法提供了一个 String 类型返回值之后，它需要在返回值里使用:redirect:

```
1  /**
2   * 重定向
3   * @return
4   */
5  @RequestMapping("/testRedirect")
6  public String testRedirect() {
7      System.out.println("AccountController 的 testRedirect 方法执行了。。。");
8      return "redirect:testReturnModelAndView";
9  }
```

它相当于“`response.sendRedirect(url)`”。需要注意的是，如果是重定向到 jsp 页面，则 jsp 页面不能写在 WEB-INF 目录中，否则无法找到。

## 1.5.@ResponseBody响应json数据

### 1.5.1.使用说明

- 作用：该注解用于将 Controller 的方法返回的对象，通过 `HttpMessageConverter` 接口转换为指定格式的数据如：json,xml 等，通过 Response 响应给客户端。

### 1.5.2.使用示例

- 需求：使用 `@ResponseBody` 注解实现将 controller 方法返回对象转换为 json 响应给客户端。
- 前置知识点：Springmvc 默认用 `MappingJacksonHttpMessageConverter` 对 json 数据进行转换，需要加入 jackson 的包。

```
1  <dependency>
2      <groupId>com.fasterxml.jackson.core</groupId>
3      <artifactId>jackson-databind</artifactId>
4      <version>2.9.0</version>
5  </dependency>
```

注意：2.7.0 以下的版本用不了

- jsp 中的代码：

```

1  <script type="text/javascript"
2  src="${pageContext.request.contextPath}/js/jquery.min.js">
  </script> <script type="text/javascript">
3      $(function(){
4          $("#testJson").click(function(){
5              $.ajax({
6                  type:"post",
7
8                  url:"${pageContext.request.contextPath}/testResponseJson",
9                  contentType:"application/json;charset=utf-8",
10                 data:'{"id":1,"name":"test","money":999.0}',
11                 dataType:"json",
12                 success:function(data){
13                     alert(data);
14                 }
15             });
16         });
17     });
18 </script>
19 <!-- 测试异步请求 -->
20 <input type="button" value="测试 ajax 请求 json 和响应 json"
21       id="testJson"/>

```

- 控制器中的代码：

```

1  /**
2   * 响应json数据的控制器
3   * @Version 1.0
4   */
5  @Controller("jsonController")
6  public class JsonController {
7
8      /**
9       * 测试响应 json 数据
10      */
11      @RequestMapping("/testResponseJson")
12      public @ResponseBody Account testResponseJson(@RequestBody
13      Account account) {
14          System.out.println("异步请求: "+account);
15          return account;
16      }
17  }

```

- 运行结果：

Console Tasks JUnit

Tomcat v6.5 Server at localhost [Apache Tomcat] E:\Java\JDK1.9\jdk-9.0.1\bin\javaw.exe (2018年3月6日 下午7:58:58)

异步请求: Account [id=1, name=test, money=999.0]

控制器方法的返回值

localhost:8585/day06\_springmvc\_01returnvalue/

应用 Tomcat\_SSH Tomcat8.5

localhost:8585 显示

Object Object

确定

返回一个字符串

没有返回值

返回ModelAndView

测试请求转发

测试重定向

测试ajax请求json和响应json

[https://blog.csdn.net/BruceLiu\\_code](https://blog.csdn.net/BruceLiu_code)