

# Spring MVC 框架学习（六） ---- 返回页面+加载静态资源

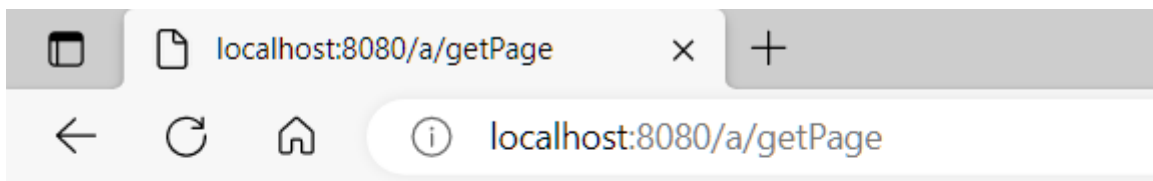
---

## 一、返回页面

---

不加 @ResponseBody ,默认是返回一个网页

```
1  @RequestMapping("/getPage")
2  public String getPage(){
3      return "index.html";
4  }
```



## 二、返回非页面的数据

---

返回非页面的数据，必须在方法或者类上加 @ResponseBody，同时 我们返回的类型 springmvc 会自动解析成对应的格式，不需要我们进行手动指定

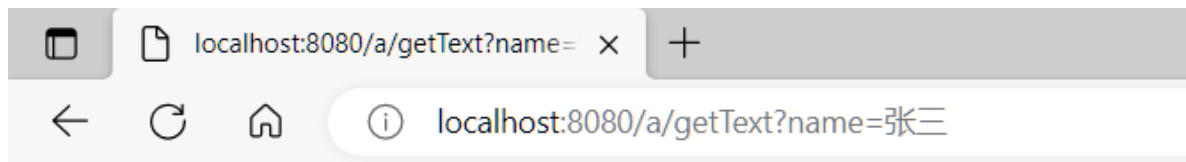
### 1、返回 text/html

```

1 @RequestMapping("/getText")
2 @ResponseBody
3 public String getHTML(String name){
4     return "<h1>你好,欢迎用户: "+name+"</h1>";
5 }
6

```

访问接口, 自动解析成 html格式



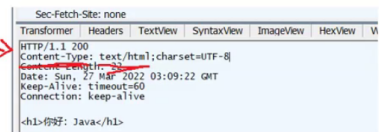
# 你好,欢迎用户: 张三

通过 Fiddler 进行抓包, 查看返回响应的格式为 text/html。

```

@ResponseBody // 当前的方法（或者类）返回是非页面的数据
@RequestMapping("/gethtml")
public String getHtml(String name) {
    return "<h1>你好: " + name + "</h1>";
}

```



## 2、返回 application/json

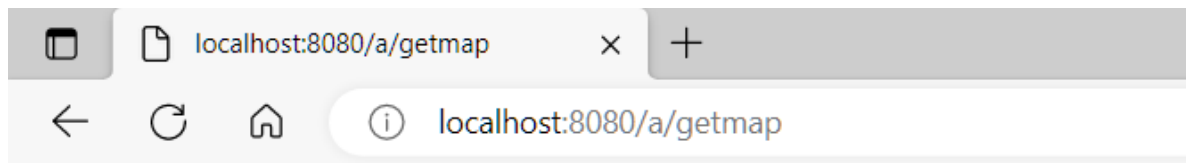
使用map存储数据, 返回map

```

1 @RequestMapping("/getmap")
2 @ResponseBody
3 public Object getJson(){
4     HashMap<Object,Object> map = new HashMap<>();
5     map.put("msg", "登陆成功");
6     map.put("status", 200);
7
8     return map;
9 }

```

自动解析称为 json 格式的数据



```
{\"msg\": \"登陆成功\", \"status\": 200}
```

### 三、加载静态资源

咱们就直接定死了写的格式

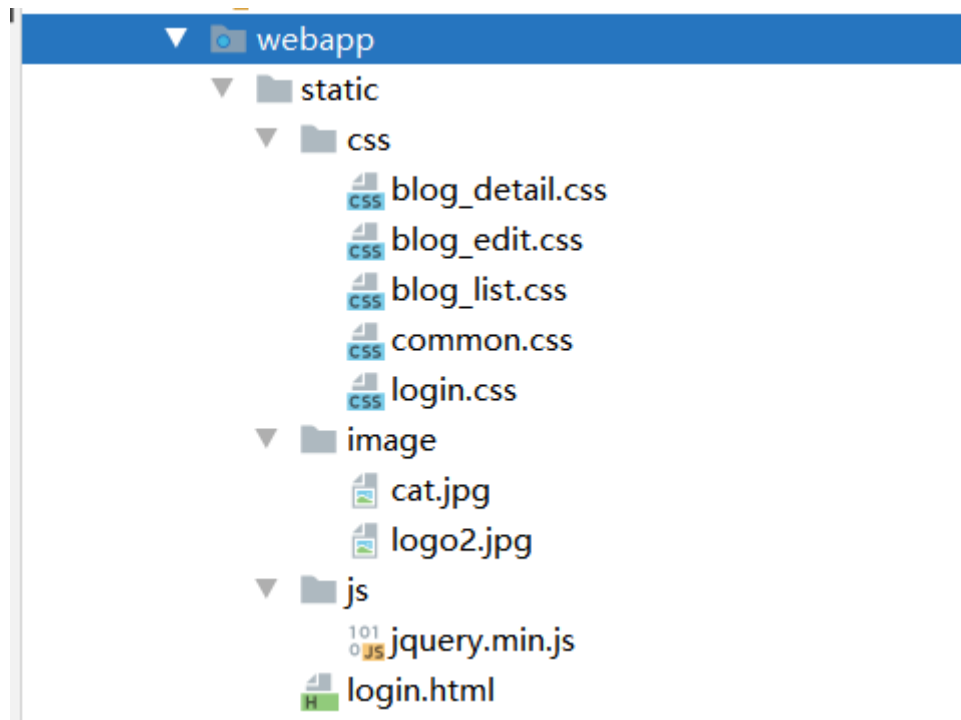
在webapp目录下创建static文件夹保存 css、js、html 资源

同时在spring-mvc.xml 文件中加入 过滤静态资源、加载静态资源的配置

```
1  <!-- 过滤静态资源， /.jsp /.html 不会经过-->
2  <mvc:default-servlet-handler/>
3
4
5  <!--加载静态资源location表示访问的路径return\"/static/login.html\",mapping表示映射的
   静态资源位置-->
6  <mvc:resources location=\"/static/css/\" mapping=\"/static/css/**\"/>
7  <mvc:resources location=\"/static/js/\" mapping=\"/static/js/**\"/>
8  <mvc:resources location=\"/static/\" mapping=\"/static/**\"/>
```

我们来试一下访问静态资源

在webapp目录下创建static文件，将css/js/html等文件添加进去



## web.xml 配置文件

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
3         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
5                             http://xmlns.jcp.org/xml/ns/javaee/web-
6                             app_4_0.xsd"
7         version="4.0">
8     <servlet>
9         <servlet-name>springmvc</servlet-name>
10        <servlet-
11class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
12        <init-param>
13            <param-name>contextConfigLocation</param-name>
14            <param-value>classpath:spring-mvc.xml</param-value>
15        </init-param>
16        <load-on-startup>1</load-on-startup>
17        <multipart-config>
18            <max-file-size>20848820</max-file-size>
19            <max-request-size>418018841</max-request-size>
20            <file-size-threshold>1048576</file-size-threshold>
21        </multipart-config>
22    </servlet>
23
24    <servlet-mapping>
25        <servlet-name>springmvc</servlet-name>
26        <url-pattern>/</url-pattern>
27    </servlet-mapping>
28 </web-app>
```

## spring-mvc.xml 配置文件

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:context="http://www.springframework.org/schema/context"
5       xmlns:mvc="http://www.springframework.org/schema/mvc"
6       xsi:schemaLocation="http://www.springframework.org/schema/beans
7                           http://www.springframework.org/schema/beans/spring-beans.xsd
8                           http://www.springframework.org/schema/context
9                           https://www.springframework.org/schema/context/spring-context.xsd
10                          http://www.springframework.org/schema/mvc
11                          http://www.springframework.org/schema/mvc/spring-mvc.xsd">
12
13
14     <!-- 开启注解扫描，将使用注解的类托管到spring 容器中-->
15     <context:component-scan base-package="com.*"/>
16
17     <!-- 过滤静态资源， /jsp /html 不会经过-->
18     <mvc:default-servlet-handler/>
19
20     <!-- 加载静态资源文件-->
21     <mvc:resources location="/static/css/" mapping="/static/css/**"/>
22     <mvc:resources location="/static/js/" mapping="/static/js/**"/>
23     <mvc:resources location="/static/" mapping="/static/**"/>
24
25     <!-- 开启mvc注解驱动-->
26     <mvc:annotation-driven>
27         <mvc:message-converters register-defaults="true">
28             <bean
29                 class="org.springframework.http.converter.StringHttpMessageConverter">
30                 <property name="supportedMediaTypes">
31                     <list>
32                         <value>text/html;charset=UTF-8</value>
33                         <value>application/json;charset=UTF-8</value>
34                     </list>
35                 </property>
36             </bean>
37         </mvc:message-converters>
38
39
40     </mvc:annotation-driven>
41
42 </beans>

```

在controller层进行访问静态html文件（经过css、js渲染）

```

1   @RequestMapping("/login")
2   public String getLog(){
3       return "redirect:/static/login.html";
4   }

```

## 三、转发和重定向

### 1、请求转发forward 和 重定向的区别

- 1、重定向 将请求重新定位到资源的位置，请求转发是服务器端进行转发的
- 2、请求重定向url地址发生改变，请求转发地址不发生变化
- 3、请求重定向于直接访问新地址的效果一样，不存在原来的外部资源不能访问，请求转发服务器端的转发可能会造成外部资源不能访问（js、css）

如果外部资源于转发访问的页面不在同一级目录下，会造成外部资源不可访问。

### 2、请求转发的访问资源问题演示

**请求转发**

我的博客系统

**登录**

用户名

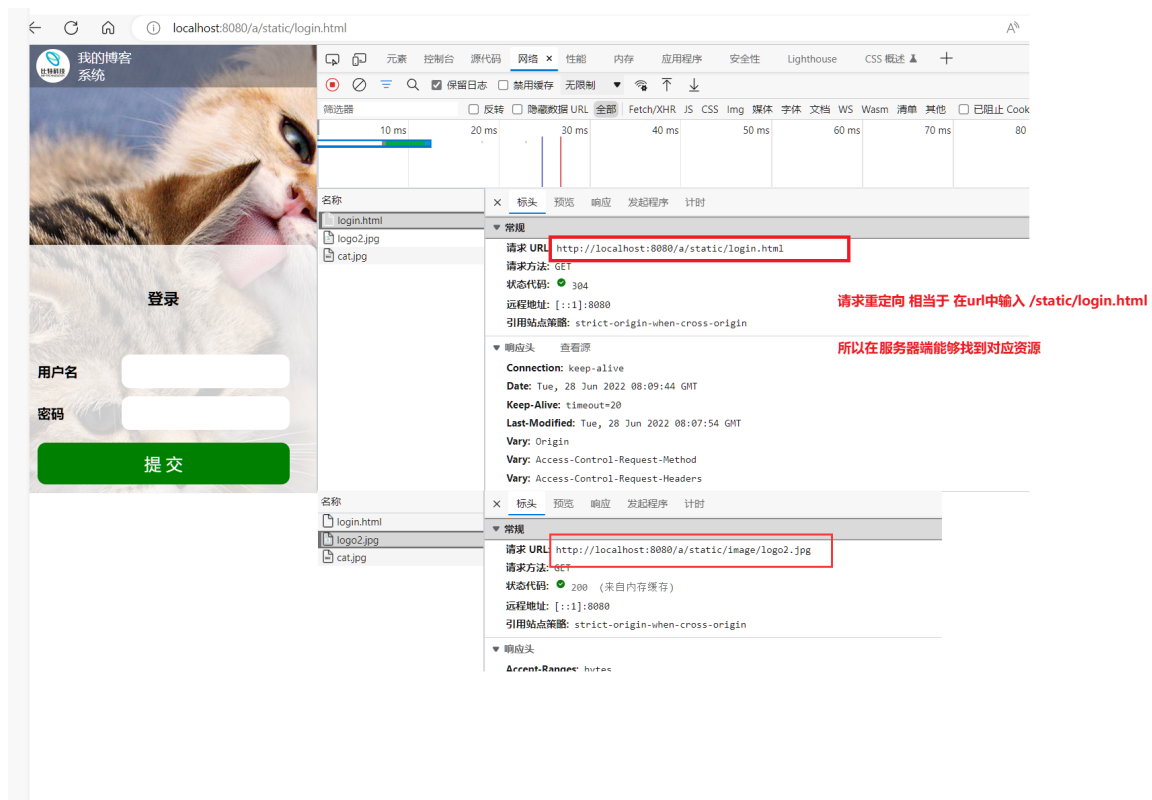
密码

请求的资源 html 与 css 不在同一个目录下面

但是服务器请求还是按照8080: /a 后面直接请求的

当然找不到外部资源文件

通过转发的请求资源都直接通过 8080:/a/login 这个接口的同一级目录下直接访问，当然找不到资源



请求重定向相当于 输入的url变了，直接访问到 /static/login/html,同时附带的资源在在这一目录下能够访问到。

### 3、页面跳转

1、请求转发：服务器放客户进行请求转发并将结果响应给客户端，URL是不会变的

2、请求重定向：服务器端请求重新定义到要访问的地址。URL会放生改变。

总结：

- 请求转发的URL地址不变，因为是服务器端进行转发和响应的，所以重定向URL地址会发生改变，因为服务器端直接将请求重定向到具体的地址上
- 使用请求转发那么有可能会发生资源丢失，访问不到外部资源。请求重定向是直接重定向到URL地址，所以请求重定向和直接访问目标地址的效果是一样的，所以不会存在外部资源丢失的情况。

## 四、组合注解

---

@RestController

相当于 @Controller + @ResponseBody

只能加到类上