

[今日课程大纲]

JSP 九大内置对象和四大作用域复习

SpringMVC 作用域传值

文件下载

文件上传

[知识点详解]

一.JSP 九大内置对象和四大作用域复习

1.九大内置对象

名称	类型	含义	获取方式
request	HttpServletRequest	封装所有请求信息	方法参数
response	HttpServletResponse	封装所有响应信息	方法参数
session	HttpSession	封装所有会话信息	req.getSession()
application	ServletContext	所有信息	getServletContext(); request.getServletContext();

out	PrintWriter	输出对象	response.getWriter()
exception	Exception	异常对象	
page	Object	当前页面对象	
pageContext	PageContext	获取其他对象	
config	ServletConfig	配置信息	

2.四大作用域

2.1 page

2.1.1 在当前页面不会重新实例化.

2.2 request

2.2.1 在一次请求中同一个对象,下次请求重新实例化一个 request 对象.

2.3 session

2.3.1 一次会话.

2.3.2 只要客户端 Cookie 中传递的 Jsessionid 不变,Session 不会重新实例化(不超过默认时间.)

2.3.3 实际有效时间:

2.3.3.1 浏览器关闭.Cookie 失效.

2.3.3.2 默认时间.在时间范围内无任何交互.在 tomcat 的 web.xml 中配置

```
<session-config>
  <session-timeout>30</session-timeout>
</session-config>
```

2.4 application

2.4.1 只有在 tomcat 启动项目时菜实例化.关闭 tomcat 时销毁
application

二.SpringMVC 作用域传值的几种方式

1. 使用原生 Servlet

1.1 在 HanlderMethod 参数中添加作用域对象

```
@RequestMapping("demo1")

public String demo1(HttpServletRequest
abc,HttpSession sessionParam){

    //request 作用域

    abc.setAttribute("req", "req 的值");

    //session 作用域

    HttpSession session = abc.getSession();

    session.setAttribute("session", "session 的值");

    sessionParam.setAttribute("sessionParam",
"sessionParam 的值");

    //appliaction 作用域

    ServletContext application =
abc.getServletContext();

    application.setAttribute("application",
```

```
"application 的值");  
  
    return "/index.jsp";  
  
}
```

2. 使用 Map 集合

2.1 把 map 中内容放在 request 作用域中

2.2 spring 会对 map 集合通过 BindingAwareModelMap 进行实例化

```
@RequestMapping("demo2")  
  
public String demo2(Map<String,Object> map){  
  
    System.out.println(map.getClass());  
  
    map.put("map","map 的值");  
  
    return "/index.jsp";  
  
}
```

3. 使用 SpringMVC 中 Model 接口

3.1 把内容最终放入到 request 作用域中。

```
@RequestMapping("demo3")  
  
public String demo3(Model model){  
  
    model.addAttribute("model", "model 的值");  
  
    return "/index.jsp";  
  
}
```

4. 使用 SpringMVC 中 ModelAndView 类

```
@RequestMapping("demo4")
```

```
public ModelAndView demo4(){  
    //参数,跳转视图  
    ModelAndView mav = new ModelAndView("/index.jsp");  
    mav.addObject("mav", "mav 的值");  
    return mav;  
}
```

三.文件下载

1.访问资源时相应头如果没有设置 Content-Disposition,浏览器默认按照 inline 值进行处理

1.1 inline 能显示就显示,不能显示就下载.

2.只需要修改相应头中 Content-Disposition="attachment;filename= 文件名"

2.1 attachment 下载,以附件形式下载.

2.2 filename=值就是下载时显示的下载文件名

3.实现步骤

3.1 导入 apatch 的两个 jar



3.2 在 jsp 中添加超链接,设置要下载文件

3.2.1 在 springmvc 中放行静态资源 files 文件夹

```
<a href="download?fileName=a.rar">下载</a>
```

3.3 编写控制器方法

```
@RequestMapping("download")

public void download(String
fileName, HttpServletResponse res, HttpServletRequest
req) throws IOException{

    //设置响应流中文件进行下载

    res.setHeader("Content-Disposition",
"attachment;filename="+fileName);

    //把二进制流放入到响应体中.

    ServletOutputStream os = res.getOutputStream();

    String path =
req.getServletContext().getRealPath("files");

    System.out.println(path);

    File file = new File(path, fileName);

    byte[] bytes =
FileUtils.readFileToByteArray(file);

    os.write(bytes);

    os.flush();

    os.close();

}
```

四.文件上传

1. 基于 apache 的 commons-fileupload.jar 完成文件上传.
2. MultipartResovler 作用:
 - 2.1 把客户端上传的文件流转换成 MutipartFile 封装类.
 - 2.2 通过 MutipartFile 封装类获取到文件流
3. 表单数据类型分类
 - 3.1 在<form>的 enctype 属性控制表单类型
 - 3.2 默认值 application/x-www-form-urlencoded,普通表单数据.(少量文字信息)
 - 3.3 text/plain 大文字量时使用的类型.邮件,论文
 - 3.4 multipart/form-data 表单中包含二进制文件内容.
4. 实现步骤:
 - 4.1 导入 springmvc 包和 apache 文件上传 commons-fileupload 和 commons-io 两个 jar
 - 4.2 编写 JSP 页面

```
<form action="upload" enctype="multipart/form-data"
method="post">

    姓名:<input type="text" name="name"/><br/>

    文件:<input type="file" name="file"/><br/>

    <input type="submit" value="提交"/>

</form>
```

4.3 配置 springmvc.xml

```
<!-- MultipartResovler 解析器 -->

<bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver">

    <property name="maxUploadSize"
value="50"></property>

</bean>

<!-- 异常解析器 -->
<bean id="exceptionResolver"
class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">

    <property name="exceptionMappings">

        <props>

            <prop
key="org.springframework.web.multipart.MaxUploadSizeExceededException">/error.jsp</prop>

        </props>

    </property>

</bean>
```

4.4 编写控制器类

4.4.1 MultipartFile 对象名必须和<input type="file"/>的 name 属

性值相同

```
@RequestMapping("upload")

public String upload(MultipartFile file,String name)
throws IOException{

    String fileName = file.getOriginalFilename();

    String suffix =
fileName.substring(fileName.lastIndexOf("."));

    //判断上传文件类型

    if(suffix.equalsIgnoreCase(".png")){

        String uuid = UUID.randomUUID().toString();

        FileUtils.copyInputStreamToFile(file.getInputStream
(), new File("E:/" +uuid+suffix));

        return "/index.jsp";

    }else{

        return "error.jsp";

    }

}
```