一、引言

1.1现在问题

> 在之前学习Servlet时，服务端通过Servlet响应客户端页面，有什么不足之处？
>
> - 开发方式麻烦：继承父类、覆盖方法、配置Web.xml或注解
> - 代码修改麻烦：重新编译、部署、重启服务
> - 显示方式麻烦：获取流、使用println("");逐行打印
> - 协同开发麻烦：UI负责美化页面，程序员负责编写代码。UI不懂Java，程序员又不能将所有前端页面的内容通过流输出

二、JSP(Java Server Pages)

2.1概念

> 简化的Servlet设计，在HTML标签中嵌套Java代码，用以高效开发Web应用的动态网页

2.2作用

> 替换显示页面部分的Servlet(使用*.jsp文件替换XxxJSP.java)

三、JSP开发【重点】

3.1创建JSP

> 在web目录下新建*.jsp文件（与WEB-INF平级）

3.1.1JSP编写Java代码

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
Now:<%= new java.util.Date()%>
</body>
</html>
```

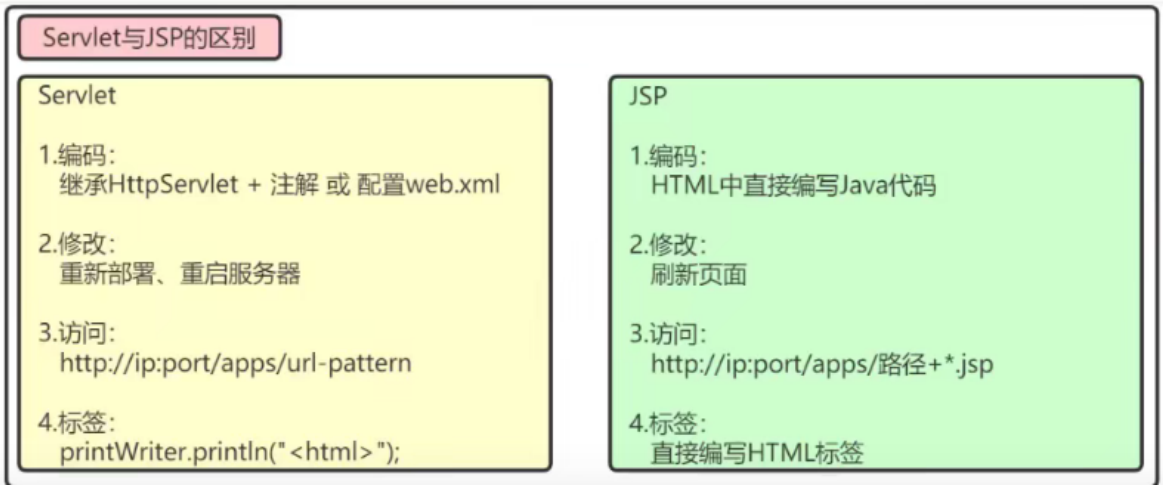
- **使用<%=%>标签编写Java代码在页面中显示系统时间**

3.1.2访问JSP

> 在浏览器输入http://ip:port/项目路径/资源名称

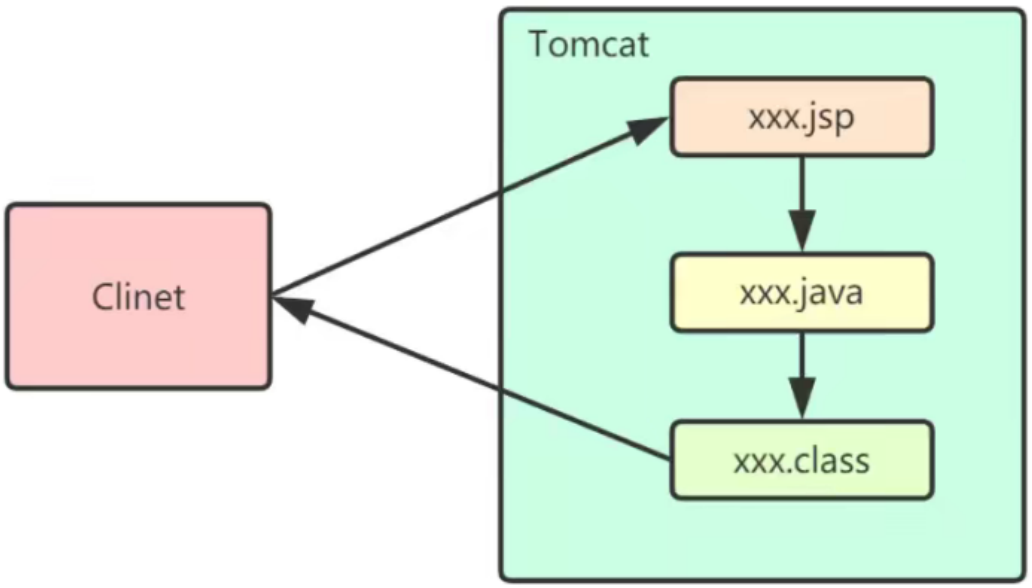
3.2JSP与Servlet

- 关系
  - JSP文件在容器中会转换成Servlet执行。
  - JSP是对Servlet的一种高级封装。本质还是Servlet。
- 区别
  - 与 Servlet 相比：JSP可以很方便的编写或者修改HTML网页而不用去面对大量的println语句。



3.3JSP实现原理

Tomcat会将xxx.jsp转换成Java代码，进而编译成.class文件运行，最终将运行结果通过response响应给客户端



3.3.1JSP源文件存放目录

> 使用IDEA开发工具，Tomcat编译后的JSP文件（Xxx_jsp.class 和 Xxx_jsp.java）的存放地点：
>
> C:\用户\账户名\.IntelliJIdea2019.1\system\tomcat\项目名称\work\Catalina\localhost\应用上下文\org\apache\jsp

四、JSP与HTML集成开发

4.1脚本

4.1.1普通脚本

4.1.2声明脚本

4.1.3输出脚本

4.2JSP注释

4.2.1语法规则

4.2.2注释41

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>脚本的使用</title>
</head>
<body>
<h1>注释</h1>
<%--JSP注释在网页中不会被显示--%>
<!--HTML注释在网页源代码中会显示-->
</body>
</html>

```

4.3JSP指令

4.3.1Page指令

4.3.2 include指令

4.3.3taglib指令

4.4动作标签

4.4.1 include

4.4.2 useBean

4.4.3 setProperty

4.4.4 getProperty

4.4.5 forward

4.4.6 param

4.5内置对象

4.5.1四大域对象

4.5.2pageContext对象

4.5.3pageContext获取其他内置对象

```
<%
    pageContext.getRequest();//返回Requst内置对象
    pageContext.getResponse();//返回respone内置对象
    pageContext.getSession();//返回exce内置对象
    pageContext.getServletContext();//返回内置对象
    pageContext.getOut();//返回内置对象
    pageContext.getException();//返回内置对象
    pageContext.getPage();//返回内置对象
    pageContext.getServletConfig();//返回内置对象
%>
```

4.5.4pageContext操作其他内置对象的作用域

4.6整合

五、EL表达式（Expression Language）

5.1概念

5.2作用

5.3EL的应用(获取基本类型、字符串)

5.3.1EL应用类型

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>EL表达式</title>
</head>
<body>
<%
  request.setAttribute("key1","value1");
  session.setAttribute("key2","value2");
  application.setAttribute("key3","value3");
%>
<%--通过作用域对象获取数据--%>
<h1>通过作用域对象获取:</h1>
<h1><%=request.getAttribute("key1")%></h1>
<h1><%=session.getAttribute("key2")%></h1>
<h1><%=application.getAttribute("key3")%></h1>
<%--通过EL表达式获取数据--%>
<hr>
<h1>通过El表达式获取数据</h1>
<h1>${requestScope.key1}</h1><%--获取request作用域中name对应的值，找到就返回，没找到返回""--%>
<h1>${sessionScope.key2}</h1>
<h1>${applicationScope.key3}</h1>
```

```
23  <h1>${key3}</h1><%--从最小作用域逐级查找name对应的值，找到就返回，没找到返回""--%>
24  </body>
25  </html>
```

5.3.2EL和JSP脚本的区别

5.4EL的应用(获取引用类型)

5.5EL的应用(获取数组、集合的元素)

EL可以获取Array、List、Map中的元素，Set由于没下标，无法直接访问元素，后续可遍历

```
<%
    int[] array = new int[]{1,2,3,4,5};
    request.setAttribute("array",array);

    List<Emp> emps = new ArrayList<>();
    emps.add(new Emp(1,"gavin",2000,19));
    emps.add(new Emp(2,"marry",3000,29));
    emps.add(new Emp(3,"jack",4000,39));
    request.setAttribute("emps",emps);

    Map<String,String> maps = new HashMap<>();
    maps.put("CN","中国");
    maps.put("FK","法国");
    maps.put("US","美国");
    request.setAttribute("maps",maps);
%>
${requestScope.array[0]}

${requestScope.emps[0]} <%-- 也可以用 ${requestScope.emps.get(0)}  --%>

${requestScope.maps.CN} <%-- 也可以用 ${requestScope.maps["US"]} -->
```

5.6EL的运算符

操作符	描述
.	访问一个Bean属性或者一个映射条目
[]	访问一个数组或者链表的元素
+	加
-	减或负
*	乘
/ or div	除
% or mod	取模
== or eq	测试是否相等
!= or ne	测试是否不等
< or lt	测试是否小于
> or gt	测试是否大于
<= or le	测试是否小于等于
>= or ge	测试是否大于等于
&& or and	测试逻辑与
|| or or	测试逻辑或
! or not	测试取反
empty	测试是否空值

```jsp
<%--
  Created by IntelliJ IDEA.
  User: 丛雨
  Date: 2021/10/7
  Time: 17:31
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>EL表达式</title>
</head>
<body>
<%
    request.setAttribute("nums",1234);
    request.setAttribute("ss","");
%>
<h1>empty运算符</h1>
<h1>${empty ss}</h1>
<hr/>
<h1>算术运算符</h1>
<h1>${nums+5}</h1>
<h1>${nums-5}</h1>
<h1>${nums*5}</h1>
<h1>${nums/5}</h1>
<h1>${nums%5}</h1>
<hr/>
<h1>关系运算符</h1>
```

```
<h1>${nums eq 1234}</h1>
<h1>${nums ne 1234}</h1>
<h1>${nums gt 1234}</h1>
<h1>${nums lt 1234}</h1>
<h1>${nums ge 1234}</h1>
<h1>${nums le 1234}</h1>
<hr/>
<h1>逻辑运算符</h1>
<h1>${nums>1000 and nums!=1200}</h1>
<h1>${nums>1000 or nums==2000}</h1>
<h1>${not(nums>1234)}</h1>
</body>
</html>
```

5.7隐式对象

> EL 表达式语言定义了11个隐式对象

隐含对象	描述
pageScope	page 作用域
requestScope	request 作用域
sessionScope	session 作用域
applicationScope	application 作用域
param	Request 对象的参数，字符串
paramValues	Request对象的参数，字符串集合
header	HTTP 信息头，字符串
headerValues	HTTP 信息头，字符串集合
initParam	上下文初始化参数
cookie	Cookie值
pageContext	当前页面的pageContext

5.7.1获得应用上下文

```
<%=request.getContextPath()%>
${pageContext.request.context}
```

5.7.2获取Cookie对象

```
<h1>${cookie.username}</h1>//获取username的cookie对象
<h1>${cookie.password}</h1>//获取password的cookie对象
<h1>${cookie.username.value}</h1>//获取username的cookie的value值
<h1>${cookie.password.value}</h1>//获取password的cookie的value值
```

六、JSTL标准标签库

6.1现有问题

- EL主要是用于作用域获取数据，虽然可以做运算判断，但是得到的都是一个结果，做展示。
- EL不存在流程控制。比如判断。
- EL对于集合只能做单点访问，不能实现遍历操作。比如循环。

6.2什么是JSTL

- JSTL:全称Java Server Pages Standard Tag Library
- JSP标准标签库（JSTL）是一个JSP标签集合。

6.3JSTL的作用

- 可对EL获取到的数据进行逻辑操作。
- 与EL合作完成数据的展示。

6.4JSTL的使用

- 导入两个 jar 文件：standard.jar 和 jstl.jar 文件拷贝到 /WEB-INF/lib/ 下
- 在JSP页面引入标签库<% @taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c">

6.5核心标签

6。5.1条件标签if判断

语法：<c:if test ="条件"> < /c:if>

```
<!-- test属性中是条件，但是条件需要使用EL表达式来书写 -->
<h3>条件标签：if</h3>
<c:if test="${8>2 }">
    8大于2是成立的
 </c:if>
<c:if test="${8<2 }">
    8小于2是成立的
</c:if>
```

```
<!--test属性中是条件，但是条件需要使用EL表达式来书写-->
<h3>条件标签:if</h3>
<c:if test="${8>2}">
8大于2是成立的
</c:if>
<c:if test="${8<2}">
8小于2是不成立的
</c:if>
```

6.5.2多条件choose判断

```
语法：<c:choose >
        <c:when test="条件1">结果1< /c:when>
        <c:when test="条件2">结果2< /c:when>
        <c:when test="条件3">结果3< /c:when>
        <c:otherwise >结果4< /c:otherwise>
      < /c:choose>
```

```
<%
    request.setAttribute("age",18);
%>
<c:choose>
    <c:when test="${age<18}"><h1>少年</h1></c:when>
    <c:when test="${age>=18&&age<30}"><h1>青年</h1></c:when>
    <c:when test="${age>=30&&age<50}"><h1>中年</h1></c:when>
    <c:otherwise><h1>老年</h1></c:otherwise>
</c:choose>
```

6.5.3迭代foreach标签

```
语法
<c:foreach
  var="变量名"
  items="集合"
  begin="起始下标"
  end="结束下标"
  step="间隔长度"
  varstatus="遍历状态">
< /c:foreach>
```

```
<h1>迭代foreach标签</h1>
<%
    List<String> list=new ArrayList<>();
    list.add("A");
    list.add("B");
    list.add("C");
    list.add("D");
    request.setAttribute("list",list);

    List<User>users=new ArrayList<>();
    users.add(new User("tom","123456"));
    users.add(new User("marry","123456"));
    users.add(new User("jack","123456"));
    users.add(new User("gavin","123456"));
    request.setAttribute("users",users);
%>
<c:forEach var="s" items="${list}" begin="1" end="2">
```

```
18  <h1>${s}</h1>
19  </c:forEach>
20  <c:forEach var="s" items="${users}">
21      <h1>${s.username}:${s.password}</h1>
22  </c:forEach>
```

6.5.4url标签

> 在Cookie禁用的情况下，通过重写URL拼接JSESSIONID来传递ID值。便于下一次访问时仍可查找到上一次的Session对象。

```
<c:url context='${pageContext.request.contextPath}' value='/xxxController' />

//在form表单的action中嵌套动态路径
<form action="<c:url context='${pageContext.request.contextPath}' value='/xxxController' />">

</form>
```

- **经验：所有涉及到页面跳转或者重定向跳转时，都应该使用URL重写。**