

概述

JavaEE是一种开发框架/架构

- jsp和servlet是javaee开发框架中的内容
-

Tomcat

- 可以修改端口号
 - tomcat安装目录/conf
 - server.xml
 - 修改
 - 重启
- 目录webapps, 简单部署直接将文件拖入
- 默认端口8080, web应用默认端口80

Tomcat 子目录及其说明

子目录	说明
bin	命令中心 (启动命令, 关闭命令.....)
conf	配置中心 (端口号, 内存大小.....)
lib	Tomcat 的库文件。Tomcat 运行时需要的 jar 包所在的目录。
logs	存放日志文件。
temp	存储临时产生的文件, 即缓存。
webapps	存放项目的文件, web 应用放置到此目录下浏览器可以直接访问。
work	编译以后的 class 文件。

JSP

- 执行过程/原理
 - 客户端发出Request请求
 - JSP Container 将 JSP 转译成 Servlet 的源代码
 - 将Servlet源代码经过编译后, 并加载到内存执行
 - 把结果 Resonse 至客户端
- JSP和Servlet的关系
 - JSP 文件在容器中会转成Servlet执行
 - JSP是对Servlet的一种高级封装, 本质还是Servlet

HTML

- 表单以及基本元素的定义和使用
 - 文本域 `<input type="text">`
 - 密码字段 `<input type="password">`
 - 单选按钮 `<input type="radio" name="sex" value="male">Male`
 - 复选框 `<input type="checkbox" name="vehicle" value="Car">I have a car`
 - 提交按钮 `<input type="submit" value="Submit">`
 - 下拉列表

- ```
<select name="cars">
 <option value="volvo">Volvo</option>
 <option value="saab">Saab</option>
 <option value="fiat">Fiat</option>
 <option value="audi">Audi</option>
</select>
```

- 文本框

- ```
<textarea rows="10" cols="30">
我是一个文本框。
</textarea>
```

JavaScript

- DOM模型
 - DOM是Javascript操作网页的接口，全称叫文档对象模型（Document Object Model）。作用是将网页转化为一个Javascript对象，从而用Javascript脚本进行各种操作（比如增删元素等）。
 - `document.getElementById()`
 - `document.getElementsByName()`
 - `document.getElementsByTagName()`
- 必须嵌在html页面中才可执行
- ajax
 - 异步发送请求
 - 局部内容的更新
 - 响应速度的提升
- XMLHttpRequest
 - 用于在后台与服务器交换数据。
 - 在不重新加载页面的情况下更新网页
 - 在页面已加载后从服务器请求数据
 - 在页面已加载后从服务器接收数据
 - 在后台向服务器发送数据

基本语法

Java脚本 和 Js脚本

- JS脚本中可以嵌入Java脚本

Java 和 JavaScript 之间的主要区别

| Java | JavaScript |
|---|---|
| Java 是强类型语言，必须首先声明变量才能在程序中使用。在 Java 中，在编译时检查变量的类型。 | JavaScript 是弱类型语言，具有更宽松的语法规则。 |
| Java 是一种面向对象的编程语言。 | JavaScript 是一种基于对象的脚本语言。 |
| Java 应用程序可以在任何 虚拟机 (JVM) 或浏览器中运行。 | JavaScript 代码过去只能在浏览器中运行，但现在它可以通过 Node.js 在服务器上运行。 |
| Java 的对象是基于类的，即使我们不能在不创建类的情况下用 Java 编写任何程序。 | JavaScript 对象是基于原型的。 |
| Java 程序的文件扩展名为“.Java”，将源代码翻译成字节码，由 JVM (Java 虚拟机) 执行。 | JavaScript 文件的文件扩展名为“.js”，它被解释但不编译，每个浏览器都有 Javascript 解释器来执行 JS 代码。 |
| Java 是一种独立语言。 | 包含在网页中并与其 HTML 内容集成。 |
| Java 有一种基于线程的并发方法。 | Javascript 具有基于事件的并发方法。 |
| Java 支持 多线程 。 | Javascript 不支持多线程。 |

响应状态码

- 200，表示成功返回响应
- 400，服务器认为客户端出现了错误，但不能明确判断为以下哪种错误时使用此错误码。
- 403，服务器理解请求的含义，但没有权限执行此请求
- 404，服务器理解请求的含义，但没有权限执行此请求
- 500，服务器内部错误

- 1xx: 指示信息--表示请求已接收, 继续处理。
- 2xx: 成功--表示请求已被成功接收、理解、接受。
- 3xx: 重定向--要完成请求必须进行更进一步的操作。
- 4xx: 客户端错误--请求有语法错误或请求无法实现。
- 5xx: 服务器端错误--服务器未能实现合法的请求。

JSP指令

| 指令 | 描述 |
|---------------------------------------|---------------------------------|
| <code><%@ page ... %></code> | 定义网页依赖属性, 比如脚本语言、error页面、缓存需求等等 |
| <code><%@ include ... %></code> | 包含其他文件 |
| <code><%@ taglib ... %></code> | 引入标签库的定义 |

- Page 指令
 - Page指令为容器提供当前页面的使用说明。一个JSP页面可以包含多个page指令。
 - Page指令的语法格式 `<%@ page attribute="value" %>`
 - 等价的XML格式 `<jsp:directive.page attribute="value" />`
- Include 指令
 - JSP可以通过include指令来包含其他文件。被包含的文件可以是JSP文件、HTML文件或文本文件。包含的文件就好像是该JSP文件的一部分, 会被同时编译执行。
 - Include指令的语法格式 `<%@ include file="文件相对 url 地址" %>`
 - 等价的XML语法 `<jsp:directive.include file="文件相对 url 地址" />`
- Taglib 指令
 - Taglib指令引入一个自定义标签集合的定义, 包括库路径、自定义标签。
 - Taglib指令的语法 `<%@ taglib uri="uri" prefix="prefixOfTag" %>`
 - 等价的XML语法 `<jsp:directive.taglib uri="uri" prefix="prefixOfTag" />`

JSP动作

| 语法 | 描述 |
|-----------------|---------------------------------|
| jsp:include | 在页面被请求的时候引入一个文件。 |
| jsp:useBean | 寻找或者实例化一个JavaBean。 |
| jsp:setProperty | 设置JavaBean的属性。 |
| jsp:getProperty | 输出某个JavaBean的属性。 |
| jsp:forward | 把请求转到一个新的页面。 |
| jsp:plugin | 根据浏览器类型为Java插件生成OBJECT或EMBED标记。 |
| jsp:element | 定义动态XML元素 |
| jsp:attribute | 设置动态定义的XML元素属性。 |
| jsp:body | 设置动态定义的XML元素内容。 |
| jsp:text | 在JSP页面和文档中使用写入文本的模板 |

- 常见属性

- id属性:

id属性是动作元素的唯一标识，可以在JSP页面中引用。动作元素创建的id值可以通过PageContext来调用。

- scope属性:

该属性用于识别动作元素的生命周期。id属性和scope属性有直接关系，scope属性定义了相关联id对象的寿命。scope属性有四个可能的值：(a) page, (b) request, (c) session, 和 (d) application。

- <jsp:include> 动作元素

- 把指定文件插入正在生成的页面

- 语法格式 `<jsp:include page="相对 URL 地址" flush="true" />`

- <jsp:forward> 动作元素

- 把请求转到另外的页面

- `<jsp:forward page="相对 URL 地址" />`

- <jsp:useBean> 动作元素

- 用来加载一个将在JSP页面中使用的JavaBean。

- `<jsp:useBean id="name" class="package.class" />`

- | 属性 | 描述 |
|----------|--|
| class | 指定Bean的完整包名。 |
| type | 指定将引用该对象变量的类型。 |
| beanName | 通过 java.beans.Beans 的 instantiate() 方法指定Bean的名字。 |

- <jsp:setProperty> 动作元素

- 用来设置已经实例化的Bean对象的属性

- ```
<jsp:useBean id="myName" ... />
...
<jsp:setProperty name="myName" property="someProperty" .../>
```
- ```
<jsp:useBean id="myName" ... >
...
    <jsp:setProperty name="myName" property="someProperty" .../>
</jsp:useBean>
```

| 属性 | 描述 |
|----------|---|
| name | name属性是必需的。它表示要设置属性的是哪个Bean。 |
| property | property属性是必需的。它表示要设置哪个属性。有一个特殊用法：如果property的值是"*"，表示所有名字和Bean属性名字匹配的请求参数都将被传递给相应的属性set方法。 |
| value | value 属性是可选的。该属性用来指定Bean属性的值。字符串数据会在目标类中通过标准的valueOf方法自动转换成数字、boolean、Boolean、byte、Byte、char、Character。例如，boolean和Boolean类型的属性值（比如"true"）通过 Boolean.valueOf转换，int和Integer类型的属性值（比如"42"）通过Integer.valueOf转换。value和param不能同时使用，但可以使用其中任意一个。 |
| param | param 是可选的。它指定用哪个请求参数作为Bean属性的值。如果当前请求没有参数，则什么事情也不做，系统不会把null传递给Bean属性的set方法。因此，你可以让Bean自己提供默认属性值，只有当请求参数明确指定了新值时才修改默认属性值。 |

○ `<jsp:getProperty>` 动作元素

- 提取指定Bean属性的值，转换成字符串，然后输出

- ```
<jsp:useBean id="myName" ... />
...
<jsp:getProperty name="myName" property="someProperty" .../>
```

属性	描述
name	要检索的Bean属性名称。Bean必须已定义。
property	表示要提取Bean属性的值

## Include指令 和 Include动作

同：

- 最终执行结果相同，都想将别的文件内容包含到当前文件中来

异：

- include指令，静态包含：
  - `<%@include file="文件的URL"%>`
  - 在转换成java文件的时候将包含文件的内容“复制”到主体文件，然后作为整体编译。
  - 不同文件中不允许存在相同的变量，因为要合到一个文件中一起编译。
- include动作，动态包含：
  - `<jsp:include page="文件的URL"%>`
  - 对每个jsp文件分别转换、分别编译。
  - 不同文件中允许存在相同的变量，反正是分开编译。

## 内置对象

- request 对象的使用
  - 设置字符编码
    - `request.setCharacterEncoding("utf-8");`
  - 获取参数，返回的是String，用于读取提交的表单中的值
    - 单值参数的获取 `request.getParameter("")`
    - 多值参数的获取 `request.getParameterValues("")`
  - 获取属性，返回的是Object，需进行转换,可用setAttribute设置成任意对象
    - `request.getAttribute("")`
    - `request.setAttribute("") = ;`
  - 读取 Cookie
    - 获取cookie数组: `cookies = request.getCookies()`
    - 遍历数组：
      - ```
for (Cookie cookie : cookies) {
    String name = cookie.getName(); // 参数名
    String value = cookie.getValue(); // 参数值
}
```
- response 对象
 - 添加 Cookie
 - `Cookie cookie = new Cookie("参数名", "参数值");`
`cookie.setMaxAge(60);` // 设置60秒的过期时间
`response.addCookie(cookie);` // 将cookie添加到response中
 - 读取 Cookie，参考 request 部分
 - 删除 Cookie
 - 通过 request 获取cookie后对其过期时间设置为0
 - `cookie.setMaxAge(0);` // 设置cookie已经过期
`response.addCookie(cookie);` // 重新添加到response中
 - 设置响应头
 - 设置响应内容的类型 `response.setContentType("text/html;charset=utf-8");`
 - 设置响应的编码集 `response.setCharacterEncoding("utf-8")`

- 重定向 `response.sendRedirect("http://www.baidu.com")`
- 重定向和转发的区别：

| | 重定向 | 转发 |
|-------------|--|--|
| 代
码 | <code>response.sendRedirect("")</code> | <code>RequestDispatcher Dis =
request.getRequestDispatcher("");
Dis.forward(request, response);</code> |
| 效
果 | 都会打开目标网页 | 都会打开目标网页 |
| 地
址
栏 | | |

■ 代码：

■ 重定向： `response.sendRedirect("http://www.baidu.com")`

■ 转发：

■ `RequestDispatcher Dis =
request.getRequestDispatcher("http://www.baidu.co");
Dis.forward(request, response);`

■ 效果：

■ 都会打开目标页面

■ 地址栏：

- 重定向后地址栏会改成目标网页的网址
- 转发以后地址栏的网址不会改变

■ 数据共享：

■

• 内置对象

- request
- cookie - 客户端的文本文件，存储名值对
- session - 会话
- application
- response （请求转发和页面重定向的区别）

• servlet

- 定义、继承
- 基本的配置方式（注解、xml文件）
- 生命周期方法
- 内置对象的获取

• javabean

- 定义规则
- jsp:useBean

- MVC设计模式 model、view、control
- el 表达式
- jstl 自定义标签库
- 常用组件
 - 文件下载
 - 发送邮件
- 数据库访问JDBC
 - java databases connectivity
 - 核心代码