### JSP/Serlvet

### 简介和标签的引入

#### jsp简介

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

每一个jsp文件的开头

Java Server Pages，是一种动态网页开发技术。jsp做页面的展现

#### Servlet简介

Servlet做业务处理

Servlet是一个功能，如果你希望你的项目功能多一些，那就要多写一此Servlet

Servlet是JavaWeb三大组件之一，也是最重要的组件！ 三大组件：Servlet、Filter、Listener

#### jsp标签引入

**<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>**

使用：

集合

<c:forEach items="${messagelist }" var = "message">

<h3>${fn:length(messagelist) }</h3>

</c:forEach>

**<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>**

引入之后可以使用foreach标签等其他

<c:forEach begin="0" items="${datas}" step="1" var="Data" varStatus="varsta">

<option value="${Data.value}">${Data.name}</option>

</c:forEach>

**<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>**

<fmt:formatNumber type="number" value="123" maxFractionDigits="2"/>

### JSP语法

#### 基本语法

**参考思维导图**

\* **局部代码 java代码声明：**

》<%java代码%> 会被转译到——jspServlet方法中，

\* **全局代码段声明：**

》<%!声明方法%> 会被转译成全局代码

\* **表达式语句（重要）：**

》<%=变量名/方法 %> 后面一定不能加分号

\* **注释：**

》HTML注释： 会被转译并发送到客户端

》css注释： 会被转译，还会被发送

》js注释： 会被转译，还会被发送

》java注释： 会被转译

》jsp注释： %-- 注释内容 --% 直接不转译

#### 使用案例

**案例1：**

<%

String path = request.getContextPath();

String basePath = request.getScheme()+"://"+request.getServerName()+":"+

request.getServerPort()+path+"/";

%>

#### 九大内置对象

四个作用域：

》pageContext 页面上下文对象，此对象封装了其他对象的获取方式 当前页面有效

》request 请求对象 同一个请求范围内有效

》session session对象 同一个会话有效

》application ServletContext对象 整个WEB服务器有效

两个输出：

》out 输出对象

》response 输出对象，比resp对象多了缓冲区，效率提高

其他三个：

》page 代表该页面对象

》config ServletConfig

》exception 异常对象，弃用

### EL表达式

学习案例：（http://www.cnblogs.com/zhplovelnn/p/6947696.html）

\*描述：

》${javaBean.property}

》EL 提供 . 和 [ ] 两种运算符来导航数据。

》${sessionScope.user.sex}等于${sessionScope.user["sex"]}

》当要存取的属性名称中包含一些特殊字符，如. 或 – 等并非字母或数字的符号，就一定要使用 [ ]，

》》例如：${user.My-Name } ，应当改为：${user["My-Name"] }

》${sessionScope.user[data]} 假如data是动态变量

》》如果要动态取值时，就可以用上述的方法来做，但. 无法做到动态取值。

\* **隐含对象（11个）：**

》pageContext 表示JSP的PageContext

》RequestScope 指定范围

》sessionScope 指定范围

》applicationScope 指定范围

》PageScope 指定范围

》param ServletRequest.getParameter(String name)。

》paramValues ServletRequest.getParameterValues(String name)。

》header ServletRequest.getHeader(String name)。

》headerValues ServletRequest.getHeaders(String name)。

》cookie HttpServletRequest.getCookies()

》initParam ServletContext.getInitParameter(String name)。

**\* 使用pageContext：**

》${pageContext.request.queryString} 取得请求的参数字符串

》${pageContext.request.requestURL} 取得请求的URL，但不包括请求之参数字符串,即servlet的HTTP地址。

》${pageContext.request.contextPath} 服务的webapplication的名称

》${pageContext.request.method} 取得HTTP的方法(GET、POST)

》${pageContext.request.protocol} 取得使用的协议(HTTP/1.1、HTTP/1.0)

》${pageContext.request.remoteUser} 取得用户名称

》${pageContext.request.remoteAddr} 取得用户的IP地址

》${pageContext.session.new} 判断session是否为新的，所谓新的session，表示刚由server产生而client尚未使用

》${pageContext.session.id} 取得session的ID

》${pageContext.servletContext.serverInfo} 取得主机端的服务信息

**\* EL算术运算：**

》所有在Java语言里支持的算术运算符，表达式语言都可以使用；

**\* EL关系运算符：**

》

**\* EL逻辑运算符：**

》

**\* Empty 运算符：**

》Empty 运算符主要用来判断值是否为空（NULL,空字符串，空集合）。

**\* 条件运算符**

》${ A ? B : C}

### 运行原理

#### Jsp原理

#### Servlet运行原理

web服务器收到一个http请求后，判断请求内容，若是静态页面数据，自行处理，若为动态数据，交给Servlet容器，Servlet容器找到相应Servlet实例处理；处理结果交给web服务器，再转交给客户端。

》针对同一个Servlet，Servlet容器会在第一次收到HTTP请求时建立一个Servlet实例，

》》然后启动一个线程，第二次收到http请求后，Servlet容器无需创建相同Servlet ,

》》仅开启第二个线程来处理请求。多线程的方式有效提高执行效率，降低服务器负担。

**运行过程**：

1、Web服务器首先检查是否已经装载并创建了该Servlet的实例对象。如果是，则执行4

2、装载并创建该Servlet的一个实例对象

3、调用Servlet实例对象的init()方法 只调用一次，对一些内容初始化

4、创建一个用于封装HTTP请求消息的HttpServletRequest对象和一个代表HTTP响应消息的HttpServletResponse对象，然后调用Servlet的service方法并将请求和响应对象作为参数传递进去。

5、WEB应用程序被停止或重新启动之前，Servlet引擎将卸载Servlet，并在卸载之前调用Serlvet的destroy()方法（以便释放它所使用的资源）

### 简单使用案例

#### Jsp

#### Servlet

**Web.xml**

<servlet>

<servlet-name>ServletDemo1</servlet-name>

<servlet-class>com.chen.serlvet<servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>ServletDemo1</servlet-name>

<url-pattern>/servlet/ServletDemo1</url-pattern>

</servlet-mapping>

控制类上使用注解：

@Servlet

### 中文乱码的解决

》生成中文乱码的根本原因，jsp页面默认编码格式“ISO-8859-1” 不支持中文

》三种支持中文的编码方式：gb2312（常用简体），gbk（比前者好），utf-8（国际编码）

post方式提交表单的解决方案：

》1、request.setCharacterEncoding("utf-8")

》2、response.setCharacterEncoding("utf-8")

get方式提交表单的解决方案：

》1.读取数据时对数据进行编码 new String(s.getBytes("iso-8859-1"), "utf-8")

》2.通过配置文件设置解决中文乱码显示

》》配置tomcat\conf\server.xml文件

》》<Connector connectionTimeout="20000" port="8080" protocol="HTTP/1.1"

》》redirectPort="8843" URIEncoding="UTF-8" />

### 过滤器

实现接口javax.servlet.Filter，拦截中可以在doFilter中做你想做的事

#### 过滤器作用：

》身份验证过滤器（

》数据压缩过滤器

》加密过滤器

》触发资源访问事件过滤器

》图像转换过滤器

》日志记录和审核过滤器

》MIME-TYPE 链过滤器

》标记化过滤器

》XSL/T 过滤器（XSL/T Filters），转换 XML 内容

#### 案例：

**代码：**

public class I18nFilter implements Filter {

@Override

public void destroy() {

System.out.println("销毁》》》》");

}

@Override

public void doFilter(ServletRequest req, ServletResponse resp,

FilterChain filterChain) throws IOException, ServletException {

HttpServletRequest r = (HttpServletRequest) req;

MyRequestWrapper request = new MyRequestWrapper(r);

System.out.println("I18nFilter拦截器正在处理》》》》");

filterChain.doFilter(request, resp);

}

}

**Web.xml**

<filter>

<filter-name>i18nFilter</filter-name>

<filter-class>com.shop.filter.I18nFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>i18nFilter</filter-name>

<url-pattern>\*</url-pattern>

</filter-mapping>

### 生命周期

### 优缺点对比

### 引用

》http://www.runoob.com/jsp/jsp-page-redirect.html

》》学习网站

》https://www.cnblogs.com/haozhengfei/p/6537909.html

》》jsp的使用

》https://www.cnblogs.com/du-0210/p/8426624.html

》》web基础知识 http servlet