# 基础(HTML篇)

## 常用标签

第一个为头标签 第二个(带斜杠的)为尾标签 从头标签开始-到尾标签结束

<HTML> </HTML> //一个网页的开始和结束

<head> </hesd> //网页的头部 用于添加标题

<title> </title> //网页标题 内容会显示在网页的顶端 也就是网页的名字

<body> <body> //主体、内容 将显示在网页中

<h1> </h1> //内容标题 可以更改为1~6 6最小1最大

<p> </p> //一个段落 也就是另起一行

<br/>  //换行 作用和p差不多

<div> </div> //创建一个分区 可以通过style设置大小和其他属性

## Style

style(外联)

连接外部的样式表

<link rel="stylesheet" type="text/css" href="mystyle.css"> //连接一个样式表 href是文件名

style(内联)

添加在头标签里,如:<h1 style="font-family:verdana"> 123456 </h1> //改变颜色

要加 - 号连接属性意思为 哪个部分 – 哪个部位 – 什么属性

1. Text- //文本

align:center //对齐:居中

1. Background- //背景

color:yellow //颜色:黄色

1. font- //字体

family:arial //家族:宋体

color:red //颜色:红色

size:20px //大小:20像素

style(内联表)

可以自定义id 然后设置属性 在任何标签应用

<style type=”text/css”> //创建css样式

.abc{ //创建一个id为abc的样式

Height:200px; //为这个样式添加属性

} //范围是这个大括号

</style> //记得加上尾标签

<div class=”abc”> //创建一个分区 使用abc样式

</div> //css样式根据应用的类型调整,div的类型应用到text类型无效

## 字体格式化

<b> </b> //粗体

<em> </em> //强调

<i> </i> //斜体

<big> </big> //大

<small> </small> //小

<sub> </sub> //下标

<sup> </sup> //上标

<pre> </pre> //适合显示代码 保留空格和缩进

## 超链接

<a href=" "> </a> //超链接 双引号里为地址 标签里为设置超链接的文本

## 注释

<!—注释内容-->

## 其他属性

tiele

<p title="123 "> abc </p > //鼠标移至abc处时显示小文本框123

# 基础(JSP篇)

## jsp注释

隐藏注释： //标准注释，网页不可见，源文件不可见

<%-- 注释内容 --%>

HTML注释： //网页不可见，源文件可见，可加jsp表达式

<!-- 注释内容[<%=jsp表达式%>] -->

Jsp注释： //网页不可见，源文件不可见

<%/\* 注释内容 \*/%> <%// 注释内容 %>

## 常用脚本元素

声明变量： //同时声明一个或多个，任何位置可用，随页面初始化

<%! int i=0; %>

<%! String str = “一二三”; %> //仅在本页面有效

表达式： //输出值

<%=str %>

脚本： //java代码，本页面有效，关闭页面销毁变量

<%

for(int x=0,x<=10,x++) {

}

%>

## **常用指令**

指定整个jsp页面的相关信息，设置属性

page指令： //定义全局属性，作用于整个jsp页面

[language:”java”] //指定jsp使用的脚本语言

[extends:”\*\*\*.class”] //导入需要的class类

[import:”\*\*\*.class|\*\*\*.\*”] //导入所需要的java类，可用多次

[session:”true/false”] //是否使用session对象，布尔值

[buffer:”none/8kb”] //设定页面缓冲区大小，none禁用，默认8KB

[autoFlush: “true/false”] //缓冲溢出是否自动刷新，布尔值

[isThreadSafe: “true/false”] //页面是否能够处理一个以上的请求，布尔值

[info:”text”] //指定页面的相关信息文本

[errorPage:”\*\*\*”] //指定错误页面

[isErrorPage:” true/false”] //页面是否为处理异常错误的页面

[contentType:”mimeType[;charset=characterSet]”] //MIME类型和jsp编码方式

[pageEncoding:”UTF-8”] //指定jap编码方式，支持中文设置UTF-8

taglib指令: //指定页面中使用的标签库以及自定义标签的前缀

<%@taglib uri=”\*\*\*” prefix=”\*\*\*”%> //uri指定标签库存放位置 prefix指定前缀

include指令: //在jsp编译时插入一个静态文件(jsp页面,HTML页面,java程序…)

<%@ include file=”hello.jsp”/> //把hello.jsp导入页面

//指令在页面转换时激活

//动作在请求期间被激活

## 常用动作:

传递参数，动态插入文件，重用JavaBean组件…

<jsp:param>动作 //用于为其他动作提供附加信息

<jsp:param

name=”\*” //name指定参数名称

value=”\*” // value指定参数值

/>

<jsp:include>动作 //把指定文件插入正在生成的页面中

<jsp:include

page=”\*\*\*” //page指定文件绝对路径

fiush=”true” //flush指定在包含目标资源之前是否刷新输出缓冲区,默认false

/> //与include区别:被调用时间

<jsp:uesBean>动作 //装载一个将在jsp页面中使用的JavaBean

<jsp:uesBean

id=”\*” // id:指定该JavaBean实例名称

class=”\*” // class:指定Bean类路径和类名

type=”\*” // type:指定引用该对象的变量的类型

beanName=”\*” // beanName:指定bean的名字,必须和type结合使用,不能喝class同时使用

scope=”page|request|session|appliclication”

scope:指定Bean作用范围,可选值有:page/request/session/application

page:默认值,表示该Bean只在当前页面可用(保存在当前页面的PageContext内)

request:表示该Bean在当前的客户请求内有效(保存在ServletRequest对象内)

session:表示该Bean对当前HttoSession内所有页面都有效,即会话作用域

application:表示该Bean在任何使用相同的application的jsp页面中都有效,即整个应用

/>

<jsp:setProperty>动作 //设置、修改已经实例化的Bean中的属性值

<jsp:setProperty

name=”\*” //必需。表示要设置哪个Bean

property=”\*” //必需。表示要设置哪个属性 \*代表所有和Bean名字匹配的请求参数都将传递给相应的set方法

param=”\*” //可选。用哪个请求参数作为Bean的值

value=”\*” //可选。指定Bean的值

<jsp:getProperty>动作 //获取指定Bean属性的值，转换成字符串，然后输出

<jsp:getProperty

name=”\*” //指定获取属性值的Bean名称

property=”\*” //指定要获取的Bean属性名

/>

<jsp:forward>动作 //引导客户端请求到另一个页面、Servlet

<jsp:forward

page=”\*” //指定另一个页面名称

/>

例题

假定用户名和密码正确，使用<jsp:forward>动作把页面跳转到success.jsp页面，否则进入login.jsp页面。三个页面分别是登录页面login.jsp、数据处理的页面 loginCheck.jsp和成功页面success.jsp。

login.jsp代码如下：

1. <%@page contentType="text/html" pageEncoding="UTF-8"%>

2. <html>

3. <head>

4. <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

5. <title>登录页面</title>

6. </head>

7. <body>

8. <form action="loginCheck.jsp">

9. <table>

10. <tr>

11. <td>输入用户名：</td>

12. <td><input type="text" name="name" size="16">

13. </td>

14. </tr>

15. <tr>

16. <td>输入密码：</td>

17. <td><input type="password" name="password"

18. size="18">

19. </td>

20. </tr>

21. <tr>

22. <td><input type="submit" value="登录"></td>

23. </tr>

24. </table>

25. </form>

26. </body>

27. </html>

# 内置对象

## out对象

用于向客户端发送数据

out.print()/println(); //输出数据，ln表示输出后换行

out.newLine(); //换行

out.flush(); //输出缓冲区的数据,然后清除缓冲

out.clearBuffer(); //清除缓冲数据,不输出

out.getBufferSize(); //获取缓冲区大小

out.Remaining(); //获取缓冲区剩余空间

out.isAutoFlush(); //判断是否自动刷新缓冲区,是true,否false

out.close(); //关闭输出流

## request对象

封装，接收客户端通过HTTP协议传送给服务器的数据

request.getParameter(String name);

//用于获取客户端传送给服务器端的指定请求参数的值,返回值是字符串,参数不存在则返回null.

request.getParameterNames();

//用于获取客户端传送给服务器端的所有请求参数名称，其结果 是一个Enumeration类的实例。

request.getParameterValues(String name)

//用于获取客户端传送给服务器端的指定请求参数的 所有值，返回值是字符串数组。 4. request.getAttribute(String name)

//用于获取request对象中指定属性的值，若指定属性不存在，就返回null。

request.setAttribute(String name, Object obj)

//用于设置request对象中的属性及其属性值。

request.removeAttribute(String name) ;

用于删除request对象中的指定属性及其属性值。

request.setCharacterEncoding(String env);

request.getCharacterEncoding();

//分别用来设置和获取客户端请求的字符编码方式。

request.getContentLength();

//用于以字节为单位获取客户端请求的大小， 如无法得到该请求的大小，则返回-1.

request.getContentType(); //用于获取客户端请求的类型。

request.getContextPath(); //用于获取Context路径。

request.getLocalAddr(); //方法用于获取服务器的IP地址。

request.getLocalName(); //获取服务器的名称。

request.getMethod(); //用于获取客户端向服务器端传送数据的方法。如 post或者get。

request.getProtocol();

//用于获取客户端向服务器端传送数据所使用的协议名称和版本。

request.getRemoteAddr(); //方法用于获取客户端的IP地址。

request.getRemoteHost(); //方法用于获取客户端的名字。

request.getRemotePort(); //用于获取客户端的端口号。

request.getRemoteUser(); //用于获取客户端的验证用户名。

request.getRequestedSessionId(); //用于获取客户端定制的会话ID。

request.getRequestURI(); //用于获取客户端请求除去域名部分的路径。

request.getRequestURL(); //用于获取客户端请求的完整路径。

request.getServerName(); //用于获取服务器的名字。

request.getServerPort(); //用于获取服务器的端口号。

request.getServletPath(); //用于获取客户端所请求的文件路径。

request.getSession();

//用于获取和此请求关联的session对象。session对 象的getId（）方法用于获取session对象的ID.

request.isRequestedSessionIdFromCookie();

//用于判断请求的会话ID是否 通过Cookie传入，如果是返回true，否则返回false。

request.isRequestedSessionIdFromURL();

//用于判断请求的会话ID是否通 过URL传入，如果是返回true，否则返回false。

request.isRequestedSessionIdValid();

//用于判断请求的会话ID是否仍然有 效，如果有效返回true，否则返回false。

request.isSecure();

//用于判断客户端请求是否是使用安全通道（如https） 发出的，如果是返回true，否则返回false。

## response对象

将服务器响应数据发送到客户端,与request结合发送返回.

response.addCookie(Cookie cook);

//用于添加Cookie对象保存客户端的相关信息

//可以使用request对象的getCookies()方法获取该Cookie对 象。

response.addHeader(String name，String value); //用于添加带有指定名称和字符串的响应头。

response.addDateHeader(String name，String value); //用于添加带有指定名称和日期值的响应头。

response.containsHeader(String name);

用于判断指定名称的响应头是否存 在，存在返回true，否则返回false。

response.flushBuffer()： //用于强制把当前缓冲区所有内容发送到客户端。

response.getBufferSize()： //用于获取实际缓冲区的大小，如果没使用缓冲区 则返回0。

response.getCharacterEncoding()： //用于获取响应的字符编码方式。

response.getContentType()： //用于获取响应的MIME类型。

response.getOutputStream()： //用于获取到客户端的输出流。

response.sendError()： //用于向客户端发送错误信息。如：404指网页找不到 错误。

response.sendRedirect()： //用于重新定向客户端的请求。

response. response.setCharacterEncoding()： //用于设置响应的字符编码方式。

response.setContent()： //用于设置响应的内容MIME类型。

response.setContentLength()： //用于设置响应内容的长度（字节数。）

response.setHeader();

//设置指定名称和字符串的响应头，该Header信息将传达到客户端，不存在就设置，存在则覆盖。

response.setDateHeader();

//设置指定名称和日期值的响应头，该Header 信息将传达到客户端，不存在就设置，存在则覆盖。

## session对象

//客户端与服务器的一次会话,从客户端连到服务器开始，直到客户端与服务器断开连接为止。

//用户首次登录系统时容 器会给用户分配一个唯一的session id标识用于区别其他的用户

session.setAttribute(String name,Object value);

//用于设定指定名字的属性 值，并且把它存储在session对象中，如果属性已存在，则更新属 性的值。

session.getAttribute(String name);

//用于获取session对象中指定属性的值， 如果属性不存在，将会返回null。

session.removeAttribute(String name);

//用于删除session对象中的指定属性 （包含属性名和属性值）。

session.getAttributeNames();

//用于返回session对象中所有属性的名称，结 果是一个枚举类型的对象。

session.getLastAccessedTime();

//用于返回和当前session对象相关的客户端 最后发送请求的时间。

session.setMaxInactiveInterval(int interval);

//用于设置session的有效时间， 单位为秒。

session.getMaxInactiveInterval();

//用于返回session对象的有效时间，单位 为秒。

session.invalidate();

//用于使session对象失效，存放在session对象中的所有 对象都不能再被访问。

session.isNew();

//用于判断目前session是否为新的Session，若是则返回 true，否则返回false。

session.getCreateTime();

//用于返回session对象被创建时间时间，单位为毫 秒（千分之一秒）。

session.getId(); //用于返回session对象的ID。

## pageContext对象

对JSP页面的所有对象 及命名空间的访问

pageContext. setAttribute(String name,Object attribute);

//用于在页面范围内设置指定 属性及属性值。

pageContext. setAttribute(String name,Object obj,int scope);

//用于在指定范围内设置指定属性及属性值。

pageContext. getAttribute(String name,int scope); //用于获取指定范围内指定属性的值。

pageContext. getAttribute(String name); //用于获取页面范围内指定属性的值。

pageContext. getOut(); //用于返回当前的out对象。

pageContext. getPage(); //用于返回当前的page对象。

pageContext. getRequest(); //用于返回当前的request对象。

pageContext. getResponse(); //用于返回当前的response对象。

pageContext. getSession(); //用于返回当前的session对象。

pageContext. getServletConfig(); //用于返回当前的config对象。

pageContext. getException(); //用于返回当前的exception对象。

pageContext. getServletContext(); //用于返回当前的application对象。

pageContext. findAttribute(String name);

//用于按照页面、请求、会话以及应用 程序范围的顺序实现对指定属性的搜索，返回其属性值或null。

pageContext. forward(java.lang.String relativeUrlPath);

//用于把页面重定向到另 一个页面或者Servlet组件上。

pageContext. removeAttribute();

//用于删除所有对象范围或特定对象范围中的指 定属性及其属性值。

pageContext. release(); //用于释放pageContext所占资源 。

pageContext. include(String relativeUrlPath); //用于在当前位置包含另一文件。

## exception对象

处理JSP文件在执行时 发生的错误和异常。

exception对象需要配合page指令一起使用，在page 指令中errorPage属性指定某一个页面为错误处理页面， 把所有的错误都集中该页面进行处理，错误页面中page 指令isErrorPage属性应设为true

exception. getMessage()： //用于返回描述异常错误的提示信息。

exception. getlocalizedMessage()： //用于获取本地化错误信息。

exception. printStackTrace()： //用于输出异常对象及其堆栈跟踪信息。

exception. toString()： //返回关于异常的简短描述消息。

## application对象

application对象保存Web应用程序中公有的数据， 可存放全局变量。服务器启动后自动创建application 对象，此对象将一直有效，直到服务器关闭。

application. getAttribute(String name); //用于返回application对象中的指定属性 的值

application. getAttributeNames();

用于以Enumeration类型返回application对象中的所有属性的名字

application. getServerInfo(); //用于返回servlet容器名称和版本信息

application. getContext(String uripath); //用于返回指定WebApplication的 application对象

application. getMimeType(String file); //用于返回指定文件的MIME类型

application. getResource(String path); //用于返回指定资源(文件及目录)的URL 路径

application. getServlet(String name); //用于返回application对象指定Servlet

application. setAttribute(String name,Object obj);

//用于设定application对象中的 指定属性及其属性值

application. removeAttribute(String name); //用于删除application对象中的指定 属性及其属性值

application.log(String msg); //用于把指定消息写入Servlet的日志文件

# 连接SQL

下载SQL，将JDBC(下载对应版本的jar)放入Tomcat-lib中

## 导入和加载SQL

导入

<%@ page import=" java.sql.\*"%> //导入类包

加载

Class.forName("com.mysql.jdbc.Driver"); //加载JDBC驱动

String SQLurl = "jdbc:mysql://localhost:3306/ltb"; //要使用哪个库的数据,我这里是ltb库

String username ="root"; //数据库用户

String password ="2539616768";//密码

Connection conn = DriverManager.getConnection(SQLurl, username, password); //开始连接