Java Web的开发不是去做网站,局限性大 .而是写一些服务器的页面(CRM 、ERP、OA 定制web界面).

网页设计：

静态网页 :HTML ,CSS ,JavaScript 静态网页开发技术

动态网页 :98前后 ASP **PHP** JSP .net(微软技术集合 VB ASP C#) 、**python**、ruby(快速开发网页)

当今网站开发 主流技术 : **python** ,php ,H5, ASP.net

静态web资源 :html css javascript 图片 音频 视频 文本

动态web资源 :程序Servlet、JSP

# tomcat

Tomcat Servlet/JSP版本 JavaEE版本 运行环境

4.1 2.3/1.2 1.3 JDK1.3

5.5 2.4/2.0 1.4 JDK1.4

6.0 2.5/2.1 5.0 JDK5.0

7.0 3.0/2.2 6.0 JDK6.0

Tomcat 支持Servlet和JSP规范，Servlet技术属于JavaEE规范

\* 随着javaEE版本提升--> Servlet版本提升--> 运行Servlet环境Tomcat版本提升

\* tomcat运行需要JDK 环境版本

Tomcat目录结构

bin ---- 存放tomcat启动关闭程序

conf --- 存放tomcat 配置文件

lib --- tomcat运行需要jar包

logs ---- tomcat日志文件

webapps ---- 网站发布目录 （所有网站可以发布到该目录）

work ----- 存放工程运行时，产生数据文件 (JSP翻译Servlet、Session持久化数据 )

网站开发好后，如果想让网站被外界访问，需要将网站发布到web服务器tomcat上(虚拟目录映射)

网站的标准目录结构

|---站点根目录main

|---静态web资源、html、css、jsp

|---WEB-INF目录

|--classes目录 （保存,class文件）

|--lib 目录 (当前网站需要jar包)

|-- web.xml (网站配置文件)

\* WEB-INF目录不是必须的，没有java动态程序代码， 可以没有WEB-INF 目录

\* WEB-INF目录下资源不能被浏览器直接访问

# hosts应用

1、限制用户上网 ，网站禁止访问

2、软件破解

# MyEclipse 整合 tomcat 进行 web 程序开发

配置tomcat:

1. 配置myeclipse 内部tomcat 插件

配置tomcat6 在 window >preferences >tomcat6 检查JDK 版本是否符和,Launch 默认采用 debug model

2． 将tomcat插件设置为enable 选择tomcat目录后 配置ok

3. MyEclipse中通过插件 tomcat6 --- start实际上 运行tomcat/bin/startup.bat

4． 在console窗口 查看tomcat日志

\* tomcat6 --- stop server 正常关闭tomcat 调用 tomcat/bin/shutdown.bat

\* 如果直接点击 myeclipse console 窗口 红色按钮 --- 强行停止tomcat

在myeclipse中编写一个web工程

1. 创建web project -->context root url 默认工程名

Javaee 规范版本 ,在使用javaee6 规范版本时，默认支持jstl技术

2. 新建html 将html编辑器 设置 html editor

window -- preference ---- general ---- editor --- File Associations

1. 发布工程到tomcat -->原理：将当前web工程复制tomcat/webapps
2. 修改 发布虚拟目录名称 --- 右键工程属性 --- myeclipse ---web ----修改Web Context-root

# Http协议

HTTP/1.0、HTTP/1.1 区别？

1.0 一次 与服务器连接 只能获得一个资源

1.1 一次与服务器连接 ，连续获得多个资源

# 第一部分 HTTP 请求

请求完整格式 包括三个部分 ： 请求行、头信息、请求体

1. 请求行 (**GET / HTTP/1.1**)包括三个部分 请求方式、资源路径、HTTP协议版本

HTTP请求方式有七种 :POST/GET/HEAD/OPTIONS/DELETE/TRACE/PUT

常用两种 :GET/POST

GET方式 ：

1.用户直接在浏览器上手动输入url地址

2.<a href="url"></a>

3.<form method="get"> ,form表单默认提交方式就是get

POST方式 :

1.<form method="post">

GET提交方式和POST提交方式区别 ？

Get url地址后携带数据有大小限制 1K ,提交数据在url上显示

http://localhost/day04/url?username=zhangsan

格式 :url?name=value&name=value&name=value 以？分隔url和参数,参数采用键值对格式,多个参数使用& 符合分隔

POST 数据在请求体中,不在url上显示 ,没有数据大小限制

username=zhangsan 请求体和头信息直接存在一个空行

1. request请求头信息 : key -value

Accept: text/html,image/\* //客户端可以接受html文件,image/\* 任意格式图片

Accept-Charset: ISO-8859-1 //客户端可以识别编码字符集

Accept-Encoding: gzip -->客户端可以识别压缩数据格式 gzip是一种压缩格式

Accept-Language:zh-cn -->客户端浏览器语言

Host: www.itcast.com:80 -->访问服务器地址

If-Modified-Since: Tue, 11 Jul 2000 18:23:51 GMT -->该请求的资源在客户端保存最后访问时间(缓存有关)

Referer: http://www.baidu.com/ -->上一次请求访问页面地址

User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0) -->客户端浏览器类型版本

Connection: close/Keep-Alive -->该次请求连接后，连接是保持还是关闭 1.0 关闭 1.1 保持

Date: Tue, 11 Jul 2000 18:23:51 GMT ---- 请求时间

重要的请求头信息: Accept-Encoding ,If-Modified-Since ,Referer

Referer 防止客户盗链数据 :通过google搜索页面直接下载 其它网站数据 （盗链）

1. 请求体： 存放客户端提交post 请求 数据

key=value&key=value&key=value ....

例如 ： username=sss

# 第二部分 :HTTP响应

响应数据 分为三个部分: 状态行 ,头信息 ,响应体

1. 状态行分为三个部分(HTTP/1.1 200 **OK**)： HTTP协议版本 ,状态码 ,描述信息;

100～199 Informational 请求信息不足 ，还需要其他信息才能处理

200～299 Success 请求处理成功

300～399 Redirection 服务器已经处理完毕，客户端还需要进一步动作

400～499 Client Error 客户端错误

500～599 Server Error 服务器端错误

200 请求处理成功 ,206 请求部分数据成功的响应码 ,302客户端进行重定向(跳转,两次请求,响应),304客户端访问资源没有被修改过 ,客户端访问本地缓存 ,404访问资源不存在 ,500 服务器内部出错

1. 响应头

* Location: http://www.it315.org/index.jsp -->重定向地址,结合302一起使用
* Refresh: 1;url=http://www.baidu.com --> 1秒后自动跳转到百度

Server:apache tomcat -->服务器类型

Content-Encoding :gzip-->设置响应数据压缩格式 -->请求 Accept-Encoding

\* 如果响应数据经过压缩 传输体积更小，性能更好

Content-Length: 80 --> 响应数据长度

Content-Language: zh-cn -->响应数据语言

* Content-Type: text/html; charset=GB2312 -->响应数据类型

\* 数据类型采用Mime协议规定类型; html文件,text/html;jpg文件 ,image/jpeg

\* 在 tomcat/conf/web.xml 中查看文件mime类型

获取文件类型getServletContext().getMineType(“文件名”)

Last-Modified: Tue, 11 Jul 2000 18:23:51 GMT ,和请求头信息 If-Modified-Since一起使用，控制服务器缓存

Content-Disposition: attachment; filename=aaa.zip -->文件下载时指定文件附件名称

Expires: -1 //设置缓存的日期.response.setDateHeader(“Expires”, “-1”)

Cache-Control: no-cache

Pragma: no-cache -->对于动态程序，经常修改， 禁止浏览器缓存内容

Connection: close/Keep-Alive --> 响应后连接是否关闭

Date: Tue, 11 Jul 2000 18:23:51 GMT -->响应时间

# 响应体

通常HTML 文件内容--> 如果设置响应体进行gzip压缩，看到gzip压缩内容

# JavaWeb之Servle技术篇

servlet技术基于Request-Response编程模型 ,与Http协议相关

可以用来开发基于HTTP web 应用程序

Servlet继承关系

|-- Servlet接口

|--GenericServlet

|--HttpServlet 引入与 协议相关 API

Servlet快速入门

1. 编写 class 继承 HttpServlet

2. 在web.xml 配置 Servlet程序虚拟访问路径 ,可以通过这个路径访问Servlet程序

3. 覆盖doGet或者doPost方法 进行输出

Servlet 动态生成网页 ,文件的执行过程

1. 客户端发起url请求: http://localhost/day05/hello -->web.xml/hello 映射 HelloServlet程序

2. get方式提交执行 HelloServlet的doGet方法 ,post方式提交执行doPost()方法

# URL-pattern 配置格式 三种：

一个Servlet可以配置多个url-pattern

1. 完全路径匹配 (以/开始 ) 例如：/hello /init

当前工程没有被正确发布，访问该工程所有静态资源、动态资源 发生404 ----- 工程启动时出错了

(调式)查看错误时 分析错误

1) 单一错误:从上到下查看第一行你自己写代码(有的错误与代码无关,查看错误信息）

2）复合错误:Caused by ,查看最后一个Caused by ,并调试

2. 目录匹配 (以/开始) 例如：/\* /abc/\*

/ 代表网站根目录 \*代表任意内容

3. 扩展名(不能以/开始)例如：\*.do 以.do结尾的任意uri(包括///)

优先级：完全匹配>目录匹配 > 扩展名匹配

ServletConfig对象

1. 通过GenericServlet类继承的方法getServletConfig() 获得ServletConfig对象,进而获得Servlet初始化参数

2. 在web.xml 中 <servlet> 标签内 通过 <init-param> 标签 为Servlet配置初始化参数

<init-param>

<param-name>itcast</param-name>

<param-value>传智播客</param-value>

</init-param>

3. 在Servlet程序中通过ServletConfig对象 获得name对应数据

getInitParameter()通过name获得value

getInitParameterNames()获得所有name

# ServletContext :上下文对象

每一个工程 对会创建 单独ServletContext对象，这个对象代表当前web工程

操作ServletContext 必须通过ServletConfig 获得对象

应用:

1. 获得整个web应用初始化参数 :<context-param>

2. 实现全局数据共享,<context-param>保存的数据,所有Context对象都可以访问

3. 实现服务器端转发功能

通过getRequestDispatcher(“/b”)方法 ,完成转发

转发与重定向的区别:

* + - 请求,响应次数不同.转发一次请求,多次响应. 转发性能好与重定向
    - 转发url不变,重定向显示新的url
    - 开发优先使用转发

4. 读取web工程资源文件

//servlet读取文件,先取出文件相对硬盘的真实路径.

String fileName = getServletContext().getRealPath("/WEB-INF/2.txt");

**例子:统计文本域中字母的出现次数**

**//知识点点,转发的写法和servletcontext保存全局数据**

**public** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

String content = request.getParameter("content").toUpperCase();

//字母统计次数

**int**[]times = **new** **int**[26];

// 遍历每一个字母

**for** (**int** i = 0; i < content.length(); i++) {

**char** c = content.charAt(i);

// 判断字母是不是26个字母之一

**if** (Character.*isLetter*(c)) { // 取代c >= 'A' && c <= 'Z'

// 如果 c 是 'A' 在数组 0 位置 +1 ，如果 c是 'B' 在数组1位置+1

times[c - 'A']++;

}

}

//1. 将数据保存到servletContext

ServletContext context = getServletContext();

context.setAttribute("times", times);

//2. 转发跳跳转到另一个servlet,并显示

RequestDispatcher dispatcher =context.getRequestDispatcher

("/servlet/letterresult");

dispatcher.forward(request, response);

}

DefaultServlet 缺省Servlet 作用 :将客户端请求的静态资源返回!

# HttpServletRequest 和HttpServletResponse(接口)

客户端每次请求都会创建request和response对象(service)传递doGet(),doPost()

|--ServletResponse

|--HttpServletResponse

|--tomcat服务器提供具体的实现类.

# response常用 API

getScheme() 获取请求协议

setStatus() 向客户端设置不同的响应码

setHeader() 设置响应头信息

getOutputStream() 获得字节流,输出响应体内容

getWriter() 获得字符流,输出响应体内容

setContentType() 设置响应的数据格式编码,html文件的编码

response.setContentType("text/html; charse=utf-8");

response.setCharacterEncoding("gbk"); //设置写入的字符编码

**//案例 : 判断登录界面,正确重定向到新的网页. --?学习重定向到的方法**

**public** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

String user = request.getParameter("username");

String password= request.getParameter("password");

**if** ("admin".equals(user)&&"admin".equals(password)) {

//重定向第一种方法

response.setStatus(302);

response.setHeader("Location", "/day06/welcome.html");

}**else** {

//第二种重定向,替代302+Location

response.sendRedirect("/day06/login.html");

}

}

# HttpServletRequest 分为四个部分

# request 常用API

request.getMethod(); //获取请求方式

request.getQueryString();//获取请求行中的参数

request.getRequestURL(); //获取请求的完整URL

request.getRequestURI(); //获取URi

request.getContextPath(); //获取工程虚拟目录名 /day

request.getRemoteAddr(); //获取客户机ip

request.setAttribute(new String(), new Object());//设置数据,转发

获得当前访问资源路径(/heelo) URI-/day: request.getRequestURI().substring(request.getContextPath().length());

1. 获取客户机信息

* URI和URL区别

url:http://localhost/day06/request1 --- 完整

uri:/day06/request1 ---- 部分

URI包含URL() ,URL一定完成路径 ,URI可以相对路径

http://localhost/day06/request1是一个URL同时也是 URI

./hello /day06/request1 //都是URI 不是URL

1. 获取请求头信息

getHeader 获得头信息的值，转换一个字符串

getHeaders 获得头信息值 ，获得Enumeration

getHeaderNames 获得所有头信息名称 返回 Enumeration

//获取指定头信息

String value = request.getHeader("Accept-Encoding");

//遍历Enumeration 获取头信息

Enumeration <String> e = request.getHeaderNames();

**while** (e.hasMoreElements()) {

String name = e.nextElement();

sop(request.getHeader(name));

1. 获取请求参数

GET/POST提交的数据

* /day06/request4?name=zhangsan&city=beijing包括两个参数的name和city
* <a href="url?xxx=xxx">
* <form action="/day06/request4" method="get"> <input name="name"/>
* <form method="post">

获取参数的API

String value = request.getParameter("username");

String [] values = request.getParameterValues();

String []names = request.getParameterNames();

Map<String String[]> map = request.getParameterMap();

//非空校验if (username != null && username.trim().length() > 0) {}

# 乱码问题(请求的参数默认进行URL编码,服务器就可能出现乱码)

提交请求优先使用post

post提交方式 request.setCharacterEncoding("编码"); //设置和请求编码一直

get乱码手动解决:

username = URLEncoder.encode(username, "ISO-8859-1");// 用ISO编码

username = URLDecoder.decode(username, "utf-8"); // 用utf-8解码

简化写法 ：username = new String(username.getBytes("ISO-8859-1"), "utf-8");

1. 利用请求域传递对象

HttpServletRequest和ServletContext 类似,都是数据域对象,以Map方式保持数据

区分：存活时间不同

ServletContext对象,服务器启动对象创建,服务器停止对象销毁

ServletRequest对象,当产生一次请求时创建,当响应结束后,对象销毁

通过request转发请求,request保存数据进行Servlet之间传递 ---- 应用？

Servlet进行数据处理 --- 生成结果 --- 转发结果给JSP显示

注意事项

1. 在使用forward之前,不能将响应内容传输到客户端

情况一 response输出流执行flush

情况二 同一个Servlet不能连续使用forward 和 redirect

2. 在执行forward 和 redirect时，清除之前写入响应流数据

3. ServletContext进行转发 路径必须/开始，request进行转发路径可以使用相对路径

# 转发和重定向区别

1. 转发一次请求、一次响应 .重定向 ,两次请求,两次响应

2. 转发只能跳转站内程序,重定向定向任何站点

3. 转发URL地址不变,重定向URL地址改变

4. 转发对客户端不可见,重定向对客户端可见

5. 转发共享同一Request中数据，重定向两次请求，不同Request对象，不能共享Request数据;request.setAttribute 必须和 request.getRequestDispatcher().forward 一起使用

HTML 页面中存在一类非常特殊标签 <meta>

<meta> 起到设置头信息作用

<meta http-equiv="content-type" content="text/html; charset=GBK"> //设置头信息,包括request的请求编码和reponse的打印编码

<meta content="3;url=/day06/result.html" http-equiv="refresh"> //完成自动跳转