JSP/Servlet

讲义1

第1章 开发环境配置

网络名词介绍

- Web：网络，互联网等技术领域
- TCP/IP ：**Transmission Control Protocol/Internet Protocol**)的简写，中文译名为**传输控制协议/网际协议**，又叫网络通讯协议，这个协议是是Internet最基本的协议、Internet国际互联网络的基础
- HTTP：**超文本传输协议**（Hypertext Transfer Protocol）
- URL:**统一资源定位符**（URL，英语Uniform Resource Locator的缩写）, Internet上的每一个网页都具有一个唯一的名称标识，通常称之为URL地址，俗称“网址”。
- 端口：如果把IP地址比作一间房子，端口就是出入这间房子的门。真正的房子只有几个门，但是一个IP地址的端口 可以有65536（即：256×256）个之多！端口是通过端口号来标记的，端口号只有整数，范围是从0 到65535（256×256-1）。

安装Tomcat

- Tomcat是一个传动的Web服务器软件，能够接收HTTP请求，并且可以构造一个HTTP响应
- 是一个Apache基金会的开源项目，可以免费下载使用
- 可以运行Servlet和JSP程序，也称为Servlet容器和JSP容器

tomcat安装目录中的子目录

- bin：此目录下包含了启动，关闭和其他一些程序；
- lib存放公用的web应用的类库，任何项目都可以使用这些类库
- conf:包含配置文件和相关DTD文件，其中非常重要的一个是tomcat的主要配置文件server.xml；
- logs:存放日志文件；
- webapps：存放应用程序示例，以后部署的应用程序需要放在此目录；
- work：tomcat运行时生成的临时文件，包括jsp编译后产生的class文件等。

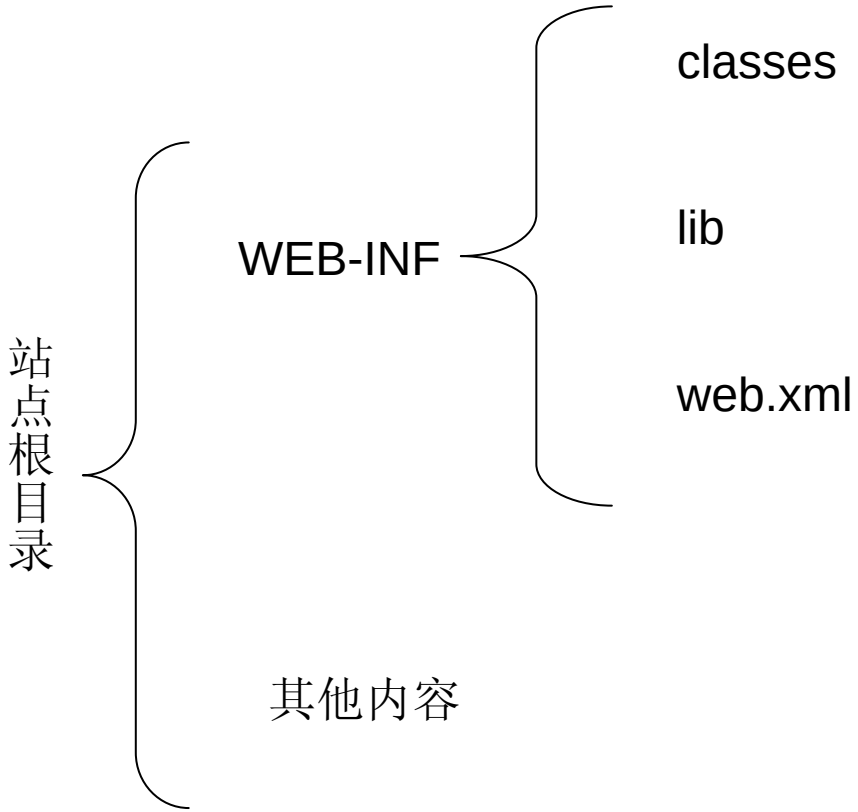
Tomcat和Servlet和JSP的对应版本关系

Tomcat	Servlet	JSP
6.0	2.5	2.1
5.0	2.4	2.0
4.0	2.3	1.2

开发工具的编码设置

- 如果没有正确的设置开发工具的编码，会导致文件乱码，应该在开发前统一编码，例如GBK或者UTF-8

WEB应用程序基本结构

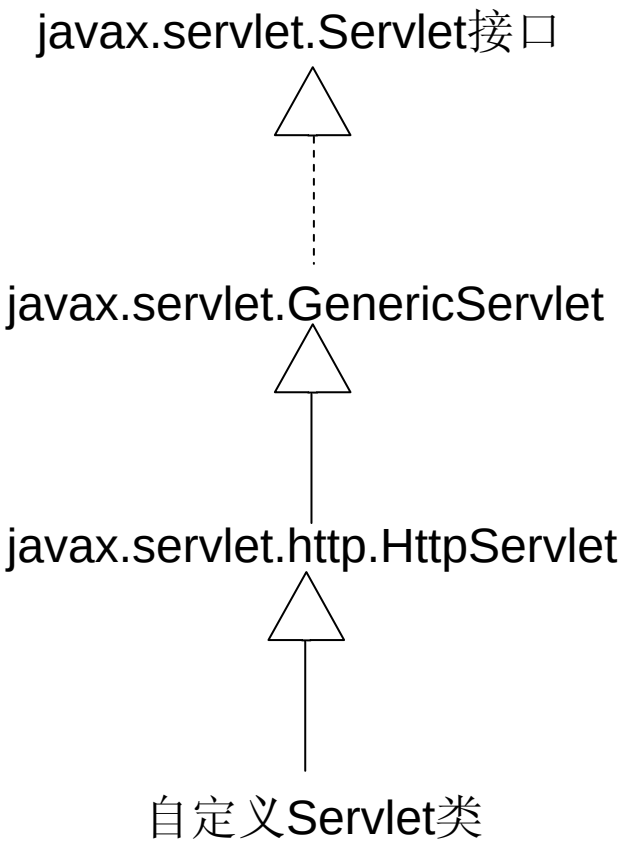


第2章 Servlet基础

Servlet

- Servlet是服务器端的组件技术，可以让客户端执行服务器端的代码，实际上servlet也就是一个java的类，只不过是客户端可以连接服务器在服务器端执行这些类的代码。
- 任何Servlet类都必须直接或者间接实现javax.servlet.Servlet接口，在这个Servlet接口中只定义了5个方法init getServletConfig service getServletInfo destroy 没有任何和http协议相关的内容。
- java提供了一个实现了Servlet接口的类：javax.servlet.GenericServlet类，这是一个抽象类，但仍然和协议无关，所以JavaEE提供一个继承了此类的子类javax.servlet.http.HttpServlet里面提供针对于http协议使用的方法，但这个类仍然是一个抽象类不能直接创建实例。
- 如果某个类要成为Servlet，则它应该从HttpServlet 继承，根据数据是通过GET还是POST发送，覆盖doGet、doPost方法之一或全部。doGet和doPost方法都有两个参数，分别为HttpServletRequest(请求对象) 类型和HttpServletResponse (响应对象)类型。

Servlet继承图


javax.servlet.Servlet接口
javax.servlet.GenericServlet
javax.servlet.http.HttpServlet
自定义Servlet类

第一个Servlet

```
import java.io.*;
import javax.servlet.http.*;

public class FirstServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
                        throws IOException {
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>HELLO, WORLD</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("hello,world");
        out.println("</body>");
        out.println("</html>");
    }
}
```

配置web.xml部署描述文件

- 加入以下标签

```
<servlet>
        <servlet-name>FirstServlet</servlet-name>
        <servlet-class>com.china.soft.FirstServlet</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>FirstServlet</servlet-name>
        <url-pattern>/FirstServlet</url-pattern>
</servlet-mapping>
```

访问Servlet

- 在将WEB应用程序站点部署之后
- 打开浏览器，在地址栏输入 http://127.0.0.1:7777/TestJSP/FirstServlet 回车

Servlet的生命周期

1. 当第一次有人访问这个servlet时首先调用构造起创建实例
2. 然后开始生命周期，执行init方法，执行初始化操作
3. 产生两个对象（request请求对象和response响应对象）调用service方法
4. service方法根据请求方式的不同调用doGet或者doPost方法
5. 当有人再次调用同样的servlet重复第3步到第5步，不会再创建同样的Servlet对象实例
6. servlet实例会常驻内存（持久性），保证多个请求使用的是一个对象实例，当对象成为垃圾对象，被gc回收时，会调用destroy方法，在tomcat中一个servlet被创建实例后只有服务器重启才会成为垃圾对象

- 其中init,service,destroy三个方法被称为Servlet的生命周期

关于doGet和doPost方法

- doGet方法对应的客户端发起的get请求
- doPost方法对应的是客户端发起的post请求

- 其中除了设置<form method=“post”>提交表单时post请求之外，其他的如<form method =“get”>或者直接通过URL访问Servlet均属于get请求
- 我们应该覆盖doGet或者doPost方法来实现功能，而尽量不去覆盖init,service方法

get和post两种请求方式

	GET	POST
传递数据的方式	追加在URL后面	在请求正文中作为消息主体一起发送
安全性	不安全	安全
数据大小	受到URL长度限制，多了丢失数据	长度几乎无限制
字符集	一种特殊的URL编码	ISO

单实例多线程

- 只有第一次访问Servlet的时候会创建Servlet对象的实例
- 当其他客户端访问相同的Servlet的时候，均使用的是同一个对象，每一个客户端都是一个独立的线程

第3章 请求对象和响应对象

请求对象：**HttpServletRequest**接口

- **javax.servlet.http.HttpServletRequest**接口
- 这个接口继承了**javax.servlet.ServletRequest**接口，包含了所有请求的信息，例如请求的来源，报头信息，客户端传给服务器端的参数数据等等。

HttpServletRequest接口常用方法

- public String getParameter(String key)
  - 获得指定参数名对应的值，需要注意的是如果参数不存在，返回一个null值，单选按钮和复选按钮如果没有选择任何值返回null，其他表单控件返回空字符串。
- public String[] getParameterValues(String key)
  - 如果一个参数可以返回多个值，比如复选框集合，则可以用此方法获得对应参数的所有值，复选按钮如果没有选择任何值返回null
- public Enumeration getParameterNames()
  - 此方法返回一个Enumeration对象，包含对应请求的所有参数名字列表

HTTP请求

- 客户端向服务器端发起的http请求由三部分组成
  - 请求行
  - 消息报头
  - 请求正文

HTTP请求——请求行

- 请求行包括方法符号和URI和协议的版本，例如
  - GET /J5EE/ThreeServlet?deptno=10 HTTP/1.1（CRLF换行）
  - POST /J5EE/Request1Servlet HTTP/1.1（CRLF换行）

HTTP请求——消息报头

- 消息报头包括很多客户端的信息，例如

accept=image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/msword, application/vnd.ms-powerpoint, application/vnd.ms-excel, */*
referer=http://localhost:7777/J5EE/api/Header.html
accept-language=zh-cn
content-type=application/x-www-form-urlencoded
accept-encoding=gzip, deflate
user-agent=Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
host=localhost:7777
content-length=148
connection=Keep-Alive
cache-control=no-cache

常见消息报头解释

- **Cache-Conctrl** 缓存指令一般设置为no-cache
- **connection** 连接形式 最常见Keep-Alive连接形式
- **accept** 表示客户端可以接收哪些类型的信息 例如text/html表示接收html，*/*表示任何类型
- **accept-encoding** 客户端支持的编码
- **accept-language** 客户端语言种类，可以在internet选项——“语言”中设置，可以判断这个值来决定显示什么语种的网页
- **host** 请求资源的主机和端口号
- **user-agent** 获得客户端操作系统信息和浏览器信息
- **referrer** 页面来源
- **content-length** 发送的请求正文的长度

HTTP请求——请求正文

- **请求正文** 就是请求最后的内容，如果用POST发送数据的话可以看到请求正文的内容，用的是ISO编码，GET方法发送数据没有请求正文，GET的传递的数据是追加在URL后面的，并且都是要先转换为ASCII编码
- 例如 username=tom&age=27&schoolLevel=%E5%88%9D%E4%B8%AD&skill=java&skill=c%2B%2B&skill=delphi&myInfo=hello

request对象-请求报头相关的方法

- String getHeader(String name)
  - 取得name报头的值
- Enumeration getHeaderNames()
  - 取得所有报头的名字

显示所有的请求报头信息

```
Enumeration e = request.getHeaderNames();

while (e.hasMoreElements()){
        String key = e.nextElement().toString();
        String value = request.getHeader(key);
        System.out.println(key + "=" + value);
}
```

request对象的其他方法

```
// 获得WEB应用路径
out.println("WEB应用路径：" + request.getContextPath());
out.println("<br>");
//获得请求方法
out.println("请求方法：" + request.getMethod());
out.println("<br>");
//获得协议名
out.println("协议名：" + request.getProtocol());
out.println("<br>");
//获得参数字符串
out.println("参数字符串：" + request.getQueryString());
out.println("<br>");
//获得请求URI
out.println("请求URI：" + request.getRequestURI());
out.println("<br>");
```

request对象的其他方法

```
//获得请求URL
out.println("请求URL：" + request.getRequestURL());
out.println("<br>");
//获得用户IP地址
out.println("客户端IP：" + request.getRemoteAddr());
out.println("<br>");
//获得服务器IP
out.println("服务器器IP：" + request.getLocalAddr());
out.println("<br>");
//获得服务器主机名
out.println("服务器主机名：" + request.getLocalName());
out.println("<br>");
//获得服务器端口
out.println("服务器端口：" + request.getLocalPort());
out.println("<br>");
```

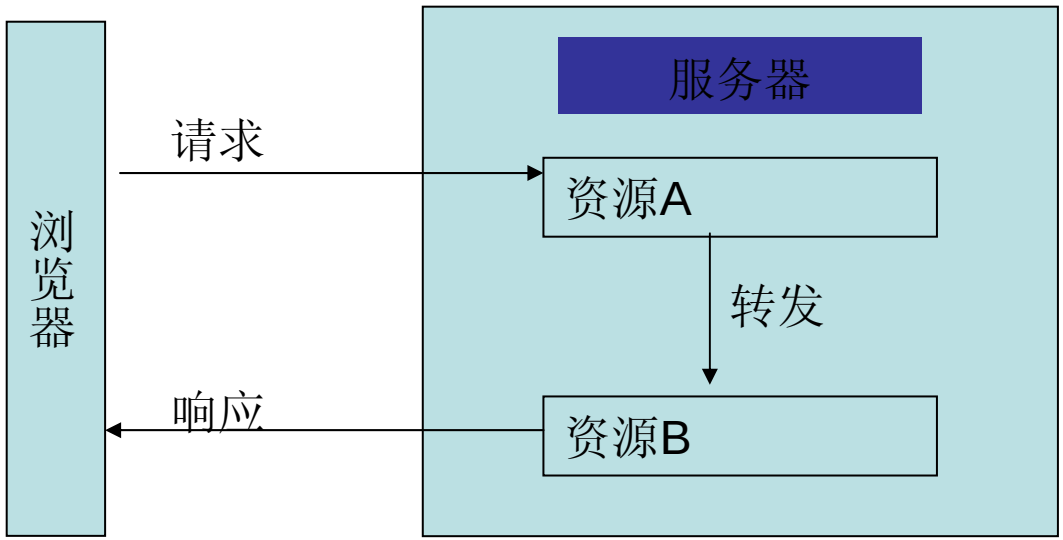
request对象的其他方法

- 打开浏览器输入 http://localhost:7777/J5EE/RequestOtherMethod?id=5&name=tom
- 输出结果

  WEB应用路径：/J5EE
  请求方法：GET
  协议名：HTTP/1.1
  参数字符串：id=5&name=tom
  请求URI：/J5EE/RequestOtherMethod
  请求URL：http://localhost:7777/J5EE/RequestOtherMethod
  客户端IP：127.0.0.1
  服务器器IP：127.0.0.1
  服务器主机名：localhost
  服务器端口：7777

请求转发

- 请求转发允许把请求转发给同一个应用程序中的其他Web组件。这种技术通常应用于Web应用中的Servlet流程控制器
- 值得注意的是，只能把请求转发给同一个web应用中的组件
- 对于请求转发，转发的源组件和目标组件共享request范围内的共享数据。



javax.servlet.RequestDispatcher

- RequestDispatcher类型的对象提供一个forward方法来实现转发功能
- 获得RequestDispatcher的两种方法
  - request.getRequestDispatcher(“转发URL");
  - this.getServletContext().getRequestDispatcher(“转发URL”)

- 示例

```
// 获得转发对象
RequestDispatcher rd =
    request.getRequestDispatcher("/Forward2Servlet");
// 转发到Forward2Servlet
rd.forward(request, response);
```

request对象请求范围相关的方法

- 设置存储键值对（属性值）
  - void setAttribute(String name,Object value)
- 通过键返回（属性）值
  - Object getAttribute(String name)
- 在request范围内删除一个键值对（属性值）
  - removeAttribute(String name)
- 示例代码

```
request.setAttribute("username", "tom");

String username = (String)request.getAttribute("username");
```

响应对象：HttpServletResponse接口

- **javax.servlet.http. HttpServletResponse接口**
- 这个接口继承了javax.servlet.ServletResponse接口，主要的作用是将Servlet处理的结果发送给客户端

response对象-报头设置常用方法

- 禁止页面缓存
  - response.setHeader("Cache-control", "no-cache");
  - response.setDateHeader("Expires", 0);
- 网页每隔2秒自动刷新
  - response.setIntHeader("Refresh", 2);
- 网页隔一段时间自动跳转到另一个地址
  - response.setHeader("refresh", "10;url=http://www.baidu.com");

response对象-其他常用方法

- 获得用于页面输出的PrintWriter对象
  - PrintWriter getWriter()
- 设置响应的MIME类型
  - void setContentType(String contentType)
  - 例如 response.setContentType("text/html;charset=utf-8");
  - 例如 word文件格式 response.setContentType("application/msword;charset=utf-8");
- 设置编码格式
  - void setCharacterEncoding(String encode)
  - 例如 response. setCharacterEncoding("utf-8");

提交数据乱码的解决方法

- 客户端默认提交的是iso-8859-1格式的编码，是一种西欧编码，所以不能显示中文或其他国家文字，我们可以在接收数据的时候进行编码转换
- 值得注意的是post发送数据并不提前转换编码，发送过来之后可以设置编码格式，而get发送数据是发送过来之前就已经编码了，并不能在接收数据的时候设置编码
- post方式提交的数据可以设置请求编码格式
  - request.setCharacterEncoding("utf-8")
- get方式提交的数据需要做编码转换
  - // 获得表单数据
  - String data = request.getParameter("data");
  - // 转换字符串编码格式
  - data = new String(data.getBytes("iso-8859-1"), "utf-8");

显示数据乱码的解决方法

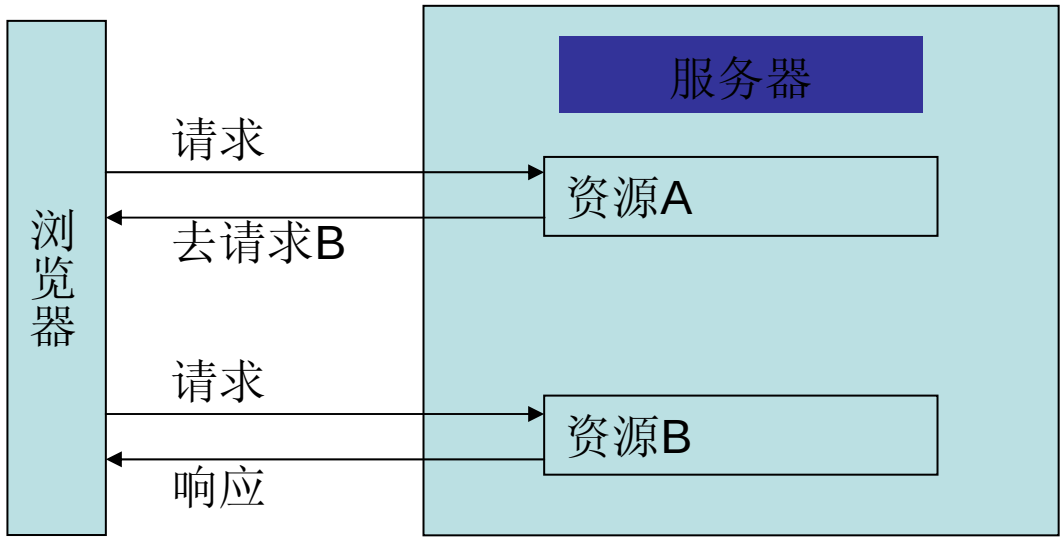
- 第一种方法
  - response.setContentType("text/html;charset=utf-8");
- 第二种方法
  - response.setCharacterEncoding("utf-8");

- 需要说明的是，在我们设置编码的时候，不一定使用UTF-8编码，应该是根据操作系统的实际情况设置，例如简体中文GBK,GB2312等

WEB资源的三种关系

- web资源中的互相访问有三种关系
  - 请求转发
    - 客户端访问资源A，资源A让资源B给客户端构造了响应返回数据
    - 使用RequestDispatcher对象的forward方法
  - 重定向
    - 客户端访问资源A，资源A给浏览器构造了响应，告诉浏览器去请求资源B，资源B给浏览器构造了响应
    - 使用response.sendRedirect方法
    - 例如response.sendRedirect("dispatcher/ServletB");
  - 包含
    - 资源A中包含了资源B
    - 使用<jsp:include>动作元素

重定向



请求转发和重定向的对比

	请求转发	重定向
目标范围	只能转发到同一个WEB应用的组件	可以是同一个WEB应用，也可以是其他站点
“/”开头含义	“/”表示WEB应用程序根目录	“/”表示整个WEB站点的根目录
地址栏显示URL	请求转发结束后，地址栏URL保持不变	重定向结束后，地址栏URL编程重定向目标URL
请求对象和响应对象	只产生一次请求对象和响应对象，资源A和资源B共享相同的请求对象和相应对象	会产生两次请求对象和响应对象