《尚学堂》Servlet&JSP&Tomcat Notes

No.1 背景知识

1. 常用的Web服务器
   1. IIS
   2. Apache
   3. Tomcat（Apache下面的针对JSP的）
2. 通常的动态网页技术
   1. CGI（Common GateWay Interface）
   2. API常用的有（NSAPI、ISAPI）
   3. ASP（Active Server Page）（已经被ASP.NET取代，支持多语言开发，不过他们没有线程，比JSP效率低）
   4. PHP（Personal Home Page）
   5. JSP/Servlet（Java Server Page）

No.3 Servlet简介

1. Servlet是服务器小应用程序
2. 用来完成B/S架构下，客户端请求的响应处理
3. 平台独立，性能优良，能以线程方式运行
4. Servlet API为Servlet提供了统一的编程接口
5. Servlet一般的在容器中运行
6. 常见的Servlet容器
   1. Tomcat

No.4 使用Servlet+HTTP基础知识获取页面的源码，这样无论网站开发人员使用怎样的JS禁用取代码都可以取下来。

形式1：GET /latest.html HTTP/1.1

Host: www.baidu.com

Content-Type: text/html

形式2：POST / servlet/serv HTTP/1.1

Host:www.baidu.com

Content-Type: application/x-www-form-urlencoded

Content-Length:32

Username=billgates&password=hehe

返回结果：HTTP/1.1 200 OK

Date:Fri/1 Dec 2003 10:23:23 GMT

Content-Type: text/html

Content-length: 280

<html>

……(具体的网页信息)

</html>

No.5 符合J2EE标准的Web Application

1. 符合J2EE标准的Web Application就是一个目录
2. 目录具体的信息
   1. WEB-INF（必须的）
      1. web.xml(必须的)：web app的配置文件
      2. lib：web app用到的库文件
      3. classes：存放编译好的servlet
   2. META-INF（可有可无）：存放web app的上下文信息，符合J2EE标准
3. Web Application可以直接放在webapps下面也可以通过配置文件指定到其他目录<Context path=”虚拟路径名” docBase=”目录位置” debug=”0” reloadable=”true”/>

No.6 Servlet的生命周期

1.生命过程

* + - * 1. 加载（ClassLoader进行类的加载）
        2. 实例化
        3. 初始化—init()
        4. 处理请求—service(),如doGet(),doPost()等
        5. 退出服务destroy()
      1. API中的过程
         1. init()

只执行一次，第一次初始化的时候

Public void init(ServletConfig config) throws ServletException

* + - * 1. service()

public void service(ServletRequest req, ServletResponse res) throws ServletExceptoin, IOException

* + - * 1. destroy()

public void destroy()

No.7 Cookie简介：Cookie就是服务器端向浏览器端写入的文本，用于记录用户的一些信息，记录在客户端！只能对应的客户端访问

1. Http协议的无线连接性要求出现一种保存c/s间状态的机制
2. Cookie：保存到客户端的一个文件，与特定客户相关
3. Cookie以“名-值”对的形式保存数据
4. 创建Cookie：new Cookie(name, value)
5. 可以使用Cookie的setXXX方法来设定一些相应的值
   1. setName(String name)/getName()
   2. setValue(String value)/getValue()
   3. setMaxAge(int age)/getAge()
   4. 利用HttpServletResponse的addCookie(Cookie)的方法将它设置到客户端
   5. 利用HttpServletRequest的getCookies()方法来读取客户端所有的Cookies，返回一个Cookie的数组

注意事项：

1. Cookie是服务器端想客户端写入的文本，其中包含了用户的一些信息
2. 写入的内容只能是文本内容
3. 客户单可以阻止服务器的写入，在浏览器的隐私安全的地方进行设置
4. 服务器端只能拿自己WebApp写入的东西
5. 一个Servlet/JSP设置的cookies能够被同一路径下的或者路径下面的Servlet/JSP访问（此处路径指的是URL）(路径！=真实路径)

No.8 Session简介：存在服务器端用于记录用户操作信息，内存会给Session分配一个内存空间用于存放Session的“名-值”对，Session是针对窗口的，窗口消失该Session就会消失

1. 在某段时间一连串客户端与服务器端的“交易”，Session的两种实现方式（面试题）：
   1. 通过Cookie实现（把Session的ID放进（临时的，不需要非临时的，因为Session是临时的）Cookie里面）
   2. 通过URL重写实现：就是把数据追加到表示会话的URL末尾
2. 可以通过程序来终止一个会话。如果客户端在一定的时间内没有操作，服务器也会自动终止会话
3. 通过HttpServlet来读写Session
4. 规则
   1. 如果浏览器支持Cookie，创建Session的时候会把SessionID保存在Cookie里面
   2. 如果浏览器不支持Cookie，必须自己编程使用URL重写的方式实现Session
      1. 需要用到的函数：resposne.encodeURL()
         1. 需要转码
         2. URL后面加入SessionID
         3. Session不像Cookie存在路径访问的问题，只要在同一个application下的servlet/jsp可以共享一个Session，前提的同一个客户端窗口

No.9 Application

1. 用于保存整个WebApplication的生命周期内都可以访问的数据
2. 在API中表现为ServletContext
3. 通过HttpServlet的getServletContext（）方法可以拿到
4. 通过ServletContext的getAttribute()/setAttribute()来获取/设置相关属性

No.10 JavaBean(一般在Servlet中使用)

1. 在Servlet中使用Bean和在通常程序中使用Bean类似
   1. 属性名称第一个字母必须小写，比如：private int productID
   2. 一般具有getters and setters(即属性的get和set方法)
2. 广义的Bean不应该具有GUI表现，狭义的具有GUI表现
3. 一般用来实现某一业务逻辑或取得特定的结果

No.11当程序中某段代码使用频繁则可以把它封装在单独的类中

No.12 JSP简介

1. JSP：Java Server Page
2. 拥有Servlet的所有特性和优点（本身就是servlet）
3. 直接在HTML中内嵌JSP代码
4. JSP程序有JSP Engine先将它转换成Servlet代码，接着将它编译成类文件载入执行
5. 只有当客户端第一次请求JSP的时候，才需要将其转换、编译，转换在Tomcat下面的work目录里面
6. 优点：
   1. 优良的性能，优于CGI、PHP、ASP
   2. 平台无关性：操作系统无关，Web服务器无关
   3. 可扩展性：tag的扩展机制，简化页面开发

No.13 JSP编程-基本语法

1. JSP传统语法
   1. Declaration声明(声明的是成员变量和成员方法)
      1. 基本语法：<%! %>（<%%>里面定义的是局部变量，所以不能声明方法）在此声明的变量是作为JSP转化为Servlet的属性（成员变量），而不是Servlet的Service方法的变量（即局部变量）
   2. Scriptlet脚本小程序
      1. 基本语法：<%Java程序代码区域%>
      2. 可以存放任何的Java程序代码
      3. 注释格式(三种)
         1. <%-- --%>
         2. <%// %>
         3. <%/\* \*/%>
   3. Expression表达式
      1. 基本语法：<%= …%>，原封不动的输出等号后面变量的值
      2. =后面必须是字符串变量或者能够转换成字符串变量的表达式
      3. 不需要以分号;结束
      4. 只有一行
      5. 例如：
         1. <%= “Hello World”%>
         2. <%= i+1 %>
         3. <%= request.getParameter(“name”) %>
   4. Comment注释(共三种注释)
      1. <!-- -->HTML注释，客户端可以通过查看源码的方式查看到该注释
      2. <%-- --%>“程序员注释”，客户端不能看到该注释，该注释只是写给程序员看的
      3. <%

// Java单行注释

/\*

Java多行注释

\*/

%>

* 1. Directives指令（编译期间就会执行，而action在运行期间才会执行）
     1. 指明与JSP Container的沟通方式
     2. 格式：<%@Directive 属性=“属性值“%>
     3. 常见的指令
        1. Page
        2. Include
        3. Taglib
     4. page基本格式
        1. <%@page language=”script language”|

Extends=”className”|

Import=”importList”|

Buffer=”none|kb size”|--none不缓冲，默认8k

Session=”true|false”|—默认true

autoFlush=“true|false |”—缓冲器是否自动清除，默认true

isThreadSafe=”true|false”|—单例模式

info=”infoText”|

errorPage=”errorPageURL” |

isErrorPge=”true|false”|

contentType=”contentTypeInfo”

%>

* + 1. include基本格式
       1. 将指定的JSP程序或者HTML文件文件包含进来
       2. 格式
       3. JSP Engine会在JSP程序的转换时期先把file属性设定的文件包含进来，然后开始执行转换及编译工作
       4. 限制
          1. 不能像fileURL中传递参数

不能abc.jsp?user=aaa

* + 1. taglib基本格式
  1. Action动作（运行期间的命令，运行期间才会执行）
     1. 常见的动作
        1. Jsp:useBean（使用javabean）（具体细节暂时不用了解）
           1. Jsp:setProperty
           2. Jsp:getProperty
        2. Jsp:include：动态的包含文件（除非这个指令会被执行到，否则不会被Tomcat等Jsp Engine执行到）
           1. 格式

<jsp:include page=”” flush=”true”/>

<jsp:include page=”” flush=”true”>

<jsp:param name=”” value=””>

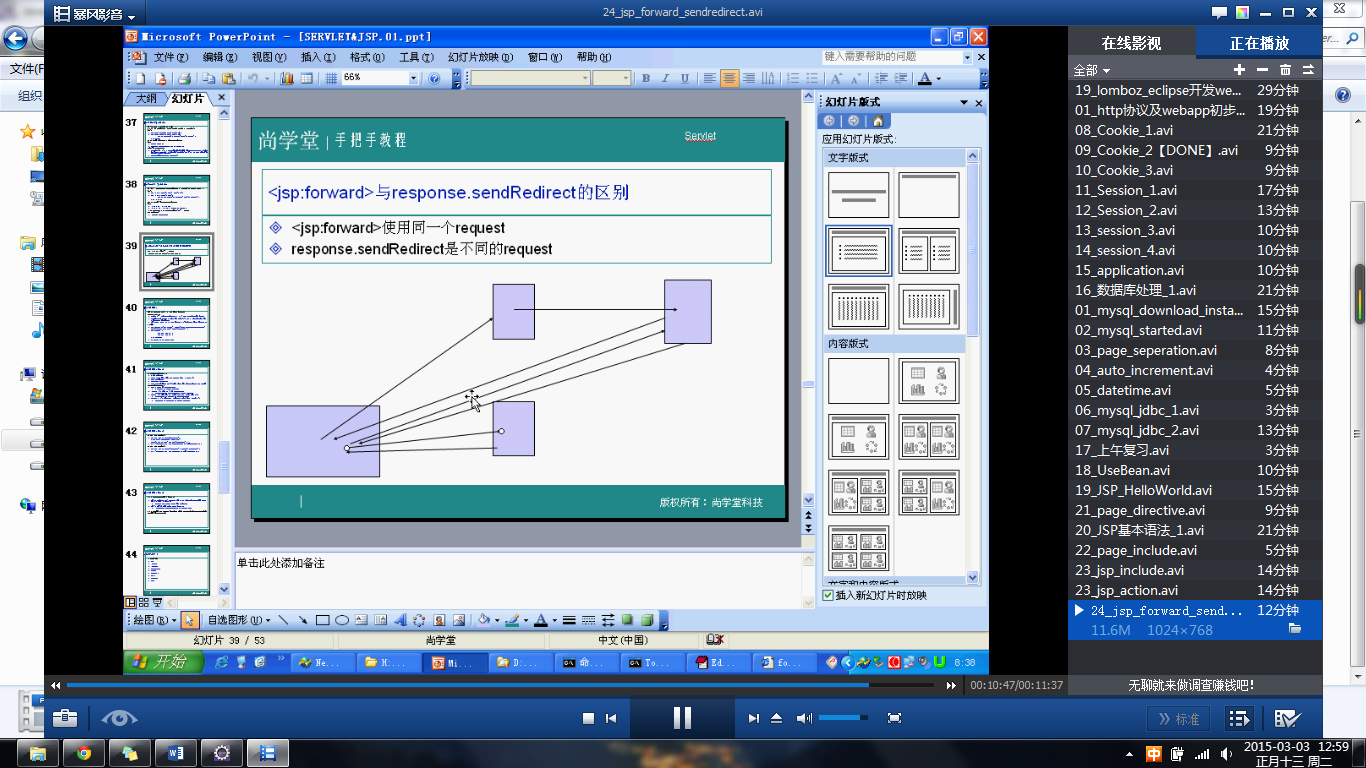
</jsp:include>

* + - * 1. Jsp:param用来设定include文件的时候参数的名称和值
        2. 和编译指令include的区别

前者（include编译指令）在JSP程序转换时期就将file属性对应的文件内容嵌入，然后编译执行；而后者（jsp:include）指令在转换时期不会被编译执行，只有在客户端请求时期被执行到才会被动态的编译载入

Include编译指令不能传递参数，而jsp:include指令却可以传递参数（使用jsp:param name=”” value=””）

* + - 1. Jsp:forward
         1. <jsp:forward page=””>
         2. Jsp:forward和response.setRedirect()的区别

前者是同一个request对象，后者是不同的request对象，因此在传递参数的时候前者可以获得，而后者却不能获得

* + - 1. Jsp:plugin(嵌入Applet程序，用到的较少)
  1. 内置对象：写出JSP常用的内置对象及其方法（面试题）
     1. out
        1. 常用方法：
           1. println()：向客户端输出各种类型
           2. newline()：输出一个换行符
           3. close()：关闭输出流
           4. flush()：输出缓冲区内的数据
           5. clearBuffer()：清除缓冲区里面的数据，同时把数据输出到客户端
           6. clear()：清除缓冲区里面的数据，但是不把数据输出到客户端
           7. getBufferSize()：返回缓冲区的大小
     2. request
        1. 常用的方法（其他的方法参考API手册）
           1. getParameter()
     3. response
        1. 常用的方法（其他的方法参考API手册）
           1. addCookie(Cookie cookie)
           2. sendRedirect(String url):重定向JSP文件
           3. setContentType()
     4. pageContext（用的很少）
     5. cookie
     6. session
     7. application
     8. config(一般的指的是web.xml，对servlet进行配置，用的很少)
     9. exception
     10. page(很少用，因为page的内容都可以直接拿到)

1. JSTL
2. JSF
3. 其他taglib（如struts）

No.14 字符乱码问题

* + - 1. 根本方法：可以先把字符串的字符接设置为ISO8859\_1，然后再转换为utf-8的形式解决乱码
      2. JSP页面中<meta http-equiv=”content-type” conent=”text/html; charset=utf-8”>是为了通知浏览器以utf-8的编码进行解码
      3. get请求和post请求提交编码方式（他们的提交方式是不同的）比较
         1. get请求通过URL的形式提交参数，而不是把参数封装在请求中，其传递的时候浏览器先把utf-8编码为ISO8859\_1传递给服务器，服务器中的doGet方法要对ISO8859\_1编码解码为utf-8才能看到中文
         2. post请求把参数存储到http请求中，不用通过URL进行传输，所以只要对请求进行编码的设置，即request.setCharacterEncoding(“utf-8”)就行了，通知request以utf-8解码，因为request默认以ISO8859\_1进行解码的

No.15 Servlet和JSP的通信

JSP调用Servlet,直接在form中的action中输入地址或者jsp:forward page=””等

Servlet调用JSP比较复杂一点，但形式不变

getServletConfig().getServletContext().getRequestDispatcher(“servlet对应的路径”).forward(request, response)