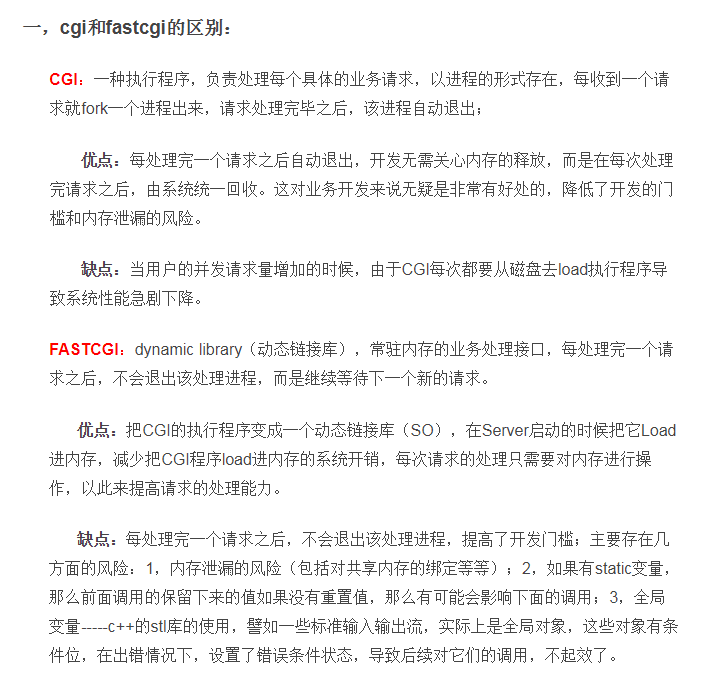
其实一部分Sevlet的笔记在jsp中，两者其实差不多，边界和模糊。

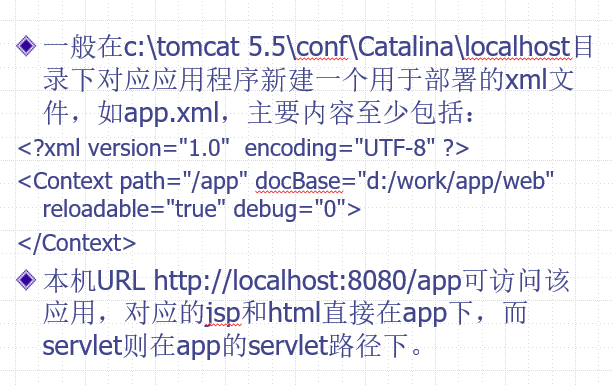
想请教一个问题，  
为什么PHP, python这种语言的后台框架不采用类似JAVA web container的架构而都使用CGI的方式呢 ？

web container 中server 和你写的应用是一体的，通讯是直接函数调用  
而cgi，你写的应用是独立进程，调用api需要通讯。c++这种编译语言 如用作后台开发也只能用cgi，cgi也好，fastcgi也罢，感觉其性能总是比不上java这种模式 。servlet常驻内存，且只有一个实例。 cgi 来一个请求开一个进程，内存开销极大；fastcgi，常驻内存，然而也是有多个进程轮流服务呀

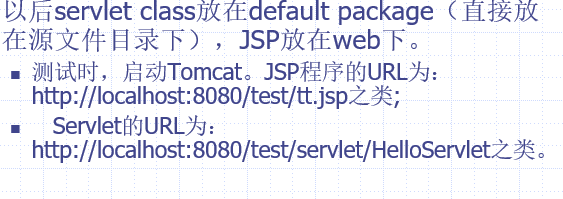
。

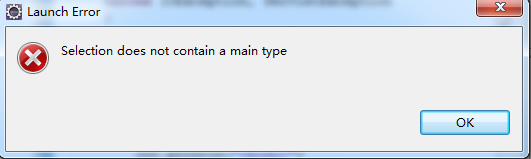
**eclipse: 项目只需要lib、src、web文件**



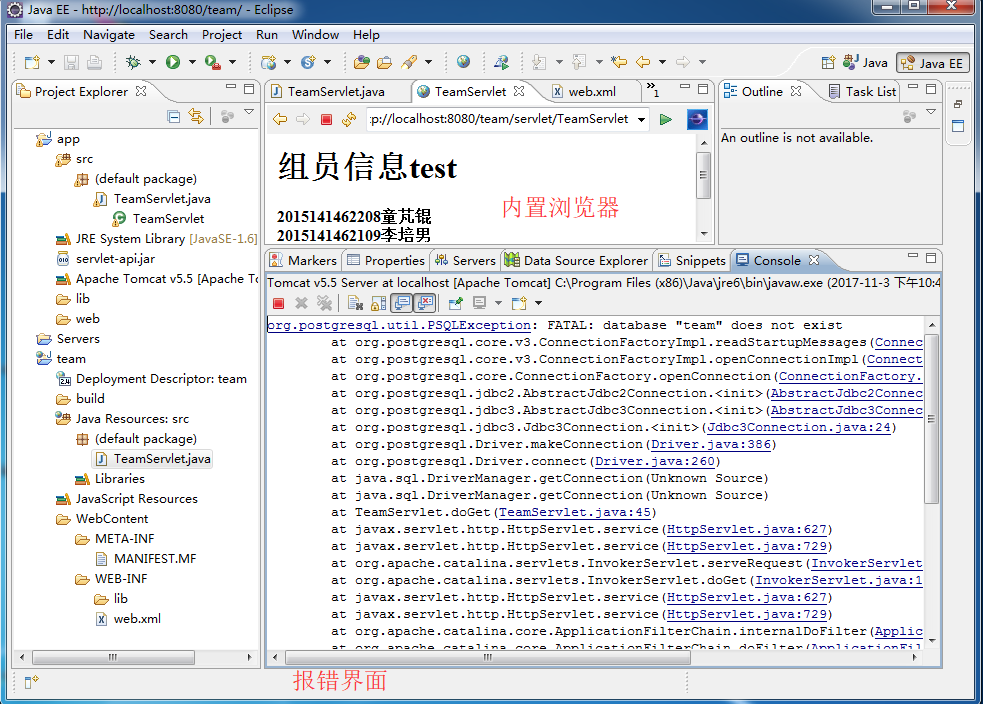


创建的xml文件名字要与路径一样。

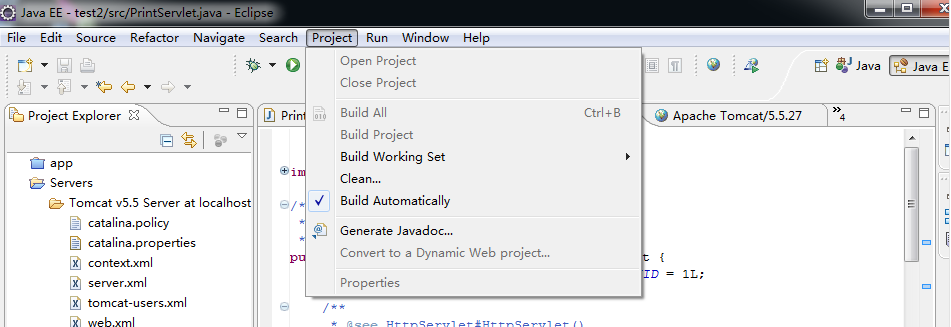


运行的时候只需要编译产生对应的class文件，而不是真正地作为一个项目运行： 

这一定不是正确的打开方式。之所以会这么蠢是因为一开始按照老师的步骤创建的是一个普通的Java Project，而不是一个web 项目，按照正常的道理应该创建一个Dynamic Web Project。创建了以后就会有Run on server，这一点一般的java不能通过配置得到：



不同于idea，改变了servlet中的内容，不知道是不是因为上面在web.xml中开启了debug的原因，不需要重新deploy。这一点似乎是因为自动编译称为class的原因，自动编译存在一小段时间的延迟，文件保存(有时候还需要被被客户端访问对应的路径)就会触发尝试自动编译。



包名自动补全：C-S-O

自动创建setter、getter函数：Shift+Alt+S

神奇的是虚拟机里面配置好环境但是不能使用自带的IE成功访问localhost:8080 应该访问http://localhost:8080/

Servlet中直接println输出自定义的html文件中文乱码问题的解决:

1.首先将文件的编码设置为utf8，这一点我通过直接设置工作目录所有文件的编码做到windows-》Preferences-》workspace 中设置。

2.在代码中加上response.setContentType("text/html;charset=UTF-8");。我想这条命令就是设置返回包中进行的编码。设置<meta>中的charset没有效果，设置response.setHeader("content-type","text/html;charset=UTF-8"); 才能让firefox浏览器自动识别为utf8编码。

3.最后方便浏览器进行自动识别。meta标签但是已经还是显示为乱码，中文全部变成问号。如果reshark看到抓到的乱码都是3F，这说明在服务器端编码转换的时候就出现了错误。

4.如果是创建了Dynamic Web Project，这时候使用的配置文件是Server项目中的配置文件

5. 听说在tomcat配置文件server.xml文件中的<Connection port=”8080”> 中加上URIEncoding=”UTF-8”可以解决有时候的乱码问题。但是我从来没有这么做过。

6.如果以上方法都救不了，我就遇到过一次莫名其妙的乱码问题，重新建立一个工程，将代码移过去就好了。

7.JSP 中文乱码：

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

pageEncoding=*"UTF-8"*%>

<http://blog.csdn.net/bestlove12345/article/details/51819458> 编码介绍

莫名其妙访问所有页面，任何一个项目的页面，都是404 删除eclipse中的服务器配置，重新加入

在test项目中写了一个PrintServler的Servlet，<http://192.168.1.235:8080/test/PrintServlet> 与 <http://192.168.1.235:8080/test/servlet/PrintServlet> 都是可以访问。

**idea:**

idea运行tomcat的时候会自动打开index.html index.jsp，因为这是tomcat里面的默认配置

idea中导入eclipse的项目需要特殊处理，不是直接open，需要File-》New-》Project from existing Sources…，这样才会有选择是哪个IDE完成的项目。

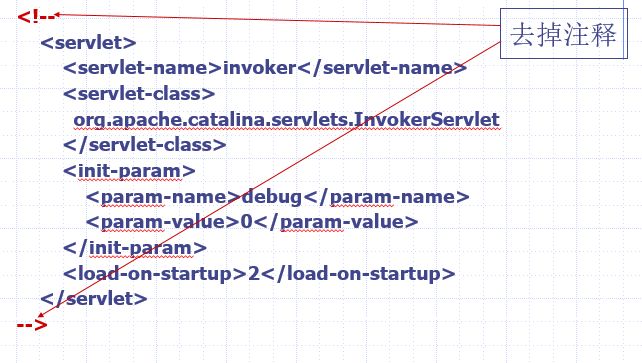
在idea中的项目不能直接update resources了，至少需要redeploy才能让servlet被销毁和重新初始化。但是每一次deploy都会清除已有的session。

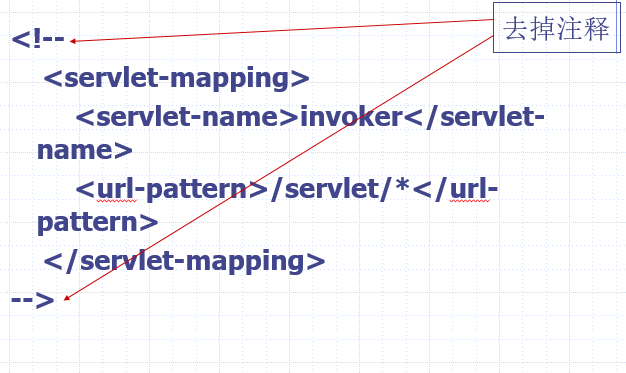
Java Servlet 通常情况下与使用 CGI（Common Gateway Interface，公共网关接口）实现的程序可以达到异曲同工的效果。但是Servlet性能更好（比如说有数据库连接池），并且是java的。

System.*out*.println("Debugging message");

就能够输出到idea下方的黑框中。

开启老版Tomcat中Servlet调试器： 在conf/web.xml中做到以下两部分





<http://blog.csdn.net/guyuealian/article/details/50762996> 关联eclipse（注意要是Java Web版的eclipse）与tomcat

旧版eclipse 会读取系统环境变量从而知道关于java的设置，赵泓尧定义了一个\_JAVA\_OPTIONS变量，这个错误的环境变量导致eclipse不能编译代码文件。以为重启eclipse就能重新读取文件，以为删掉Run Configure中的SDK再重新加入就好，但是最终重启电脑才是真正的解决办法。这是第一次因为环境变量导致重启的。

整个请求过程：

1. 客户端请求该Servlet
2. Servlet被加载到内存，创建实例，调用init()进行初始化
3. 服务器将会创建出一个请求对象用来处理客户端的请求，创建一个响应对象用来响应客户端请求。然后激活Serlvet的service(()，并传递请求与响应对象作为参数。如果有更多的服务器请求就只会重复这一步。
4. 当 Server 不再需要 Servlet 时（一般当 Server 关闭时），Server 调用 Servlet 的 destroy() 方法。我们在doGet中的最后调用destroy方法，于是每一次请求都会有destory 方法被调用，servlet 被销毁，但是并没有立即被回收，再次请求时，并没有重新初始化，控制台输出如下：

servlet初始化……

servlet销毁！

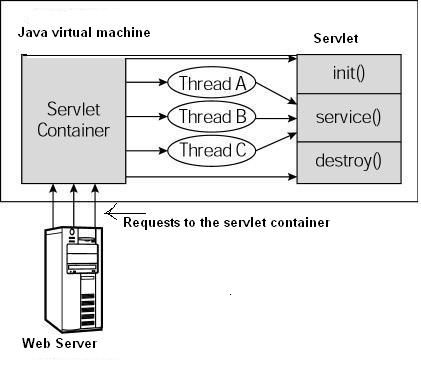
servlet销毁！

servlet销毁！

servlet销毁！

servlet销毁！

servlet销毁！



默认情况下，Servlet 应用程序位于路径 <Tomcat-installation-directory>/webapps/ROOT 下，且类文件放在 <Tomcat-installation-directory>/webapps/ROOT/WEB-INF/classes 中。如果您有一个完全合格的类名称 **com.myorg.MyServlet**，那么这个 Servlet 类必须位于 WEB-INF/classes/com/myorg/MyServlet.class 中。在classes文件夹汇总放入文件以后还需要在上层目录也就是WEB-INF中的web.xml增加对应的描述条目：  
<web-app>

<servlet>

<servlet-name>HelloWorld</servlet-name>

<servlet-class>HelloWorld</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>HelloWorld</servlet-name>

<url-pattern>/HelloWorld</url-pattern>

</servlet-mapping>

</web-app>

在idea中进行编程：

<http://blog.csdn.net/u012532559/article/details/51013400>，有个地方可以进行更改：“Application context”创建的地方，如果不填上/WebClient而是默认的/，那么在URL中就不需要加上/WebClient

完全是种类的写法



在网址上看起来就是.do结尾，需要一个xml文件对类名与url地址（xxx.do）进行映射。这个文件中还配置了优先级（<servlet> 标签下配置 <load-on-startup> 标签，配置的值为整型，值越小 Servlet 的启动优先级越高）。

输出： http://www.cnblogs.com/1314wamm/p/5951515.html

Process proc = Runtime.*getRuntime*().exec("cmd /c C:\\Users\\Battery\\Desktop\\example.bat");  
InputStream in=proc.getInputStream();  
BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(in));  
String tmpString;  
ServletOutputStream out = response.getOutputStream();  
while((tmpString=bufferedReader.readLine())!=null) {  
 out.print(tmpString);  
}

getOutputStream() has already been called for this response 这个错误的出现表示response.getWriter()和response.getOutputStream()等接口在调用时发生了资源占用。网上面有个例子是因为没有调用close()在关闭流输出的接口。

//ServletOutputStream out = response.getOutputStream(); 会输出不全  
PrintWriter out = response.getWriter();  
out.print(json);out.flush();  
out.close();

// 导入必需的 java 库

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

// 扩展 HttpServlet 类

public class HelloWorld extends HttpServlet {

private String message;

public void init() throws ServletException

{

// 执行必需的初始化

message = "Hello World";

}

public void doGet(HttpServletRequest request,

HttpServletResponse response)

throws ServletException, IOException

{

response.setCharacterEncoding("UTF-8");

response.setContentType("text/html;charset=utf-8");//解决中文乱码

// 设置响应内容类型

response.setContentType("text/html");

// 实际的逻辑是在这里

PrintWriter out = response.getWriter();

out.println("<h1>" + message + "</h1>");

}

public void destroy()

{

// 什么也不做

}

}

由源代码产生到WEB-INF/classes中的class文件其实很简单，就是没有定义的类声明那样。

request和response类和jsp中的基本操作是一样的

Servlet3.0新增了request.getParts()/getPart(String filename) api，用于获取使用multipart/form-data格式传递的http请求的请求体，通常用于获取上传文件。

1. // 上传单个文件
2. // Servlet3.0将multipart/form-data的POST请求封装成Part，通过Part对上传的文件进行操作。
3. // Part part = parts[0];//从上传的文件集合中获取Part对象
4. Part file = request.getPart("file");// 通过表单file控件(<input type="file" name="file">)的名字直接获取Part对象
6. // Servlet3没有提供直接获取文件名的方法,需要从请求头中解析出来
7. // 获取请求头，请求头的格式：form-data; name="file"; filename="snmp4j--api.zip"
8. String header = file.getHeader("content-disposition");
10. // 获取文件名
11. String fileName = getFileName(header);
13. // 把文件写到指定路径
14. file.write(savePath + File.separator + fileName);
15. String fileAddr = savePath + "/" + fileName;

在web.xml中指定错误页面（对于某一个响应码或者异常）：

<!-- error-code 相关的错误页面 -->

<error-page>

<error-code>404</error-code>

<location>/ErrorHandler</location>

</error-page>

<error-page>

<error-code>403</error-code>

<location>/ErrorHandler</location>

</error-page>

<!-- exception-type 相关的错误页面 -->

<error-page>

<exception-type>

javax.servlet.ServletException

</exception-type >

<location>/ErrorHandler</location>

</error-page>

<error-page>

<exception-type>java.io.IOException</exception-type >

<location>/ErrorHandler</location>

</error-page>



String servletName = (String)

request.getAttribute("javax.servlet.error.servlet\_name");

if (servletName == null){

servletName = "Unknown";

}

然后就可以编写一个错误处理页面并在web.xml中进行注册

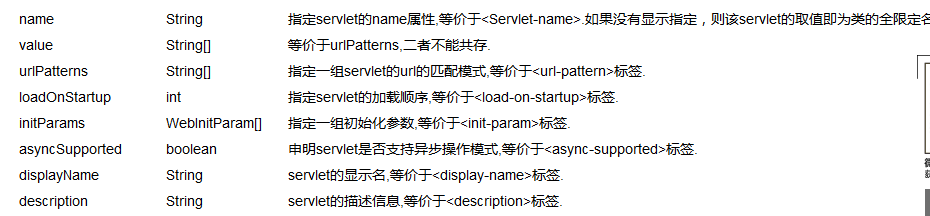
没想到用到的jar包要添加两次，一次作为辅助项目编译设置在Project Structure中，还有就是执行Servlet而将之放到WEB-INF下设定的library root directory也就是lib文件夹中。有待研究？？？

servlet3.0中的新特性： 和Android好像好像

@WebServlet(name = "register",urlPatterns = "/register.do")

就相当于在web.xml中进行生命与url映射等

<servlet>  
 <servlet-name>register</servlet-name>  
 <servlet-class>register</servlet-class>  
</servlet>  
<servlet-mapping>  
 <servlet-name>register</servlet-name>  
 <url-pattern>/register.do</url-pattern>  
</servlet-mapping>



//要告知账号或密码错误  
response.sendRedirect("login.jsp?error=yes");  
/\*  
 \* <script>  
 var errori ='<%=request.getParameter("error")%>';  
 if(errori=='yes'){  
 alert("登录失败!");  
 }  
 </script>  
\* \*/

server.xml



sparkjava 是java8的web后台编写框架，目标是用最少的代码来进行编程的框架。在我看来就是换了一种编程方式，这种框架把不同的路径集中在一起写在类的main函数中，所以不太喜欢。

request处理以后传递出去：

public void doPost( HttpServletRequest request, HttpServletResponse response )

{ String output = null;

try

{

String name = request.getParameter( "myname" );

output = new String( "Hello " + name + "." );

}

catch( Exception e)

{

System.err.println( e.toString() );

}

request.setAttribute("output",output);

try

{

RequestDispatcher rd;

rd = getServletContext().getRequestDispatcher( "/showname.jsp" );

rd.forward(request, response);

}

catch( Exception e )

{}

}

}

getRequestDispatcher 与

@WebServlet(name = "Staff\_Login", urlPatterns = "/Staff\_Login")

urlPatterns中的路径都是相对的。

response.sendRedirect("/hotel/hotel\_index.jsp");

以及js文件中的XMLHttp发送URL路径都是绝对的，需要自己将放到服务器之后的路径映射考虑进去。

**RequestDispatcher：**

通过使用这个类提供的forward方法或者include方法将其他的servlet、jsp连接起来

**forward（ServletRequest，ServletResponse）方法：**

将请求处理以后转发给服务器上另外一个Servlet，JSP页面，或者HTML文件。这种跳转不同于response.sendRedirect()，它是一种服务器端的跳转，所以用户不会受到302导致“浏览器”的URL发生变化。注意，只有在尚未向客户端输出响应时才可以调用forward()方法，否则会抛出IllegalStateException异常。

**include（ServletRequest，ServletResponse）方法 ：**

用于在响应中包含其他资源(Servlet，JSP页面或HTML文件)的内容。即请求转发后，原先的Servlet还可以继续输出响应信息，转发到的Servlet对请求做出的响应将并入原先Servlet的响应对象中。

doGet方法后边有throws ServletException, IOException这样的语句，表示有此类异常抛出，交给服务器处理，没有处理的会显示默认的错误信息。但是并不是所有的异常都可以给服务器处理，SQLException就不行。

tomcat启动报错No UserDatabase component found under key UserDatabase

非正常关机，造成了tomcat下conf文件中的tomcat-user.xml文件异常，copy一个新的tomcat-user.xml覆盖就可以了.

我们这边中文使用的编码是