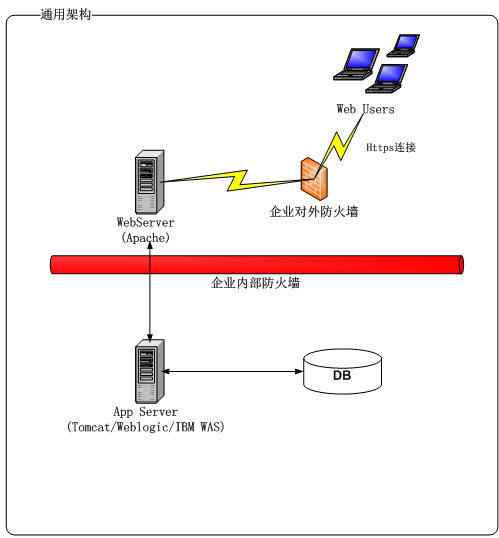
**一、先从J2EE工程的通用架构说起**



这是一个通用的Web即B/S工程的架构，它由：

ü   Web Server

ü   App Server

ü   DB Server

三大部分组成，其中：

²  Web Server

置于企业防火墙外，这个防火墙，大家可以认为是一个CISCO路由器，然后在CISCO路由器上开放了两个端口为：80和443。

80端口：用于正常的http访问

443端口：用于https访问，即如果你在ie里打入<https://xxx.xxx.xx>这样的地址，默认

走的是443这个端口。

         WebServer专门：

用于解析HTML、JS（JavaScript）、CSS、JPG/GIF等图片格式文件、TXT、

VBSCRIPT、PHP等一切一切“静态”网页内容。

²  App Server

置于企业防火墙内，它和Web Server之间的连接必须且一定为内部IP连接。

外部IP：即Internet IP地址，我们的web服务器一般会有一个内部IP一个外部IP，因此在这里，我们的App Server没有任何外部IP，只有内部IP，所以我在这边说App Server与Web Server只能以内部IP形式连接。

打比方说我们用的是tomcat，它的端口为8080，那么这个ip地址上的8080端口只能由任何内部ip才能访问，外部的internet是访问不了的，这样做就是为了安全。

App Server用于解析我们的任何需要Java编译器才能解析的“动态”网页，其实App Server本身也能解析任何静态网页的。

那么我们这样来想一下：

我们让负责专门解析静态网页的Web Server来解析html等内容，而让App Server专门用于解析任何需要Java编译器才能解析的东西，让它们“两人”各司其职。这样作的好处：

1）    为App Server“减压”，同时也提高了performance

2）    不用再把8080这个端口暴露在internet上了，也很安全，必经我们的app server上可是有我们的代码的，就算是编译过的代码也容易被“反编译”，这是很不安全的。

3）    为将来的进一步的“集群扩展”打好了基础

²  DB Server

打比方说我们用的是Oracle，它需要通过1521与App Server进行连接是不是？那么这个1521我们称为数据库连接端口，如果把它暴露在Internet上，是不是在危险了点？就算我们的密码很复杂，但 对于高明的黑客来说，要攻破你的口令也只是时间上的问题而己。

因此我们把我们的DB Server也和App Server一样，置于内网的防火墙。任何的DB连接与管理只能通过内网即在公司企业内部来访问，就是这个道理。

**二、动手来架构**

**2.1 Oracle数据加的安装与配置**

DB(Oracle)我已经为大家准备好了，连接信息为：

|  |  |
| --- | --- |
| **IP：** | **10.225.10x.xx** |
| **Port:** | **1521** |
| **Username/Password:** | **xxx/xxx** |
| **Sid:** | **Jcoedb1** |
| **url:** | **jdbc:oracle:thin:@10.225.10x.xx:1521:xxx** |

所以，根据上述的架构，我们可以把如下这样的一份清单丢给NSS或者是相关的网络管理部门，让他们给我们开通相应的端口：

|  |  |
| --- | --- |
| **Web Server** | **对外IP: xxx.xxx.xxx.xxx 对内IP：10.225.xxx.xxx**  **向internet开通80与443端口** |
| **App Server** | **对内IP: 10.225.xxx.xxx**  **只对10.225.段的ip开放8080,8009等端口，** |
| **Db Server** | **对内IP: 10.225.xxx.xxx**  **只对10.225.段的ip开放1521端口** |

**2.2 App Server的安装**

直接解压tomcat至你的本地如：d:\tomcat，我这边用的目录名叫tomcat2，大家随意，最好名字能够越简单越好d:\tomcat或者c:\tomcat就行，不要放得太“深”。

**2.3 Web Server的安装**

我们在这边将安装Apache For Win 2.2.x，它将占用你机器的80和443端口。因此如果你机器上有任何程序占用你的80和443端口，必须将它关闭掉，比如说：

*我们装有微软的IIS，这本身也是一个WebServer，那么请你将它关闭：*

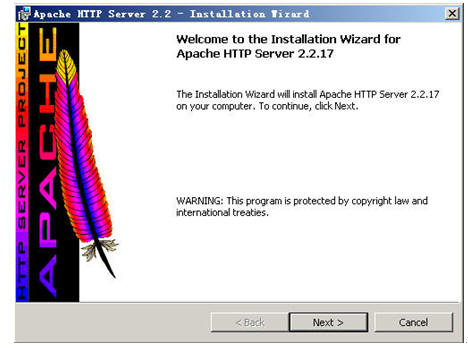
*ControlPanel->Administrative Tools->Service，找到IISAdmin和，将它全部关闭并将启动方式设为:manual以便于不用每次重启后再要去手动关闭一下。*

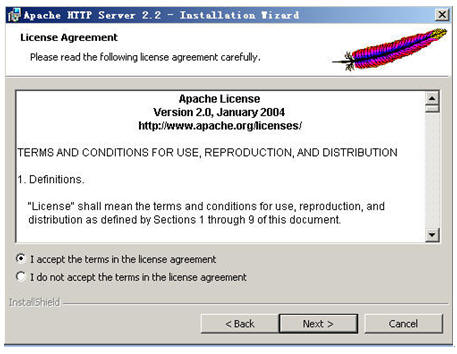
*然后用netstat –ano找到任何还在占用80端口的程序，将它关闭掉。*

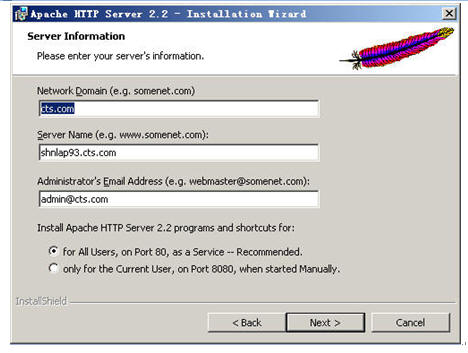
**2.4 开始安装Apache Http Server**

我们将安装这个版本的apache http server作为我们今后一直使用的Web Server

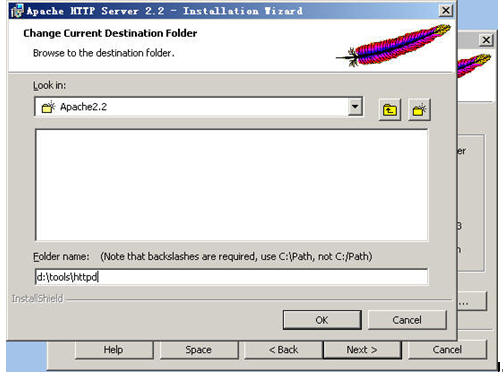


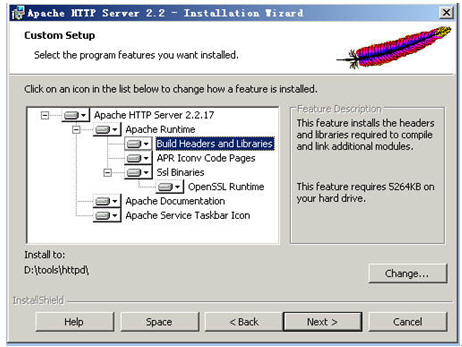




****

**这边的server name你们要填入自己的server的真实名，不能用我这个，这个servername如：shnlap93.cts.com只能够我用，这个名称是全局唯一的，和你的IP一样。**

****

****

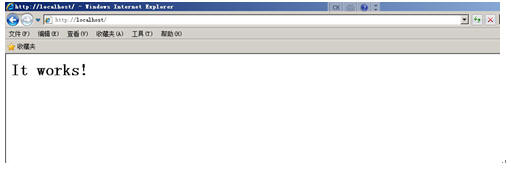
**选全部安装**



装完后你会多出一个这样的图标来，点击该图标，里面有用于控制apache http server的启动、停止与重启等操作选项。同时在你们的“服务”面板中，也能发现这样的一个服务项，它启动时默认是随着系统的启动而启动的，我们把它改 成“手动”吧，因为将来我们还要安装IBM Http Server来作练习。



装完后，在Apache2.2启动的前提下，打开一个ie输入<http://localhost>，你将会得到这样的一个页面，就说明你的Apache的安装是成功的。



**2.5 Apache的配置**

学Java的人，必须会这个Apache的配置，要不然你怎么模拟环境、搭建环境和架构环境？光会Coding是远远不够的，你将永远只配作个码农。。。嘿嘿嘿！有很多人发觉到了后面JAVA学不上去了，关键因素在于：配置。

你会配环境了，那么你就能模拟任何客户方、开发方的环境。

你会配环境了，你的代码将来上线时才能成功运行。

你会配环境了，所以整个工程的技术核心就是你。

跟着我的教程，你们将会安装和运行达近百个各种软件与配置，搞得你一股臭味一股臭味！！

你准备好了没有？

当然，不用怕，因为我的配置都是实际运行的环境，所以网上的一些东西你可以不用去看，因为很多人都是在网上进行拷贝、复制，有时也不经过验证，会让你走很多的弯路到头来还是落得个BUG一天世界，就看我的教程吧。

Apache的配置主要集中在**httpd.conf**文件，它位于你的安装目录，比如：

**D:\tools\httpd\conf\**

我们用ultraedit或者相关文本编辑工具打开它，来看它的内容：

先来查找到如下这一行：

#ServerName

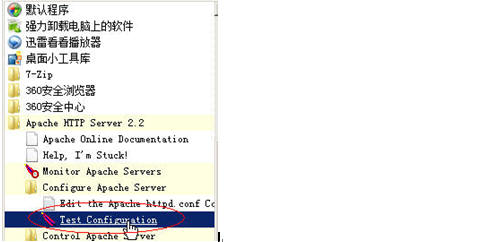
我们可以得到如下这一行内容：

|  |
| --- |
| #ServerName shnlap93.cts.com:80 |

这就是我们的主机名了，我们可以将前面的“#”去掉，并将其改为：

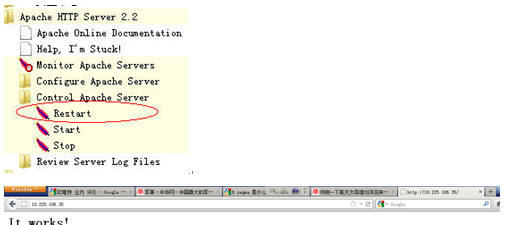
|  |
| --- |
| ServerName 10.225.106.35:80 |

改完后存盘，在重启你的Apache2.2前我们先测试一下我们的Apache的配置文件是否改得对：



如果在你点了Test Configuration后，黑屏一闪而过，说明你的改动无误，否则这个黑屏会一直停留在当前状态，并且告诉你，你的配置改动有错，错在哪里。

重新启动你的Apache



找到如下这行：

DocumentRoot

你会发下有这样的一行内容：

|  |
| --- |
| DocumentRoot "D:/tools/httpd/htdocs" |

这个叫作DocumentRoot即webroot，即：发布目录，发布在这个目录下的任何工程都会在Apache服务开启时被装载成标准的web工程，我们现在动手来把这个WebRoot定位到我们自己的发布目录中去吧。

|  |
| --- |
| DocumentRoot "d:/www" |

我们把它改到了d盘的www目录中去了，然后我们在该目录中放入一个index.html文件，内容为：

|  |
| --- |
| <html><body><h1>Hey man, apache works!</h1></body></html> |

重启我们的Apache服务，来测试一下：



嘿嘿，我们得到了什么？禁止访问，为什么？

找到下面这一段：

|  |
| --- |
| <Directory />      Options FollowSymLinks      AllowOverride None      Order deny,allow  **deny from all**  </Directory> |

看到了没？

现在，把这个”deny from all”改成”allow fromall’吧。

|  |
| --- |
| <Directory />      Options FollowSymLinks      AllowOverride None      Order deny,allow  **allow from all**  </Directory> |

修改完后重启你的Apache服务



Ok,我们的Apache的发布目录已经成功更改到了d:\www目录下了，我们再来做一个实验：

我们在IE浏览器中输入: http://localhost/css/，我们看到了什么？



这还了得，用户如果是个初级黑客都可以知道我们的服务器上有哪些文件，哪些目录甚至可以直接看到我们的文件内容，怎么办？

找到下面这行

|  |
| --- |
| Options FollowSymLinks indexes |

把它注掉改成下面这样

|  |
| --- |
| #Options FollowSymLinks indexes  Options None |

不要急，再往下找，还有

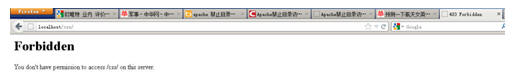
|  |
| --- |
| Options Indexes FollowSymLinks |

又来一个，再改掉

|  |
| --- |
| #Options Indexes FollowSymLinks  Options None |

改完这两条后重启你的Apache服务

再次打开一个新的IE，输入：<http://localhost/css/>，我们看到了如下的界面：



好了，Apache的基本配置完成了即：

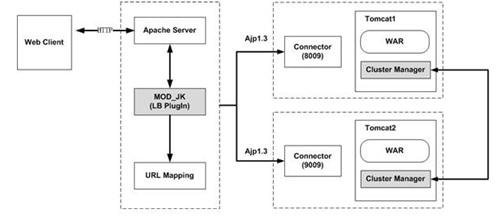
1）  基本的安全配置，不允许目录访问

2）  把WebRoot改到另一个物理目录上而不使用Apache自带的WebRoot目录

**2.6 整合Apache与Tomcat**

Apache(Web Server)负责处理HTML静态内容；

Tomcat(App Server)负责处理动态内容；



其实就是上述这样的一个架构，下面是原理

1)      Apache装有一个模块，这个模块叫mod\_jk

2)      Apache通过80端口负责解析任何静态web内容

3)      任何不能解析的内容，用表达式告诉mod\_jk，让mod\_jk**派发**给相关的app server去解释。

通过上述的文字描述我们可以得知：

1）  我们需要在Apache中先装一个mod\_jk

2）  我们需要在httpd.conf中写点表达式

下面来实现。

1)      把mod\_jk-1.2.31-httpd-2.2.3.so手工copy进我们的Apache安装目录的modules目录下， 这个文件的全名叫: mod\_jk-1.2.31-httpd-2.2.3.so，大家可以从ftp上的“/JavaArchitect/mod\_jk/”目录中获取，因为这 个文件是我用C++在本地重新编译过的，网上下载的是src即源码，省去大家再去编译的时间了，而且一些其它网上下载的mod\_jk.so是无法使用的。

2)      用ultraedit打开httpd.conf文件，跑到文件最后面加入以下几行：

|  |
| --- |
| LoadModule jk\_module modules/mod\_jk-1.2.31-httpd-2.2.3.so  JKWorkersFile conf/workers.properties  JkLogFile logs/mod\_jk.log  <VirtualHost \*>  ServerAdmin localhost  DocumentRoot d:/www/  ServerName localhost  DirectoryIndex index.html index.htm index.jsp index.action  ErrorLog logs/shsc-error\_log.txt  CustomLog logs/shsc-access\_log.txt common    JkMount /\*WEB-INF ajp13  JkMount /\*j\_spring\_security\_check ajp13  JkMount /\*.action ajp13  JkMount /servlet/\* ajp13  JkMount /\*.jsp ajp13  JkMount /\*.do ajp13  JkMount /\*.action ajp13    JkMount /\*fckeditor/editor/filemanager/connectors/\*.\* ajp13  JkMount /fckeditor/editor/filemanager/connectors/\* ajp13  </VirtualHost> |

关键的是这两句：

LoadModule jk\_module modules/mod\_jk-1.2.31-httpd-2.2.3.so

JKWorkersFile conf/workers.properties

代表：

ü   Apache载入一个额外的插件，用于连接tomcat。

ü   连接时的配置参数描述位于Apache安装目录的/conf目录下的一个叫workers.properties文件中，mod\_jk一般使用ajp13协议连接，使用的是tomcat的8009端口。

3)      Worker.properties文件内容如下：

|  |
| --- |
| workers.tomcat\_home=d:/tomcat2  workers.java\_home=C:/jdk1.6.32  ps=/  worker.list=ajp13  worker.ajp13.port=8009  worker.ajp13.host=localhost  worker.ajp13.type=ajp13 |

4)      告诉我们的Apache，哪些是要交给tomcat来解析，**除此之外都由Apache本身来解析**：

|  |
| --- |
| <VirtualHost \*>  ServerAdmin localhost  DocumentRoot d:/www/  ServerName localhost  DirectoryIndex index.html index.htm index.jsp index.action  ErrorLog logs/shsc-error\_log.txt  CustomLog logs/shsc-access\_log.txt common    JkMount /\*WEB-INF ajp13  JkMount /\*j\_spring\_security\_check ajp13  JkMount /\*.action ajp13  JkMount /servlet/\* ajp13  JkMount /\*.jsp ajp13  JkMount /\*.do ajp13  JkMount /\*.action ajp13    JkMount /\*fckeditor/editor/filemanager/connectors/\*.\* ajp13  JkMount /fckeditor/editor/filemanager/connectors/\* ajp13  </VirtualHost> |

大家看到没，所有的/servlet/\*都由tomcat负责解析，所有的jsp, .do, .action都由tomcat解析。

此处还有一个特殊的/fckeditor，这个是我们使用的一个博客编辑器，这个因为是servlet的，因此也需要交给tomcat铁析。

5)      将/cbbs工程布署到tomcat的webapps目录下

6)      将/cbbs同样手工copy一份到d:/www目录下

7)      删除d:/www/cbbs/WEB-INF这个目录，嘿嘿，因为d:/www下的东西是由Apache解析的，所有的WEB-INF下的都是Java，我们只需要布署在tomcat下即可，是不是？

8)      重启tomcat，重启Apache，在ie中直接输入： <http://localhost/cbbs>，使用sally/abcdefg登录，操作一下，一切成功

Oh…yeah, tomcat+apache一步搞定。

**三、用于实验的cbbs工程配置**

***最后附上cbbs布署需要用到的配置，相关的工程可通过ftp的” /Java Architect/Project/”下的cbbs.zip来获取。***

ü   在tomcat中打开server.xml加入：

|  |
| --- |
| <Resource  driverClassName="oracle.jdbc.OracleDriver"  factory="org.apache.commons.dbcp.BasicDataSourceFactory"  maxActive="25" maxIdle="100" maxWait="5000" name="jdbc/eltds"  password="xxx"  type="javax.sql.DataSource"  url="jdbc:oracle:thin:@10.225.101.51:1521:jcoedb1"           username="xxx"/> |

和

|  |
| --- |
| <Context crossContext="true" docBase="D:/upload" path="/uploadpic" reloadable="true"/>  <Context docBase="cbbs" path="/cbbs" reloadable="true"/> |

ü   手工在d盘根目录建立一个upload目录，在此目录内再建立一个image目录。

ü   在tomcat中打开context.xml加入

|  |
| --- |
| <ResourceLink   name="jdbc/cbbsds"   type="javax.sql.DataSource"   global="jdbc/cbbsds"/> |