通向架構師的道路（第一天）之Apache整合Tomcat - lifetragedy的專欄 - 博客頻道

分類：

一、先從J2EE工程的通用架構說起

這是一個通用的Web即B/S工程的架構，它由：

ü Web Server

ü App Server

ü DB Server

三大部分組成，其中：

² Web Server

置於企業防火牆外，這個防火牆，大家可以認為是一個CISCO路由器，然後在CISCO路由器上開放了兩個端口為：80和443。

80端口：用於正常的http訪問

443端口：用於https訪問，即如果你在ie裡打入https://xxx.xxx.xx這樣的地址，默認

走的是443這個端口。

WebServer專門：

用於解析HTML、JS（JavaScript）、CSS、JPG/GIF等圖片格式文件、TXT、

VBSCRIPT、PHP等一切一切「靜態」網頁內容。

² App Server

置於企業防火牆內，它和Web Server之間的連接必須且一定為內部IP連接。

外部IP：即Internet IP地址，我們的web服務器一般會有一個內部IP一個外部IP，因此在這裡，我們的App Server沒有任何外部IP，只有內部IP，所以我在這邊說App Server與Web Server只能以內部IP形式連接。

打比方說我們用的是tomcat，它的端口為8080，那麼這個ip地址上的8080端口只能由任何內部ip才能訪問，外部的internet是訪問不了的，這樣做就是為了安全。

App Server用於解析我們的任何需要Java編譯器才能解析的「動態」網頁，其實App Server本身也能解析任何靜態網頁的。

那麼我們這樣來想一下：

我們讓負責專門解析靜態網頁的Web Server來解析html等內容，而讓App Server專門用於解析任何需要Java編譯器才能解析的東西，讓它們「兩人」各司其職。這樣作的好處：

1） 為App Server「減壓」，同時也提高了performance

2） 不用再把8080這個端口暴露在internet上了，也很安全，必經我們的app server上可是有我們的代碼的，就算是編譯過的代碼也容易被「反編譯」，這是很不安全的。

3） 為將來的進一步的「集群擴展」打好了基礎

² DB Server

打比方說我們用的是Oracle，它需要通過1521與App Server進行連接是不是？那麼這個1521我們稱為數據庫連接端口，如果把它暴露在Internet上，是不是在危險了點？就算我們的密碼很複雜，但對於高明的黑客來說，要攻破你的口令也只是時間上的問題而已。

因此我們把我們的DB Server也和App Server一樣，置於內網的防火牆。任何的DB連接與管理只能通過內網即在公司企業內部來訪問，就是這個道理。

二、動手來架構

2.1 Oracle數據加的安裝與配置

DB(Oracle)我已經為大家準備好了，連接信息為：

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，根據上述的架構，我們可以把如下這樣的一份清單丟給NSS或者是相關的網絡管理部門，讓他們給我們開通相應的端口：

Web Server	**對外IP: xxx.xxx.xxx.xxx** **對內IP：10.225.xxx.xxx** **向internet開通80與443端口**
App Server	**對內IP: 10.225.xxx.xxx** **只對10.225.段的ip開放8080,8009等端口，**
Db Server	**對內IP: 10.225.xxx.xxx** **只對10.225.段的ip開放1521端口**

2.2 App Server的安裝

直接解壓tomcat至你的本地如：d:\tomcat，我這邊用的目錄名叫tomcat2，大家隨意，最好名字能夠越簡單越好d:\tomcat或者c:\tomcat就行，不要放得太「深」。

2.3 Web Server的安裝

我們在這邊將安裝Apache For Win 2.2.x，它將佔用你機器的80和443端口。因此如果你機器上有任何程序佔用你的80和443端口，必須將它關閉掉，比如說：

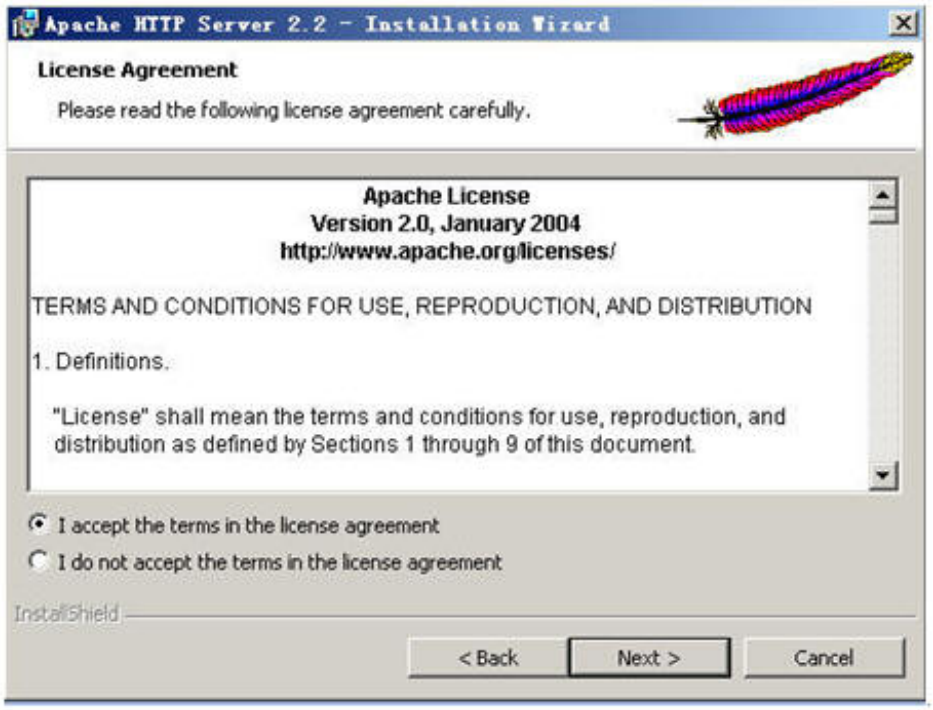
我們裝有微軟的IIS，這本身也是一個WebServer，那麼請你將它關閉：

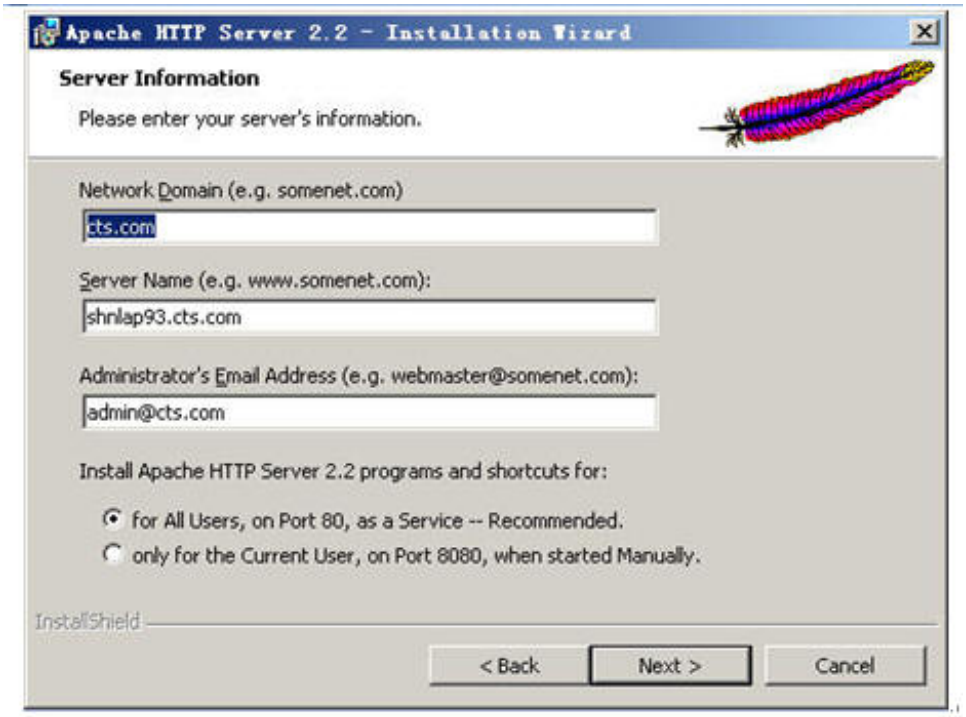
ControlPanel->Administrative Tools->Service，找到IISAdmin和，將它全部關閉並將啟動方式設為:manual以便於不用每次重啟後再要去手動關閉一下。

然後用netstat –ano找到任何還在佔用80端口的程序，將它關閉掉。

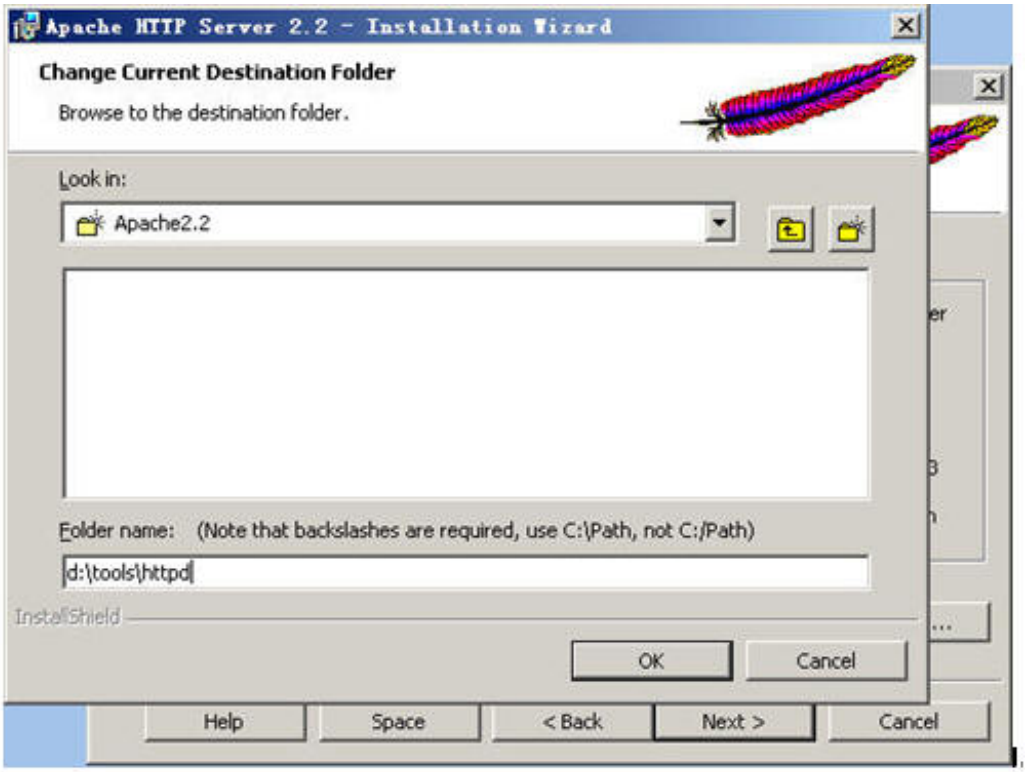
2.4 開始安裝Apache Http Server

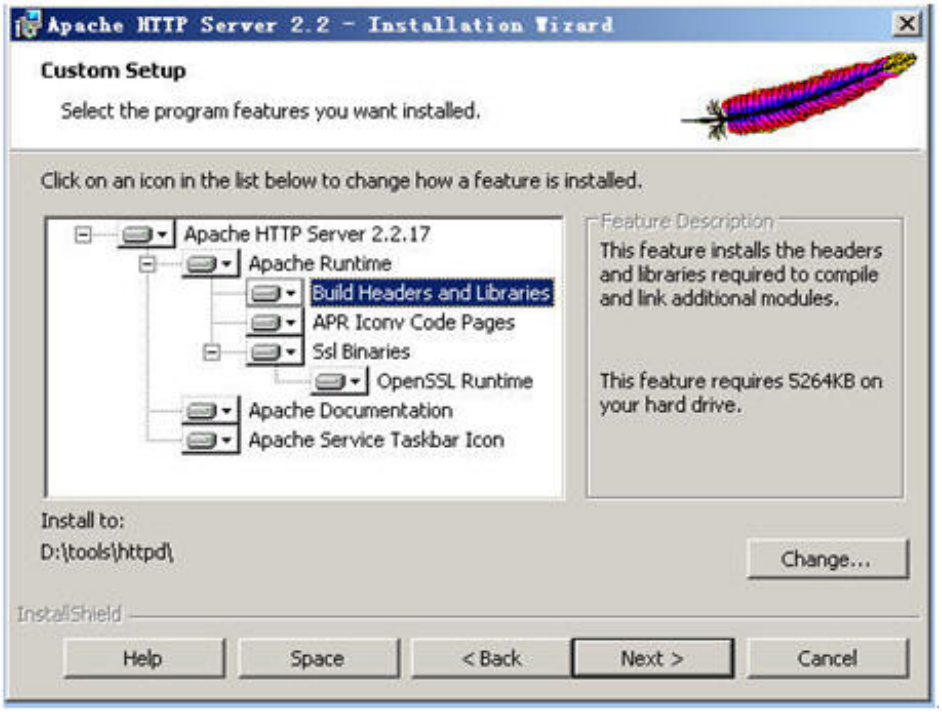
我們將安裝這個版本的apache http server作為我們今後一直使用的Web Server





這邊的server name你們要填入自己的server的真實名，不能用我這個，這個servername如：shnlap93.cts.com只能夠我用，這個名稱是全局唯一的，和你的IP一樣。





選全部安裝

裝完後你會多出一個這樣的圖標來，點擊該圖標，裡面有用於控制apache http server的啟動、停止與重啟等操作選項。同時在你們的「服務」面板中，也能發現這樣的一個服務項，它啟動時默認是隨著系統的啟動而啟動的，我們把它改成「手動」吧，因為將來我們還要安裝IBM Http Server來作練習。

裝完後，在Apache2.2啟動的前提下，打開一個ie輸入http://localhost，你將會得到這樣的一個頁面，就說明你的Apache的安裝是成功的。

2.5 Apache的配置

學Java的人，必須會這個Apache的配置，要不然你怎麼模擬環境、搭建環境和架構環境？光會Coding是遠遠不夠的，你將永遠只配作個碼農。。。嘿嘿嘿！有很多人發覺到了後面JAVA學不上去了，關鍵因素在於：配置。

你會配環境了，那麼你就能模擬任何客戶方、開發方的環境。

你會配環境了，你的代碼將來上線時才能成功運行。

你會配環境了，所以整個工程的技術核心就是你。

跟著我的教程，你們將會安裝和運行達近百個各種軟件與配置，搞得你一股臭味一股臭味！！

你準備好了沒有？

當然，不用怕，因為我的配置都是實際運行的環境，所以網上的一些東西你可以不用去看，因為很多人都是在網上進行拷貝、複製，有時也不經過驗證，會讓你走很多的彎路到頭來還是落得個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，它位於你的安裝目錄，比如：

D:\tools\httpd\conf

我們用ultraedit或者相關文本編輯工具打開它，來看它的內容：

先來查找到如下這一行：

#ServerName

我們可以得到如下這一行內容：

```
#ServerName shnlap93.cts.com:80
```

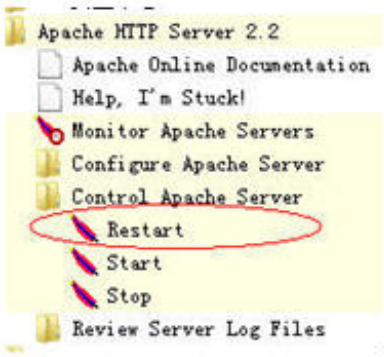
這就是我們的主機名了，我們可以將前面的「#」去掉，並將其改為：

```
ServerName 10.225.106.35:80
```

改完後存盤，在重啟你的Apache2.2前我們先測試一下我們的Apache的配置文件是否改得對：

如果在你點了Test Configuration後，黑屏一閃而過，說明你的改動無誤，否則這個黑屏會一直停留在當前狀態，並且告訴你，你的配置改動有錯，錯在哪裡。

重新啟動你的Apache



找到如下這行：

DocumentRoot

你會發下有這樣的一行內容：

```
DocumentRoot "D:/tools/httpd/htdocs"
```

這個叫作DocumentRoot即webroot，即：發佈目錄，發佈在這個目錄下的任何工程都會在Apache服務開啟時被裝載成標準的web工程，我們現在動手來把這個WebRoot定位到我們自己的發布目錄中去吧。

```
DocumentRoot "d:/www"
```

我們把它改到了d盤的www目錄中去了，然後我們在該目錄中放入一個index.html文件，內容為：

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

重啟我們的Apache服務，來測試一下：

嘿嘿，我們得到了什麼？禁止訪問，為什麼？

找到下面這一段：

```
<Directory />
    Options FollowSymLinks
    AllowOverride None
    Order deny,allow
    deny from all
</Directory>
```

看到了沒？

現在，把這個」deny from all」改成」allow fromall'吧。

```
<Directory />
    Options FollowSymLinks
    AllowOverride None
    Order deny,allow
    allow from all
</Directory>
```

修改完後重啟你的Apache服務

Ok,我們的Apache的發布目錄已經成功更改到了d:\www目錄下了，我們再來做一個實驗：

我們在IE瀏覽器中輸入: http://localhost/css/，我們看到了什麼？

這還了得，用戶如果是個初級黑客都可以知道我們的服務器上有哪些文件，哪些目錄甚至可以直接看到我們的文件內容，怎麼辦？

找到下面這行

```
Options FollowSymLinks indexes
```

把它注掉改成下面這樣

```
#Options FollowSymLinks indexes
Options None
```

不要急，再往下找，還有

```
Options Indexes FollowSymLinks
```

又來一個，再改掉

```
#Options Indexes FollowSymLinks
Options None
```

改完這兩條後重啟你的Apache服務

再次打開一個新的IE，輸入：http://localhost/css/，我們看到了如下的界面：

好了，Apache的基本配置完成了即：

1） 基本的安全配置，不允許目錄訪問

2） 把WebRoot改到另一個物理目錄上而不使用Apache自帶的WebRoot目錄

2.6 整合Apache與Tomcat

Apache(Web Server)負責處理HTML靜態內容；

Tomcat(App Server)負責處理動態內容；

其實就是上述這樣的一個架構，下面是原理

1) Apache裝有一個模塊，這個模塊叫mod_jk

2) Apache通過80端口負責解析任何靜態web內容

3) 任何不能解析的內容，用表達式告訴mod_jk，讓mod_jk**派發**給相關的app server去解釋。

通過上述的文字描述我們可以得知：

1） 我們需要在Apache中先裝一個mod_jk

2） 我們需要在httpd.conf中寫點表達式

下面來實現。

1) 把mod_jk-1.2.31-httpd-2.2.3.so手工copy進我們的Apache安裝目錄的modules目錄下，這個文件的全名叫:mod_jk-1.2.31-httpd-2.2.3.so，大家可以從ftp上的「/JavaArchitect/mod_jk/」目錄中獲取，因為這個文件是我用C++在本地重新編譯過的，網上下載的是src即源碼，省去大家再去編譯的時間了，而且一些其它網上下載的mod_jk.so是無法使用的。

2) 用ultraedit打開httpd.conf文件，跑到文件最後面加入以下幾行：

```
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>
```

關鍵的是這兩句：

LoadModule jk_module modules/mod_jk-1.2.31-httpd-2.2.3.so

JKWorkersFile conf/workers.properties

代表：

ü Apache載入一個額外的插件，用於連接tomcat。

ü 連接時的配置參數描述位於Apache安裝目錄的/conf目錄下的一個叫workers.properties文件中，mod_jk一般使用ajp13協議連接，使用的是tomcat的8009端口。

3) Worker.properties文件內容如下：

```
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) 告訴我們的Apache，哪些是要交給tomcat來解析，**除此之外都由Apache本身來解析**：

```
<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>
```

大家看到沒，所有的/servlet/*都由tomcat負責解析，所有的jsp, .do, .action都由tomcat解析。

此處還有一個特殊的/fckeditor，這個是我們使用的一個博客編輯器，這個因為是servlet的，因此也需要交給tomcat鐵析。

5) 將/cbbs工程佈署到tomcat的webapps目錄下

6)　將/cbbs同樣手工copy一份到d:/www目錄下

7)　刪除d:/www/cbbs/WEB-INF這個目錄，嘿嘿，因為d:/www下的東西是由Apache解析的，所有的WEB-INF下的都是Java，我們只需要佈署在tomcat下即可，是不是？

8)　重啟tomcat，重啟Apache，在ie中直接輸入： http://localhost/cbbs，使用sally/abcdefg登錄，操作一下，一切成功

Oh…yeah, tomcat+apache一步搞定。

三、用於實驗的cbbs工程配置

最後附上cbbs佈署需要用到的配置，相關的工程可通過ftp的」 /Java Architect/Project/」 下的cbbs.zip來獲取。

ü　在tomcat中打開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盤根目錄建立一個upload目錄，在此目錄內再建立一個image目錄。

ü　在tomcat中打開context.xml加入

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

我的同類文章

- •2016-05-11
- •2016-04-14

- •2016-04-13
- •2016-03-14
- •2016-02-03
- •2015-03-09
- •2016-05-05
- •2016-04-14
- •2016-03-30
- •2016-02-14
- •2016-01-27

更多文章