# 官网

http://tomcat.apache.org/

# 基本概念

Tomcat是Apache 软件基金会的Jakarta 项目中的一个核心项目，由Apache、Sun 和其他一些公司及个人共同开发而成。由于有了Sun 的参与和支持，最新的Servlet 和JSP 规范总是能在Tomcat 中得到体现。因为Tomcat 技术先进、性能稳定，而且免费，因而深受Java 爱好者的喜爱并得到了部分软件开发商的认可，成为目前比较流行的Web 应用服务器。

Tomcat 服务器是一个免费开放源代码的Web 应用服务器，属于轻量级应用[服务器](https://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E5%99%A8)，在中小型系统和并发访问用户不是很多的场合下被普遍使用，是开发和调试JSP 程序的首选。对于一个初学者来说，可以这样认为当在一台机器上配置好Apache 服务器，可利用它响应[HTML](https://baike.baidu.com/item/HTML)页面的访问请求。实际上Tomcat是Apache 服务器的扩展，但运行时它是独立运行的，所以当你运行tomcat 时，它实际上作为一个与Apache 独立的进程单独运行的。

诀窍是当配置正确时，Apache 为HTML页面服务，而Tomcat 实际上运行JSP 页面和Servlet。另外Tomcat和[IIS](https://baike.baidu.com/item/IIS)等Web服务器一样，具有处理HTML页面的功能，另外它还是一个Servlet和JSP容器，独立的Servlet容器是Tomcat的默认模式。不过Tomcat处理静态[HTML](https://baike.baidu.com/item/HTML)的能力不如Apache服务器。

# 版本差异

## Tomcat6.x

建立在Tomcat5.5的改进和实现Servlet 2.5和 JSP 2.1规范, 除此之外，它还包括以下改进

（1）内存使用优化

（2）高级IO功能

（3）重构聚类

## Tomcat7.x

建立在Tomcat6.0.x的改进和实现Servlet 3.0， JSP 2.2，EL 2.2和 WebSocket的1.1规格。除此之外，它还包括以下改进：

（1）Web应用程序内存泄露检测和预防

（2）提高了Manager和Host Manager应用程序的安全性

（3）通用CSRF保护

（4）支持直接在Web应用程序中包含外部内容

（5）重构（连接器，生命周期）和大量的内部代码清理

## Tomcat8.x

建立在Tomcat7.x基础上并实施Servlet 3.1，JSP 2.3，EL 3.0 和WebSocket的1.1规格。除此之外，还包括以下重大改进：

单个公共资源实现来替换早期版本中提供的多个资源扩展特性

Apache Tomcat 8.5.x支持相同的Servlet，JSP，EL和WebSocket规范的版本， 除此之外，它也实现了JASPIC 1.1规范。还有在许多领域显著变化引擎下，从而提高了性能、稳定性和总拥有成本。

## Tomcat9.x

建立在Tomcat 8.0.x和实现了目前草案的Servlet 4.0规范，也将执行 JSP 2.4、EL 3.1，目前对WebSocket的1.2 和JASPIC 1.1 规范工作的一次更新上这些规范为Java EE 8除此之外启动时，它包括以下显著改进：

（1）添加对HTTP / 2的支持（需要APR /本地库）

（2）添加对TLS虚拟主机的支持

（3）添加了对使用JSSE连接器（NIO和NIO2）使用OpenSSL for TLS支持的支持

**Alpha**版本可能含有大量的规范和/或显著的bug需要未经测试/缺少的功能，并且预计不会稳定地任何时间运行。

**Beta**版本中可能含有一些未经测试的功能和/或一些相对较小的错误。Beta版本预计不会稳定运行。

**Stable**的版本可能包含少量相对较小的错误。稳定的释放用于生产使用，预计可以稳定运行长时间。

# Tomcat目录结构

## bin

存放启动和关闭tomcat脚本

## conf

包含不同的配置文件，server.xml和web.xml是tomcat的主要配置文件。

## work

存放jsp编译后产生的class文件。

## webapps

存放应用程序示例，以后你要部署的应用程序也要放到此目录下。

## temp

存放Tomcat运行时所产生的临时文件

## logs

存放日志文件

## lib

主要存放tomcat所需的jar文件。

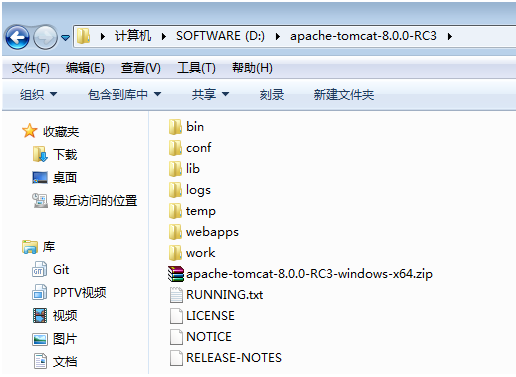
# 安装步骤

## 1、下载和安装Java

安装和设置JAVA\_HOME环境变量

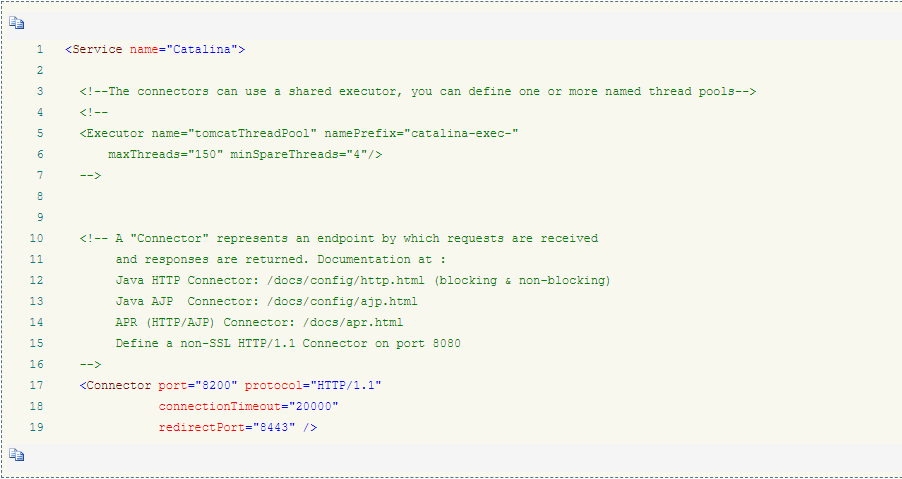
## 2、下载和解压Tomcat

下载地址：<http://tomcat.apache.org/>



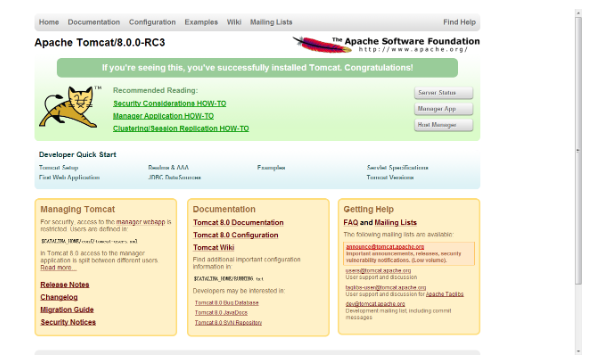
## 3、修改端口号

修改D:\apache-tomcat-8.0.0-RC3\conf\server.xml



## 4、测试Tomcat

运行：D:\apache-tomcat-8.0.0-RC3\bin\startup.bat，如果窗口没有立即关闭就代表成功了。



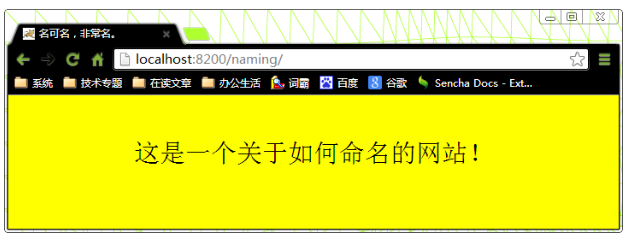
## 5、发布自己的网站

最简单的发布方式莫过于将项目直接拷贝到：D:\apache-tomcat-8.0.0-RC3\webapps，这里介绍另外一种方式

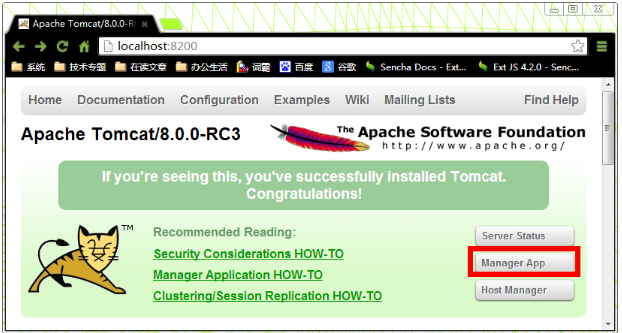
在D:\apache-tomcat-8.0.0-RC3\conf\Catalina\localhost下面建立一个naming.xml文件，内容如下：



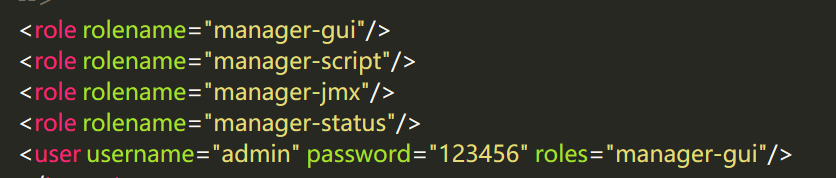
效果：



## 6、管理Web App



当点击 Manager App 的时候，会要求输入用户名和密码，可以在 D:\apache-tomcat-8.0.0-RC3\conf\tomcat-users.xml 中增加用户名和密码，如下：



说明：角色名一定不能修改，按照上面的去写，用户名和密码可以自己去定义。

请注意，对于Tomcat 7以上，使用管理器应用程序所需的角色已从单个管理器角色更改为以下四个角色。 您需要分配您希望访问的功能所需的角色。

（1）manager-gui - 允许访问HTML GUI和状态页面

（2）manager-script - 允许访问文本接口和状态页面

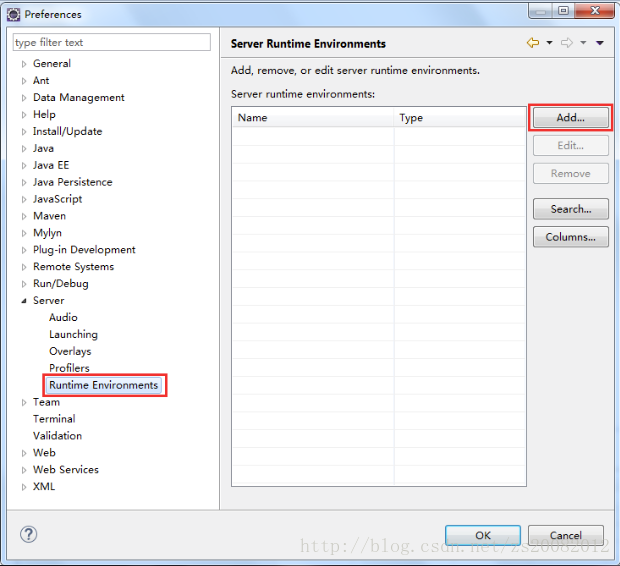
（3）manager-jmx - 允许访问JMX代理和状态页面

（4）manager-status -仅允许访问状态页面

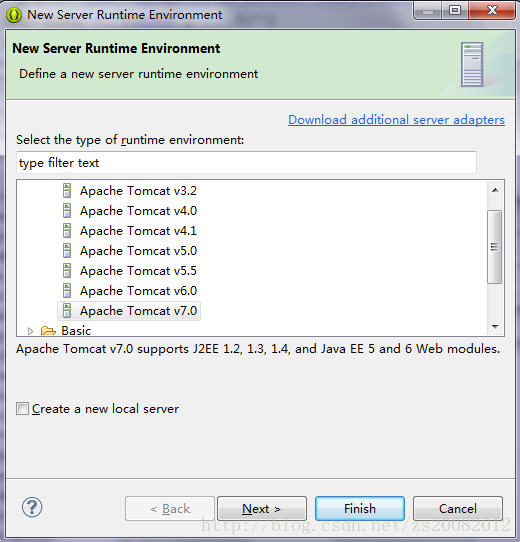
注意：这是因为 Tomcat 支持 JAAS。

# eclipse配置tomcat

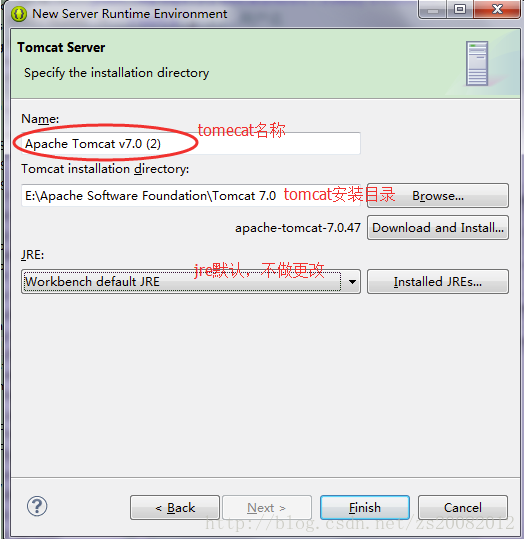
打开Eclipse，单击“Window”菜单，选择下方的“Preferences”，看是否存在“Server”选项，如图：



上图中，点击“add”的添加按钮，弹出如下界面：

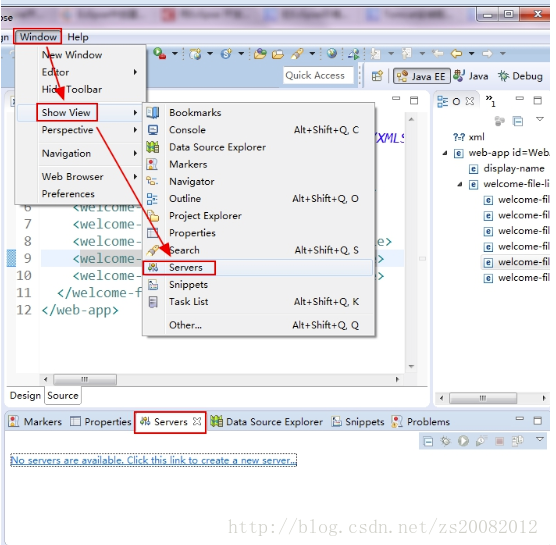


上图中，选择对应的Tomcat版本，点击next，如图

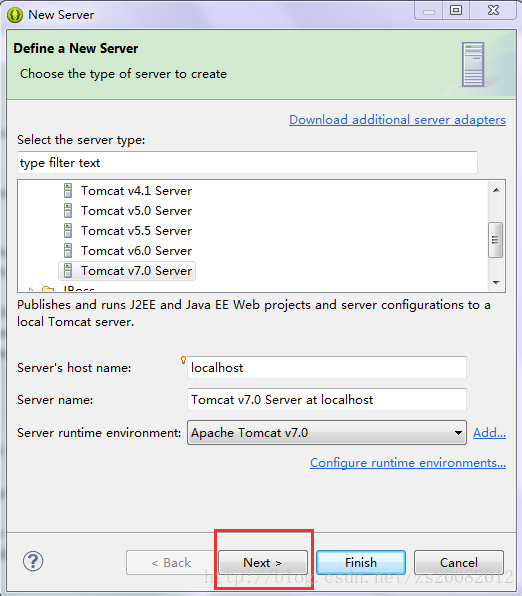


上图中，选择Tomcat的路径，以及JRE（使用工作平台默认的，不用做额外添加修改），点击“完成”，配置完毕

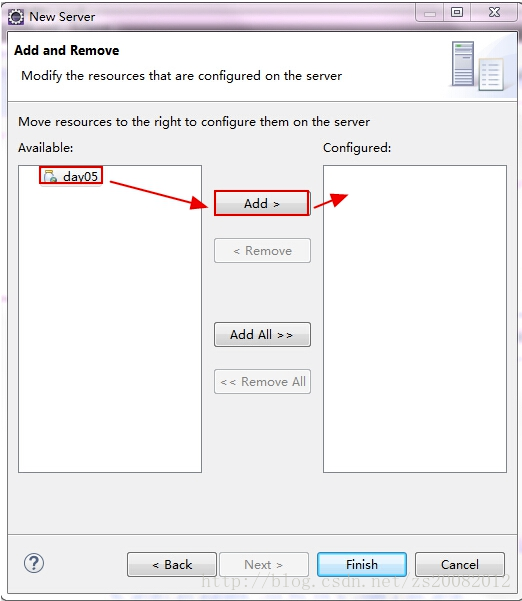
让tomcat服务器显示在控制台上，将web应用部署到tomcat中，Window → Show View → Servers如图



点击链接No servers are available. Click ths link to create a new server. ，在弹出的对话框中选择Tomcat版本 如图

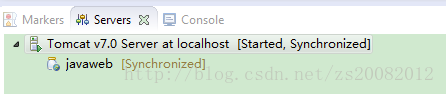


点击“Next”，添加我们的项目 如图：，点击finish

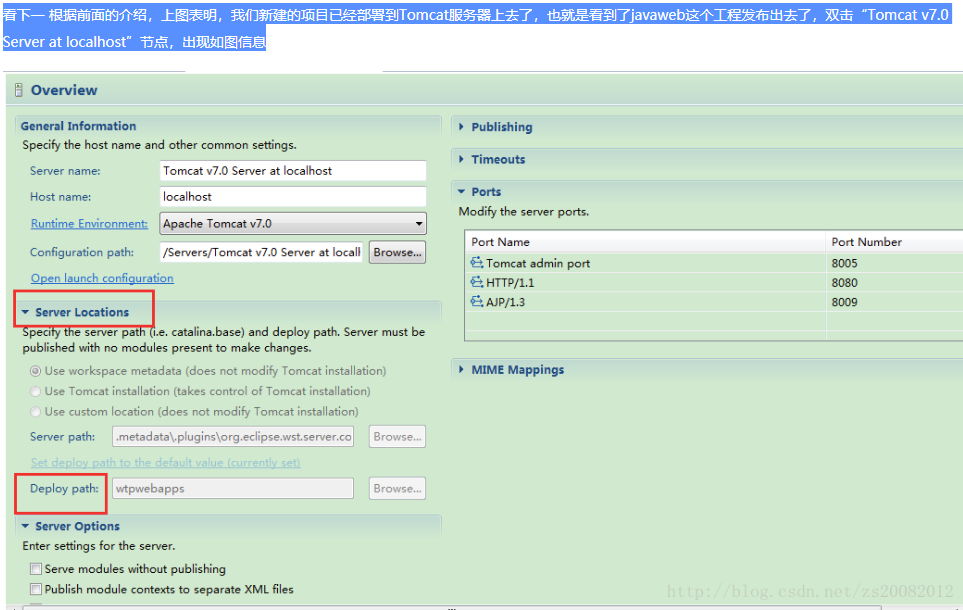


点击finish完成

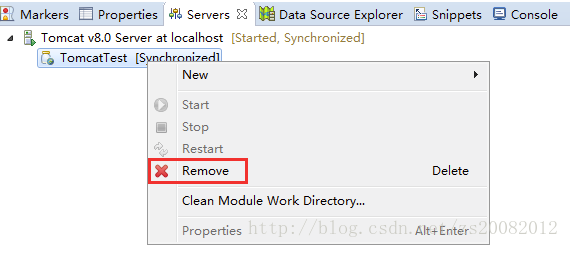
再看下一个细节



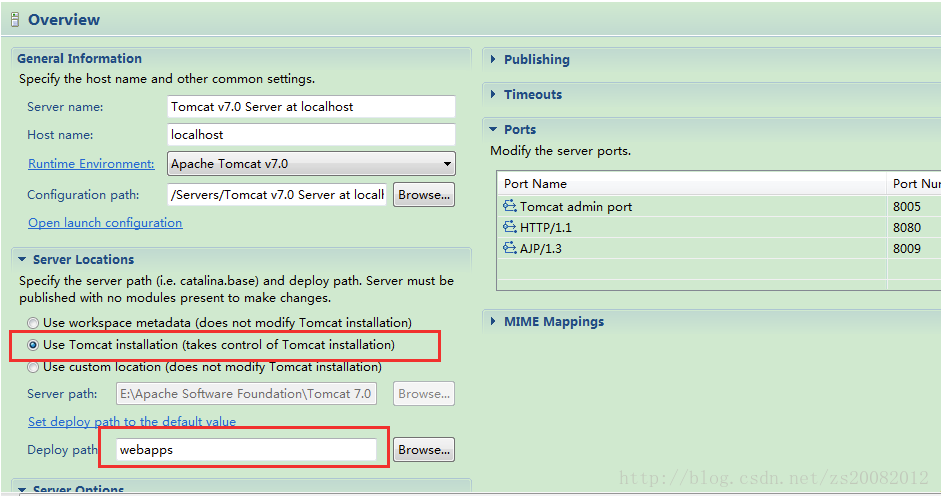
看下一 根据前面的介绍，上图表明，我们新建的项目已经部署到Tomcat服务器上去了，也就是看到了javaweb这个工程发布出去了，双击“Tomcat v7.0 Server at localhost”节点，出现如图信息



上图的红框部分表明，服务的部署是在eclipse里面（默认是放在工作空间里的.metadata文件夹下），而不是在Tomcat里面。我们来改一下，前提是工程并没有发布到Tomcat中去，那我们先把之前发布的版本删了吧（稍后重新发布）



这时再次双击“Tomcat v7.0 Server at localhost”节点，使用Tomcat的安装目录作为部署的位置，并修改部署的路径Deploy path（建议改成Tomcat的webapps目录下），然后保存即可。



这时，重新运行程序。现在来到Tomcat的webapps目录下，发现多了一个javaweb文件夹（即工程文件名），并且文件夹下包含了工程文件中WebContent的内容：



上图说明，说明这才是真正将程序发布到服务器上。

# context.xml文件（重点）

首先需要为数据源配置一个JNDI资源。 我们的数据源JNDI资源应该定义在context元素中。 在tomcat6版本中，context元素已经从server.xml文件中独立出来了，放在一个context.xml文件中。因为server.xml是不可动态重加载的资源，服务器一旦启动了以后，要修改这个文件，就得重启服务器才能重新加载，而context.xml文件则不然，tomcat服务器会定时去扫描这个文件，一旦发现文件被修改（时间戳改变了），就会自动重新加载这个文件而不需要重启服务器。我们当然推荐把应用需要的JNDI资源配置在context.xml文件中，而不是server.xml文件中。首先将数据库的驱动程序copy到tomcat6.0\lib下，假如没有copy，当运行程序的时候报找不到驱动的异常。

将下面的代码放到Tomcat 6.0\conf\context.xml中，如：

<Context reloadable="true">

<Resource

name="jdbc/数据库名"

type="javax.sql.DataSource"

maxActive="100"

maxIdle="30"

maxWait="5000"

username="用户名"

password="密码"

driverClassName="数据库的驱动"

url="数据库的连接地址" />

</Context>

解释：

maxActive="最大可以有100名用户连接数据源"

maxIdle="假如没有用户连接，空出30个连接等待用户连接"

maxWait="假如已连接用户5000秒内没有再次连接数据源，则放弃此连接"。

# 各种数据库的配置

说明：name是自定义的

## （1）sql2000

<Resource

name="jdbc/数据库名"

type="javax.sql.DataSource"

maxActive="100"

maxIdle="30"

maxWait="5000"

username="用户名"

password="密码"

driverClassName="com.microsoft.jdbc.sqlserver.SQLServerDriver"

url="jdbc:microsoft:sqlserver://127.0.0.1:1433;DatabaseName=数据库名"

/>

## （2）oracle

<Resource

name="jdbc/数据库名"

type="javax.sql.DataSource"

maxActive="100"

maxIdle="30"

maxWait="5000"

username="用户名"

password="密码"

driverClassName="oracle.jdbc.driver.OracleDriver"

url="jdbc:oracle:thin:@localhost:1521:ora9"

/>

## （3）mysql

<Resource name="jdbc/mysql"

auth="Container"

type="javax.sql.DataSource" //资源类型

driverClassName="com.mysql.jdbc.Driver"

url="jdbc:mysql://localhost/数据库名"

username="用户名"

password="密码"

maxActive="100" //最大连接数

maxIdle="30" //最大空闲时间，0为无限制

maxWait="10000"/> //建立连接的的最大等待时间

# 如何优化Tomcat6.0服务器

## 1、使用Tomcat的线程池

只需要在server.xml中改改配置就可以了。

核心代码：

<Executor name="tomcatThreadPool" namePrefix="catalina-exec-" maxThreads="1000" minSpareThreads="50" maxIdleTime="600000"/>

## 2、调整platform数据源的线程设置

假设发布目录为Webroot，那么在WebRoot/META-INF/下建立context.xml文件，调整数据源配置

<?xml version=”1.0″ encoding=”UTF-8″?>

<Context path= “/WebRoot ” privileged= “true” reloadable=”false”>

<Resource name=”sysDataSource” auth=”Container”

type=”javax.sql.DataSource” maxActive=”5000″ maxIdle=”300″ maxWait=”60000″

logAbandoned=”true” username=”yfjz” password=”password”

driverClassName=”oracle.jdbc.driver.OracleDriver”

url=”jdbc:oracle:thin:@10.10.10.XX:1521:al32″ />

<Resource name=”yfjzDataSource” auth=”Container”

type=”javax.sql.DataSource” maxActive=”4000″ maxIdle=”200″ maxWait=”60000″

logAbandoned=”true” username=”yfjz” password=”password”

driverClassName=”oracle.jdbc.driver.OracleDriver”

url=”jdbc:oracle:thin:@10.10.10.XX:1521:al32″ />

</Context>

## 3、调整Tomcat6响应池

查找tomcat6安装目录下conf目录中的server.xml文件，进行编辑，屏蔽tomcat默认Connector。

<!–

<Connector port=”8080″ protocol=”HTTP/1.1″

connectionTimeout=”20000″

redirectPort=”8443″ />

–>

创建高线程的Connector：

<Connector port=”8080″ redirectPort=”8443″

maxHttpHeaderSize=”8192″ useBodyEncodingForURI=”true”

minProcessors=”100″ maxProcessors=”5000″

maxThreads=”5000″ minSpareThreads=”1000″ maxSpareThreads=”4000″

enableLookups=”false” acceptCount=”3500″

compression=”on” compressionMinSize=”2048″

compressableMimeType=”text/html,text/xml,text/javascript,text/css,text/plain”

connectionTimeout=”60000″ disableUploadTimeout=”true” debug=”0″ URIEncoding=”UTF-8″/>

（注：加入响应线程数控制，加入压缩传递模式，调整超时设置，屏蔽调试模式）

## 4、增加tomcat6启动内存

查找tomcat6安装目录下bin目录中catalina.sh文件，在开始增加如下：

JAVA\_OPTS=” -Xms1400m -Xmx1400m -XX:PermSize=64M -XX:MaxNewSize=256m -XX:MaxPermSize=128m -Djava.awt.headless=true ”

## 5、增加oracle响应线程数

王老五将oracle数据线程响应定为1000,这是测试用的，同时监控oracle链接资源

(想知道这个怎么设置的，认识王老五的就找他聊聊，不认识的就网上找找吧，呵呵)

## 6、开始压测1000户

晶晶开始测试后，确发现有十来户一直处于等待状态，R61本本cpu资源从96%降到10%以下，硬盘灯狂闪，Desktop系统操作响应减缓使用硬盘资源检查命令发现，硬盘使用资源为100%，说明硬盘满了，系统在寻找缓存操作，硬盘和内存疯狂交互这真是“一波刚平一波起，平湖落石浪千层……”