

-

整理自互联网，仅供学习交流。



2018-10-30

[公司名称]

[公司地址]

目录

[1.1 配置 1](#_Toc8422882)

[1.2 Tomcat优化 1](#_Toc8422883)

[1.2.1 基本思路 1](#_Toc8422884)

[1.2.2 Tomcat的基本优化 2](#_Toc8422885)

[1.2.3 Tomcat本身的配置优化 2](#_Toc8422886)

[1.2.4 Tomcat三种模式 3](#_Toc8422887)

# 配置

## 默认项目

需求：

把发布到Tomcat下的web项目，访问路径去掉项目名称

### 实现方式及原理：

#### 方式一：

原理：Tomcat的默认根目录是ROOT，实际上ROOT这个项目在实际生产环境是没有用的，所以我们可以用我们的项目覆盖ROOT项目

操作过程：

1.删除ROOT下所有文件及文件夹

2.把我们项目的war包解压后，项目目录下的所有文件和子目录都拷贝到ROOT目录下即可

或者有更狠的一招：直接删掉ROOT目录，然后把我们的项目打包名称改成ROOT.war，放到webapps下就行

#### 方式二：

原理：Tomcat本身可以配置虚拟目录。方法就是在Server.xml中<Engine><Host>节点下加入Context信息。如我们可以配置<Context path="/abc" docBase="D:\app\abc" ... />，那我们可以通过地址http://localhost:8080/abc来访问我们放在D:\app\下面的abc项目。我们可以把这个path="/abc"修改为path=""。意思就是把abc映射到根目录，访问路径就会变成http://localhost:8080/。

操作过程：

按照配置虚拟目录的方式，在<Engine><Host>下添加一个Context节点，具体配置如下：

Xml代码

<Engine name="Catalina" defaultHost="localhost"...>

...

<Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">

<Context path="" docBase="Interface" reloadable="true" />

<!--注：我这里使用的是相对路径，Interface项目是放在Tomcat的webapps目录下的，当然也可以改为绝对路径-->

...

</Host>

...

</Engine>

访问方式就可以用http://localhost:8080/SearchReqService.asmx?wsdl了

如果用虚拟目录的方式，地址http://localhost:8080/Interface/SearchReqService.asmx?wsdl也可以访问。

同样的方式，我们可以为path指定不同的路径，解决访问路径区别项目名称的需要。

其它，去掉访问的端口号8080.就是利用了HTTP请求访问的端口默认是80的方式实现的，iis也一样。我们只用把Tomcat的HTTP监听端口号改为80（修改<Connector port="8080" protocol="HTTP/1.1"这里的端口号为80）即可。

### 注意

tomcat的server.xml配置文件中的<Host>标签中加<Context>，docBase=项目路径加名称（不带.war后缀名）

<Host name="localhost"  appBase="webapps"

            unpackWARs="true" autoDeploy="true"

            xmlValidation="false" xmlNamespaceAware="false">

<Context path="" docBase="$ProjectName" debug="0" reloadable="true"></Context>

</Host>

reloadable="true/false" 启用tomcat热加载

或部署多个项目/Context，参数path后项目名称不同即可

访问：http://$ip:$port

tomcat 端口指定范围：1~65535（如果访问tomcat没有反应也有可能是端口号改大了，我就被这个坑过）

### 完整代码

|  |
| --- |
| <?xml version='1.0' encoding='utf-8'?>  <!-- Licensed to the Apache Software Foundation (ASF) under one or more contributor  license agreements. See the NOTICE file distributed with this work for additional  information regarding copyright ownership. The ASF licenses this file to  You under the Apache License, Version 2.0 (the "License"); you may not use  this file except in compliance with the License. You may obtain a copy of  the License at http://www.apache.org/licenses/LICENSE-2.0 Unless required  by applicable law or agreed to in writing, software distributed under the  License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS  OF ANY KIND, either express or implied. See the License for the specific  language governing permissions and limitations under the License. -->  <!-- Note: A "Server" is not itself a "Container", so you may not define  subcomponents such as "Valves" at this level. Documentation at /docs/config/server.html -->  <Server port="8005" shutdown="SHUTDOWN">  <Listener className="org.apache.catalina.startup.VersionLoggerListener" />  <!-- Security listener. Documentation at /docs/config/listeners.html <Listener  className="org.apache.catalina.security.SecurityListener" /> -->  <!--APR library loader. Documentation at /docs/apr.html -->  <Listener className="org.apache.catalina.core.AprLifecycleListener"  SSLEngine="on" />  <!--Initialize Jasper prior to webapps are loaded. Documentation at /docs/jasper-howto.html -->  <Listener className="org.apache.catalina.core.JasperListener" />  <!-- Prevent memory leaks due to use of particular java/javax APIs -->  <Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener" />  <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />  <Listener className="org.apache.catalina.core.ThreadLocalLeakPreventionListener" />  <!-- Global JNDI resources Documentation at /docs/jndi-resources-howto.html -->  <GlobalNamingResources>  <!-- Editable user database that can also be used by UserDatabaseRealm  to authenticate users -->  <Resource name="UserDatabase" auth="Container"  type="org.apache.catalina.UserDatabase" description="User database that can be updated and saved"  factory="org.apache.catalina.users.MemoryUserDatabaseFactory"  pathname="conf/tomcat-users.xml" />  </GlobalNamingResources>  <!-- A "Service" is a collection of one or more "Connectors" that share  a single "Container" Note: A "Service" is not itself a "Container", so you  may not define subcomponents such as "Valves" at this level. Documentation  at /docs/config/service.html -->  <Service name="Catalina">  <!--The connectors can use a shared executor, you can define one or more  named thread pools -->  <!-- <Executor name="tomcatThreadPool" namePrefix="catalina-exec-" maxThreads="150"  minSpareThreads="4"/> -->  <!-- A "Connector" represents an endpoint by which requests are received  and responses are returned. Documentation at : Java HTTP Connector: /docs/config/http.html  (blocking & non-blocking) Java AJP Connector: /docs/config/ajp.html APR (HTTP/AJP)  Connector: /docs/apr.html Define a non-SSL HTTP/1.1 Connector on port 8080 -->  <Connector port="8079" protocol="HTTP/1.1"  connectionTimeout="20000" redirectPort="8442" />  <!-- A "Connector" using the shared thread pool -->  <!-- <Connector executor="tomcatThreadPool" port="8080" protocol="HTTP/1.1"  connectionTimeout="20000" redirectPort="8443" /> -->  <!-- Define a SSL HTTP/1.1 Connector on port 8443 This connector uses the  BIO implementation that requires the JSSE style configuration. When using  the APR/native implementation, the OpenSSL style configuration is required  as described in the APR/native documentation -->  <!-- <Connector port="8443" protocol="org.apache.coyote.http11.Http11Protocol"  maxThreads="150" SSLEnabled="true" scheme="https" secure="true" clientAuth="false"  sslProtocol="TLS" /> -->  <!-- Define an AJP 1.3 Connector on port 8009 -->  <Connector port="8008" protocol="AJP/1.3" redirectPort="8442" />  <!-- An Engine represents the entry point (within Catalina) that processes  every request. The Engine implementation for Tomcat stand alone analyzes  the HTTP headers included with the request, and passes them on to the appropriate  Host (virtual host). Documentation at /docs/config/engine.html -->  <!-- You should set jvmRoute to support load-balancing via AJP ie : <Engine  name="Catalina" defaultHost="localhost" jvmRoute="jvm1"> -->  <Engine name="Catalina" defaultHost="localhost">  <!--For clustering, please take a look at documentation at: /docs/cluster-howto.html  (simple how to) /docs/config/cluster.html (reference documentation) -->  <!-- <Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/> -->  <!-- Use the LockOutRealm to prevent attempts to guess user passwords  via a brute-force attack -->  <Realm className="org.apache.catalina.realm.LockOutRealm">  <!-- This Realm uses the UserDatabase configured in the global JNDI resources  under the key "UserDatabase". Any edits that are performed against this UserDatabase  are immediately available for use by the Realm. -->  <Realm className="org.apache.catalina.realm.UserDatabaseRealm"  resourceName="UserDatabase" />  </Realm>  <Host name="localhost" appBase="webapps" unpackWARs="true"  autoDeploy="true">  <!-- SingleSignOn valve, share authentication between web applications  Documentation at: /docs/config/valve.html -->  <!-- <Valve className="org.apache.catalina.authenticator.SingleSignOn"  /> -->  <Context path="" docBase="navigation" debug="0" reloadable="true" />  <!-- Access log processes all example. Documentation at: /docs/config/valve.html  Note: The pattern used is equivalent to using pattern="common" -->  <Valve className="org.apache.catalina.valves.AccessLogValve"  directory="logs" prefix="localhost\_access\_log." suffix=".txt"  pattern="%h %l %u %t &quot;%r&quot; %s %b" />  </Host>  </Engine>  </Service>  <Service name="Catalina1">  <!--The connectors can use a shared executor, you can define one or more  named thread pools -->  <!-- <Executor name="tomcatThreadPool" namePrefix="catalina-exec-" maxThreads="150"  minSpareThreads="4"/> -->  <!-- A "Connector" represents an endpoint by which requests are received  and responses are returned. Documentation at : Java HTTP Connector: /docs/config/http.html  (blocking & non-blocking) Java AJP Connector: /docs/config/ajp.html APR (HTTP/AJP)  Connector: /docs/apr.html Define a non-SSL HTTP/1.1 Connector on port 8080 -->  <Connector port="8081" protocol="HTTP/1.1"  connectionTimeout="20000" redirectPort="8443" />  <!-- A "Connector" using the shared thread pool -->  <!-- <Connector executor="tomcatThreadPool" port="8080" protocol="HTTP/1.1"  connectionTimeout="20000" redirectPort="8443" /> -->  <!-- Define a SSL HTTP/1.1 Connector on port 8443 This connector uses the  BIO implementation that requires the JSSE style configuration. When using  the APR/native implementation, the OpenSSL style configuration is required  as described in the APR/native documentation -->  <!-- <Connector port="8443" protocol="org.apache.coyote.http11.Http11Protocol"  maxThreads="150" SSLEnabled="true" scheme="https" secure="true" clientAuth="false"  sslProtocol="TLS" /> -->  <!-- Define an AJP 1.3 Connector on port 8009 -->  <!--<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />-->  <!-- An Engine represents the entry point (within Catalina) that processes  every request. The Engine implementation for Tomcat stand alone analyzes  the HTTP headers included with the request, and passes them on to the appropriate  Host (virtual host). Documentation at /docs/config/engine.html -->  <!-- You should set jvmRoute to support load-balancing via AJP ie : <Engine  name="Catalina" defaultHost="localhost" jvmRoute="jvm1"> -->  <Engine name="Catalina1" defaultHost="localhost">  <!--For clustering, please take a look at documentation at: /docs/cluster-howto.html  (simple how to) /docs/config/cluster.html (reference documentation) -->  <!-- <Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/> -->  <!-- Use the LockOutRealm to prevent attempts to guess user passwords  via a brute-force attack -->  <Realm className="org.apache.catalina.realm.LockOutRealm">  <!-- This Realm uses the UserDatabase configured in the global JNDI resources  under the key "UserDatabase". Any edits that are performed against this UserDatabase  are immediately available for use by the Realm. -->  <Realm className="org.apache.catalina.realm.UserDatabaseRealm"  resourceName="UserDatabase" />  </Realm>  <Host name="localhost" appBase="webapps1" unpackWARs="true"  autoDeploy="true">  <!-- SingleSignOn valve, share authentication between web applications  Documentation at: /docs/config/valve.html -->  <!-- <Valve className="org.apache.catalina.authenticator.SingleSignOn"  /> -->  <Context path="" docBase="wn" debug="0" reloadable="true" />  <!-- Access log processes all example. Documentation at: /docs/config/valve.html  Note: The pattern used is equivalent to using pattern="common" -->  <Valve className="org.apache.catalina.valves.AccessLogValve"  directory="logs" prefix="localhost\_access\_log." suffix=".txt"  pattern="%h %l %u %t &quot;%r&quot; %s %b" />  </Host>  </Engine>  </Service>  </Server> |

## user

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  -<tomcat-users>  <!--NOTE: By default, no user is included in the "manager-gui" role requiredto operate the "/manager/html" web application. If you wish to use this app,you must define such a user - the username and password are arbitrary. It isstrongly recommended that you do NOT use one of the users in the commented outsection below since they are intended for use with the examples webapplication. -->  <!--NOTE: The sample user and role entries below are intended for use with theexamples web application. They are wrapped in a comment and thus are ignoredwhen reading this file. If you wish to configure these users for use with theexamples web application, do not forget to remove the <!.. ..> that surroundsthem. You will also need to set the passwords to something appropriate. -->  <!--<role rolename="tomcat"/><role rolename="role1"/><user username="tomcat" password="<must-be-changed>" roles="tomcat"/><user username="both" password="<must-be-changed>" roles="tomcat,role1"/><user username="role1" password="<must-be-changed>" roles="role1"/> -->  <role rolename="manager-gui"/>  <role rolename="manager-script"/>  <role rolename="manager-jmx"/>  <role rolename="manager-status"/>  <role rolename="admin-gui"/>  <role rolename="admin-script"/>  <user roles="manager-gui, manager-script, manager-jmx, manager-status, admin-gui, admin-script" password="yaoSIyuan" username="yaosiyuan"/>  </tomcat-users> |

## Tomcat优化

### 基本思路

* 尽量缩短单个请求的处理时间
* 尽量可能多的并发处理请求
* 一定要做到能横向扩展

### Tomcat的基本优化

合理分配Tomcat需要的内存，设置**catalina.sh**中的**JAVA\_OPTS**

* **-server**：启用JDK的Server版
* **-Xms**:虚拟机初始化时的最小内存
* **-Xmx**:虚拟机可使用的最大内存（建议到物理内存的80%）  
  -**-XX:PermSize**：持久代初始值
* **-XX:MaxPermSize**：持久代最大内存(默认是32M)
* **-XX:MaxNewSize**：新生代内存的最大内存(默认是16M)

一般设置-Xms，-Xmx相等以避免在每次GC后调整堆的大小，因为默认空余堆内存小于40%时，JVM就会增大堆直到-Xmx的最大限制；空余堆内存大于70%时，JVM会减少堆直到-Xms的最小限制  
查看配置是否生效： jmap -heap tomcat的进程号

### Tomcat本身的配置优化

在server.xml中**<Connector>**中配置

**maxConnections**:最大连接数，对BIO模式，默认等于maxThreads；对NIO默认10000，对APR/native默认8192

**maxThreads**:最大线程数，即同时处理的任务个数，默认值是200

**acceptCount**：当处理任务的线程数达到最大时，接受排队的请求个数，默认是100

**minSpareThreads**:最小空闲线程数，默认10

**compression**：设置是否开启GZip压缩

**compressableMimeType**:那些类型需要压缩，默认是text/html,text/xml,text/plain

**compressionMinSize**:启用压缩的输出内容大小，默认是2048

**enableLookups**:是否反差域名，为了提高处理能力，应设置false

**connectionTimeout**:网络连接超时，单位毫秒，设置为-1表示永久不超时，通常可设置为2000毫秒

如果要加大并发连接数，应同时加大maxThreads和acceptCount，可以两个设置一样  
WebServer允许的最大连接数还受制于操作系统的内核参数设置，可通过ulimit -a查看  
如果配置了<Executor>，在<Connector>中通过executor属性指定参照<Executor>，那么<Connector>中关于线程的配置失效，以<Executor>中配置为准

### Tomcat三种模式

#### BIO

BIO是最稳定最老的一个连接器，是采用阻塞的方式，以为着每个连接线程绑定到每个Http请求，直到获得Http响应返回，如果Http客户端请求的是keep-Alive连接，那么这些连接也许一直保持着知道到达timeout时间，此期间不能用于其他请求。

#### NIO

NIO是使用Java的异步IO技术，不做阻塞，要使用的话，直接修改server.xml里的Connector节点，修改protocol为：protocol="org.apache.coyote.http11.Http11NioProtocol"

#### APR

APR是使用原生C语言编写的非阻塞I/O,但是需要安装apr和native，直接启动就支持apr，能大幅度提升性能。使用时指定protocol为protocol="org.apache.coyote.http11.Http11AprProtocol".