mini Tomcat，朋友，要不要试试？

2015-12-15 Tomcat那些事儿

前面几篇文章，我们基本都是在分析Tomcat内部的一些组件，实现原理等。而一般的Web应用，都需要在开发完成后以目录或者War包的形式，部署到Tomcat中，查看运行结果。
这样其实也没有什么不好，但在有些场景下，我们可能需要特殊的实现形式。

比如：

- **在Tomcat中做了一些特定的配置，比如端口，比如工作目录，甚至通道的传输编码等**。这个时候，如果我们直接将应用发给用户时，由于编码等一些原因可能导致演示等结果不理想。

- 在做一些Demo演示，甚至开源的项目时，经常会提供一些showcase等quickstart的功能。对于有些习惯了**开箱即用**的用户，再去单独下载Tomcat，再按照用户手册配置，就稍显冗余且复杂。

其实，Tomcat一直对外提供一种可以嵌入(Embeded)使用的mini Tomcat。
在Tomcat7开始，这种实现形式更加容易。同时，还提供了maven的插件，直接以一条maven的命令就可以完成应用的部署，启动，再不需要单独的配置等，满足用户的开箱即用。

用户所需要做的，只是在maven项目的pom.xml中，增加如下配置：

```
<plugin>
    <groupId>org.apache.tomcat.maven</groupId>
    <artifactId>tomcat7-maven-plugin</artifactId>
    <version>2.2</version>
</plugin>
```


之后，执行命令**mvn tomcat7:run** 即可实现应用的启动部署。打开浏览器即可查看。

是不是很容易？

这一maven plugin实现的基础，正是我们上面提到的mini Tomcat，官方称之为Embeded Tomcat。
完整的Tomcat内部，在启动的时候，需要解析其server.xml文件，获取内部配置的各类组件等，而plugin中使用了一系列预定义的配置项，**如果需要修改，可以通过命令行参数或者maven中增加配置项来修改**。

tomcat内部，是通过哪种方式实现的呢？
通过堆栈调用，我们来看下：

在mvn tomcat7:run命令执行时，是从maven的插件Mojo来执行，引导到Tomcat内部。从上面的调用链看，在AbstractRunMojo之后，调用了**org.apache.catalina.startup.Tomcat**这个类。一切所谓的嵌入式使用Tomcat，都是以此类为基础实现。

此类的javadoc对其功能精确概括如下：

> **Minimal tomcat starter for embedding/unit tests.** Tomcat supports multiple styles of configuration and startup - the most common and stable is server.xml-based, implemented in org.apache.catalina.startup.Bootstrap. **This class is for use in apps that embed tomcat.**

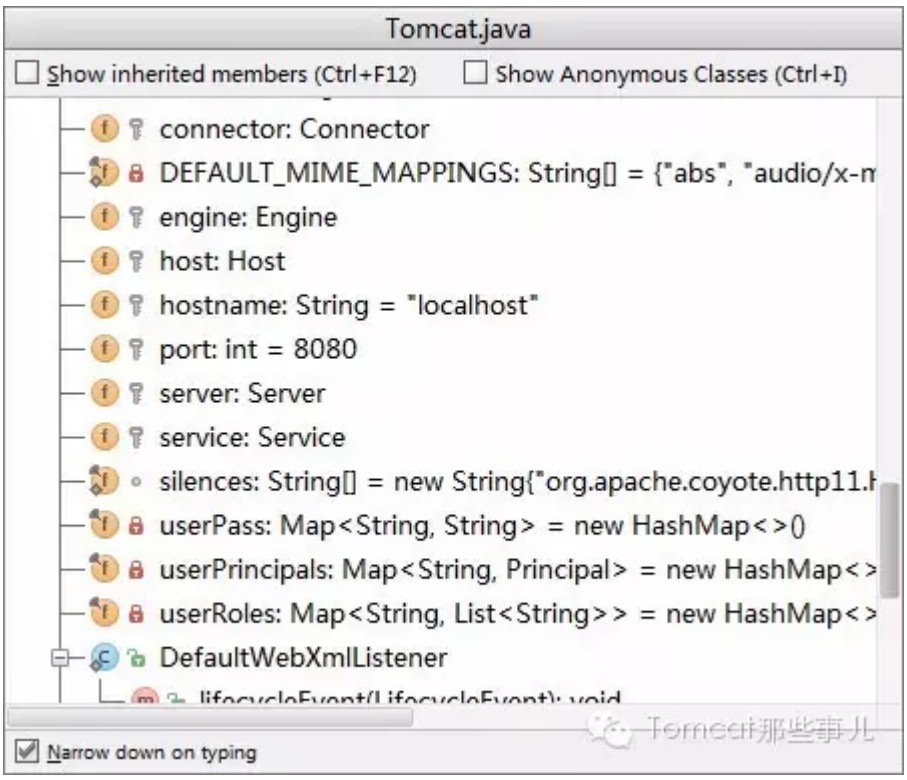
看下此类的内部
从AbstractRunMojo的startContainer方法内部，包含对其调用

```
Tomcat embeddedTomcat = new ExtendedTomcat( configurationDir );

embeddedTomcat.setBaseDir( configurationDir.getAbsolutePath() );
MemoryRealm memoryRealm = new MemoryRealm();

if ( tomcatUsers != null )
{
    if ( !tomcatUsers.exists() )
    {
        throw new MojoExecutionException( " tomcatUsers " + tomcatUs
    }
    getLog().info( "use tomcat-users.xml from " + tomcatUsers.getAbs
    memoryRealm.setPathname( tomcatUsers.getAbsolutePath() );
}
```

Tomcat类内部预设置的一些值



maven命令执行时，会将命令执行的当前目录做为tomcat的工作目录，代码是下面这个样子：

```
System.setProperty( "catalina.base", configurationDir.getAbsolutePa
```

了解catalina.base，可以后台回复**001**了解详情。

同时，此类在对于Tomcat的单元测试中也大量使用，看下面的代码：

```
@Test
public void testProgrammatic() throws Exception {
    Tomcat tomcat = getTomcatInstance();

    // No file system docBase required
    org.apache.catalina.Context ctx = tomcat.addContext("", null);

    Tomcat.addServlet(ctx, "myServlet", new HelloWorld());
    ctx.addServletMapping("/", "myServlet");

    tomcat.start();

    ByteChunk res = getUrl("http://localhost:" + getPort() + "/");
    assertEquals("Hello world", res.toString());
}
```


Tomcat类内提供的大量方便快速操作容器的方法，可以高效完成测试。

对于tomcat maven plugin的源码，官网默认下载的源码中并不包含。如果有朋友需要，可以在此下载
http://svn.apache.org/repos/asf/tomcat/maven-plugin/
下载时需要svn。

了解了mini Tomcat，以后的开发中，可以快速在maven中使用它，来提高你的工作效率吧。
如果感觉有用，请转发到朋友圈，让更多朋友看到。

扫描或长按下方二维码，关注我，一起交流。