我们上一讲，已经说了，单元测试这一块的内容

一般来说，我们在公司里面开发，都是一个常规的路径，写代码，写单元测试，运行单元测试，看一看单测覆盖率，覆盖率不达标，得继续写单元测试，覆盖大部分的代码

但是对于这个web工程来说，除了在本地单元测试以外，你还需要在本地启动web服务，然后调用一下web接口，简单的测试一下，做一个简单的冒烟测试，看看基本的功能是否正常

冒烟测试，微软那边传出来，开发工程师，不能完全依赖于测试工程师去帮你检测代码的质量，至少说你在把代码交付给测试工程师去检测之前，你需要用抽一根烟的时间，简单对你的代码核心和基础的功能，跑一下，看看，基本可以跑通。一般来说，测试工程师，吐槽最多的就是，开发工程师自己看都没看，写完代码就直接给我提测了。

加入jetty插件的配置

<plugin>

<groupId>org.mortbay.jetty</groupId>

<artifactId>jetty-maven-plugin</artifactId>

<version>7.1.6.v20100715</version>

<configuration>

<scanIntervalSeconds>10</scanIntervalSeconds>

<webAppConfig>

<contextPath>/test</contextPath>

</webAppConfig>

</configuration>

</plugin>

scanIntervalSeconds，指的是自动扫描项目代码变更的时间

contextPath，就是部署之后的contextPath

jetty插件的groupId不是默认的org.apache.maven.plugins，因此如果要执行该插件，需要在settings.xml中进行配置：

<pluginGroups>

<pluginGroup>org.mortbay.jetty</pluginGroup>

</pluginGroups>

接着mvn jetty:run，执行jetty插件的run goal，会用jetty容器启动web应用，默认绑定8080端口

用mvn jetty:run -Djetty.port=8081可以修改端口号

settings.xml文件里，有一个东西叫做pluginGroups

就是我们一般如果配置了插件以后，然后如果我们就直接在命令行运行jetty:run，cobertura:cobertura命令，其实用的是插件的缩写，jetty插件，cobertura插件。但是如果要将插件的缩写和插件自己的groupId对应起来的话，有的时候还是需要你去手动做一下配置的，不然在某些环境下可能直接运行类似jetty:run的命令可能会有问题。

但是有些插件，比如说dependency:tree，他的groupId，是org.maven.plugins，这个是默认的maven内核中的插件，所以不需要额外配置，但是对于我们上面说的两个插件，最好是配置一下。853769620