

Maven3 实战笔记 12Maven 构建 Web 应用

刘岩

Email:suhuanzheng7784877@163.com

1. 构建 Web 应用介绍

Java Web 项目是我们最熟悉的项目类型了，一个 Web 项目的结构大家也是烂熟于心。

之前咱们使用 Maven 构建的都是 JavaSE 类型的项目，虽然说 Java Web 项目可以借助类似于 Eclipse 这种 IDE 可以很好的帮助我们打包生成 war，而且 IDE 可以集成 Web 服务器，联调也不麻烦。那如果说我在构建 Web 项目的同时又想跑一下全部的单元测试用例呢，又比如我的 Web 项目引用了很多已有的 Maven 私服上的 jar 包呢（比如就是 Struts+Spring+Hibernate），再比如，我想看看此次 Web 项目构建过程中具体的报告信息等等，还是用 Maven 管理 Java Web 项目更为科学吧~当然了，笔者在这里绝不是让大家将所有的项目都迁移到 Maven 模式管理下，绝无此意，而是觉得在 JavaSE 上能体现 Maven 的优点同样适用于 Java Web 项目，虽然这种迁移会让您稍微改变一下以往的开发模式，但是个人认为从长期角度讲，对项目，对企业，对客户还是值得的。

2. 使用 Maven 构建 Java Web 项目

Java Web 项目和别的项目不一样的地方主要在于 2 点，第一就是 pom.xml 文件中的构件类型，第二就是 web 资源（包括页面、web.xml、图片资源文件）。

在 pom.xml 文件中需要指定打包类型为 war。pom.xml 文件内容如下

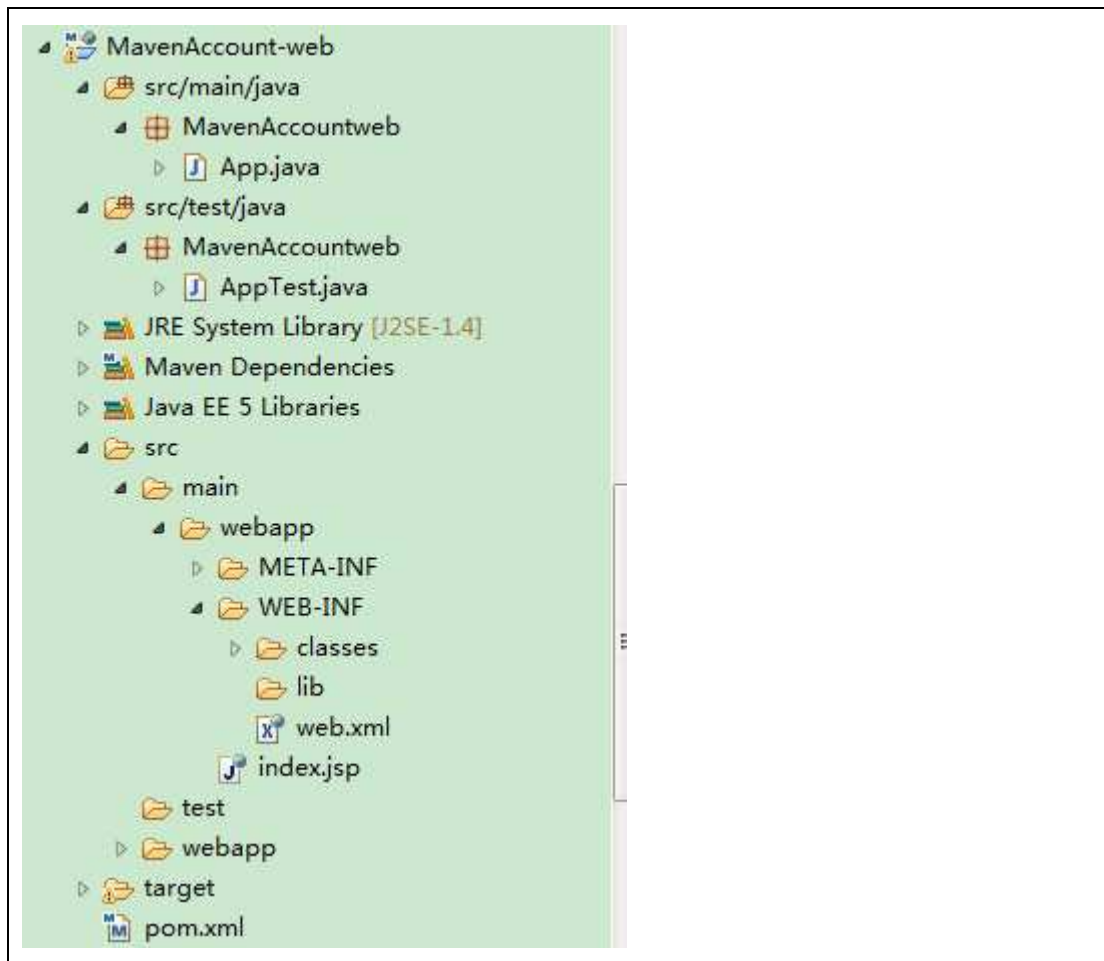
```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>MavenAccount-web</groupId>
  <artifactId>MavenAccount-web</artifactId>
  <version>0.0.1-SNAPSHOT</version>
```

```
<packaging>war</packaging>
<name>MavenAccount-web</name>
<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
</dependencies>
</project>
```

之后就是放置 web 文件的位置，Maven 约定 web 文件放到 src/main/webapp 下面，用过 MyEclipse 的朋友们可能知道，开发 web 项目根目录下除了有 src 目录，都有一个叫做 WebRoot 的目录用于存储 web 资源的。



webapp 下面就是那个放置页面资源的文件夹。

下面就是 src/main/java 下一个简单的 JavaBean 的代码。

```
package MavenAccountweb;

/**
 * Hello world!
 */

public class App {

    public String sayWebHello(String str) {
```

```
        System.out.println(str);

        return "hello:" + str;

    }

}
```

主页面入口 index.jsp 内容如下

```
<%@ page language="java" import="MavenAccountweb.App"

    pageEncoding="utf-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

    <head>

        <title>Hello Web</title>

        <!--

        <link rel="stylesheet" type="text/css" href="styles.css">

        -->

    </head>

    <body>

        <%

            App app = new App();

            String newString = app.sayWebHello("liuyan");

        %>

    </body>

</html>
```

```
%>

[<%=newString%>]

<br>

</body>

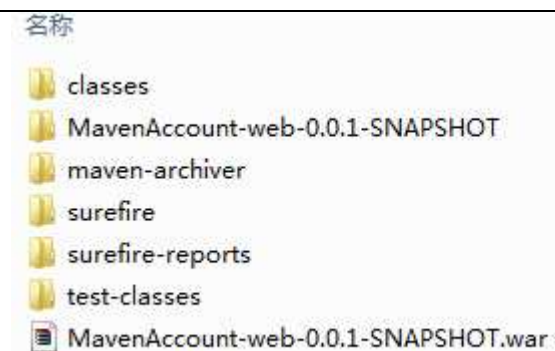
</html>
```

它引用了业务 JavaBean 输出一段内容。

在项目根目录下运行 maven 脚本

```
mvn clean package
```

之后看到目标输出目录出现如下内容

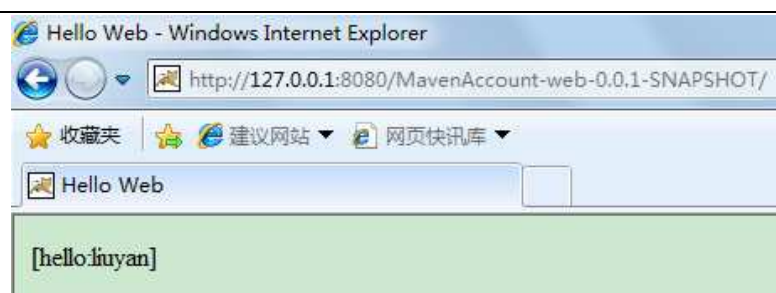


名称

- classes
- MavenAccount-web-0.0.1-SNAPSHOT
- maven-archiver
- surefire
- surefire-reports
- test-classes
- MavenAccount-web-0.0.1-SNAPSHOT.war

其中 MavenAccount-web-0.0.1-SNAPSHOT.war，我们将其部署到本机的 tomcat 上运

行。看到页面效果如下



Ok，证明你的 Java Web 项目构建成功。因为 Java Web 具体技术研发不在本次讨论的重点范围内，因此仅仅给出一个十分微小的 Web 项目。需要注意的是大家平时调试的时候完全可以使用 IDE 集成 Web 应用服务器，这是没有问题的，笔者看有些地方介绍了

Maven 插件——jetty-maven-plugin 进行嵌入式的 web 调试，其实笔者认为大可不必，有 IDE 的时候其实使用此 Maven 插件感觉并不比使用 IDE 集成的 Web 服务器方便多少，而且各种 Web 服务器实现 JSP、Servlet、JSF 规范是不一样的，Tomcat 自己的实现与 Jetty 就有很大差异，没准一个 JSF 项目在 Tomcat 下面运行得好好的，放到了 Jetty 下面就出现了不可预知的异常。所以还是用 IDE 集成的 Web 服务器查看问题更方便一些。当然此处仅仅是笔者经验的一家之言，不同项目因情况而异。

在 Web 项目中依赖 Spring、Struts 等等第三方的包，和 JavaSE 的依赖配置一样。此处不必详说。而因为笔者调试阶段是使用 IDE 工具集成的 Web 服务器——Tomcat，所以不必在 pom.xml 中显示声明 JSP-impl、Servlet-impl 等等 Java Web 规范的依赖项，Tomcat 自身就是一个 Java Web 容器，自己有相关 jar 实现了此规范。(至于 JavaEE 规范详细请查看笔者 Blog: <http://suhuanzheng7784877.iteye.com/blog/908380>)

3. 总结

Web 项目是我们搞 Java 研发人员中接触最多的，使用 Maven 配合 MyEclipse 这种 IDE 能让 Web 项目构建、升级第三方包、依赖第三方包更加简单、方便。也节省了很多查找 Web 依赖的时间。尤其是 SSH 项目，有时使用 Spring 需要依赖其他很多 jar 包，Maven 会为您管理、下载这种间接依赖的。从此我们的 Web 开发显得构建简单了。当然可以参考其他文章看看使用 Maven 插件进行 web 项目部署

<http://www.huomo.cn/developer/article-196fd.html>

http://article.yeyan.org/view/jdonee/19631?from_com

此处就不细细道来了。