

👍

21

🔖

💬

📺

💬

👤

🔍

📝 写博客

💬 发Chat

👤 登录 (https://passport.csdn.net/account/login)

👤 注册 (https://passport.csdn.net/account/mobileregister?action=mobileRegister)

(//so.csdn.net/so/np.blog.csdn.net/posted/new/gitchat/activity?utm_source=csdn-douban)

Maven多项目依赖配置,多maven项目聚合的实例

转载

2017年01月17日 20:38:27

标签：maven (http://so.csdn.net/so/search/s.do?q=maven&t=blog) /

架构 (http://so.csdn.net/so/search/s.do?q=架构&t=blog)

📖 5558

本文介绍一个多maven项目的实例demo，展示了聚合、继承、工程依赖、单元测试、多war聚合、cargo发布等场景

一、工程介绍

该项目由5个maven项目组成

- ▶  task-aggregator [task-aggregator]
- ▶  task-common [task-common]
- ▶  task-sla [task-sla]
- ▶  task-sla-web [task-sla-web]
- ▶  task-web-dist [task-web-dist]

task-aggregator是父工程，同时承担聚合模块和父模块的作用，没有实际代码和资源文件

task-common是基础工程，里面是公共的代码

task-sla是某一个业务子模块，不包含web内容

task-sla-web是某一个web子模块

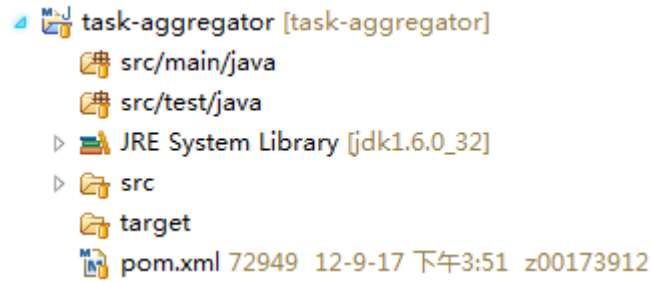
task-web-dist是最外围的web工程，聚合多个web工程，形成最终的war包

依赖关系是：task-common <-- task-sla <-- task-sla-web <-- task-web-dist

二、task-aggregator



21



这个工程是起到聚合作用，并充当parent pom，所以没有任何实际代码和资源文件。我这里选择了平行结构，另外一种方式是树形结构，我个人感觉平行结构看起来更舒服一点

下面是pom，有所简化：

Xml代码

```
01. <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
02.     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
03.     4.0.0.xsd">
04.     <!-- 定义公共变量 -->
05.     <properties>
06.         <spring.version>3.1.0.RELEASE</spring.version>
07.         <struts2.version>2.3.1</struts2.version>
08.         <hibernate.version>3.2.7.ga</hibernate.version>
09.     </properties>
10.
11.     <modelVersion>4.0.0</modelVersion>
12.     <groupId>com.xxx.task</groupId>
13.     <artifactId>task-aggregator</artifactId>
14.     <version>0.0.1-SNAPSHOT</version>
15.     <packaging>pom</packaging>
16.
17.     <!-- 待聚合模块 -->
18.     <modules>
19.         <module>../task-common</module>
20.         <module>../task-sla</module>
21.         <module>../task-sla-web</module>
22.         <module>../task-web-dist</module>
23.     </modules>
24.
25.     <!-- 配置部署的远程仓库 -->
26.     <distributionManagement>
27.         <snapshotRepository>
28.             <id>nexus-snapshots</id>
29.             <name>nexus distribution snapshot repository</name>
30.             <url>http://10.78.68.122:9090/nexus-2.1.1/content/repositories/snapshots</url>
31.         </snapshotRepository>
32.     </distributionManagement>
33.
34.     <build>
```



21



```
35.
36.     <pluginManagement>
37.         <plugins>
38.
39.             <plugin>
40.                 <groupId>org.apache.maven.plugins</groupId>
41.                 <artifactId>maven-resources-plugin</artifactId>
42.                 <version>2.6</version>
43.                 <configuration>
44.                     <encoding>UTF-8</encoding>
45.                 </configuration>
46.             </plugin>
47.
48.             <plugin>
49.                 <groupId>org.apache.maven.plugins</groupId>
50.                 <artifactId>maven-compiler-plugin</artifactId>
51.                 <version>2.5.1</version>
52.                 <configuration>
53.                     <encoding>UTF-8</encoding>
54.                 </configuration>
55.             </plugin>
56.
57.         </plugins>
58.     </pluginManagement>
59.
60. </build>
61.
62. <dependencyManagement>
63.
64.     <dependencies>
65.
66.         <dependency>
67.             <groupId>com.sun</groupId>
68.             <artifactId>tools</artifactId>
69.             <version>1.6.0</version>
70.             <scope>system</scope>
71.             <systemPath>${env.JAVA_HOME}/lib/tools.jar</systemPath>
72.         </dependency>
73.
74.     </dependencies>
75.
76. </dependencyManagement>
77.
78. </project>
```

基本上是一目了然，只是有几点注意下：

- 1、这里配置了<distributionManagement>，这样子项目就不需要重复配置了
- 2、通过<pluginManagement>，对一些插件进行了公共的配置，这里主要是为了消除构建时的告警



21

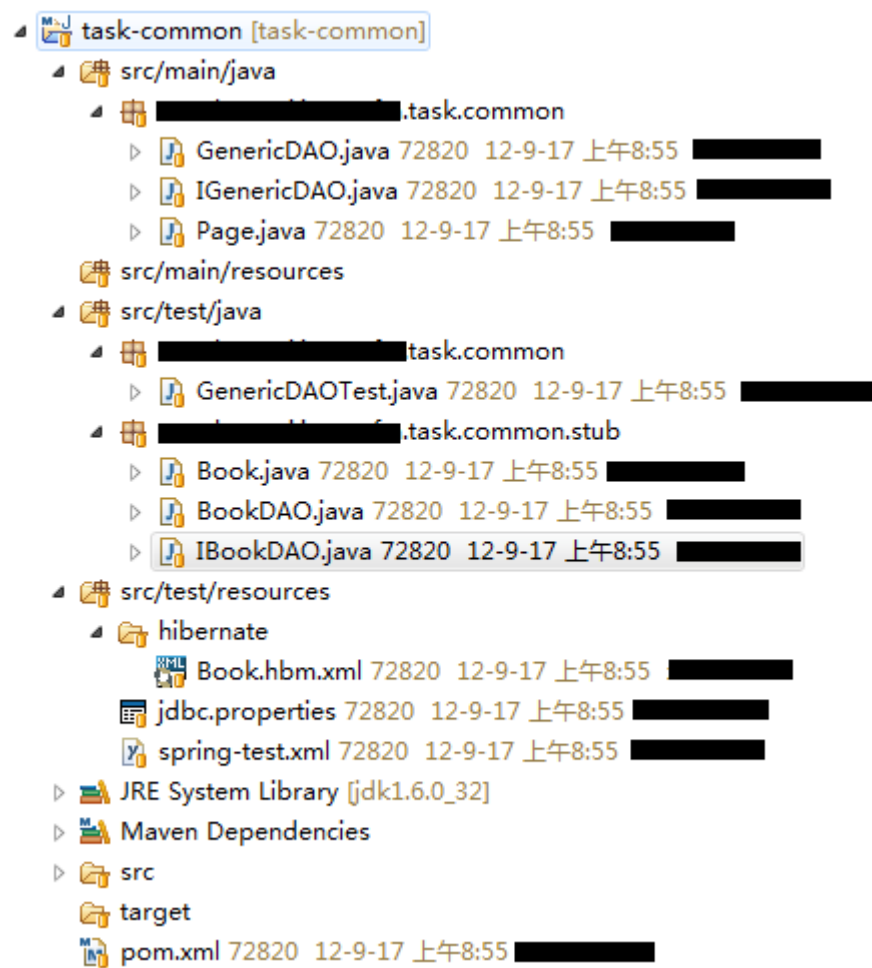


3、配置tools，是因为实际中发现，其他开发人员从svn上check out工程以后，有的人会报错，找不到tools.jar，这样配置以后就好了

三、task-common

该工程是公共工程，提供了项目中的公共代码，这里只包括了通用的DAO组件，作为示例。

该工程不依赖任何其他工程



该工程里有几点要点：

1、在代码内部用了Spring的注解

```
Java代码
01. public abstract class GenericDAO<T> implements IGenericDAO<T> {
02.
03.     private Class<T> entityClass;
04.
05.     public GenericDAO(Class<T> clazz) {
```



21



```
06.         this.entityClass = clazz;
07.     }
08.
09.     @Autowired
10.     private HibernateTemplate hibernateTemplate;
11.
12. }
```

这里用到了@Autowired注解，所以最终形成的war包，必须在spring配置文件中声明HibernateTemplate类型的bean，否则会报错

我这里用的maven环境是maven3.0.4，这个版本打出的jar包，带有Directory Entries信息，所以spring的注解即使在jar包中也可生效，如果是比较老的版本，spring的注解在jar包中不好用，关于这个问题的详细描述，见另外一篇博客：

<http://kyfxbl.iteye.com/blog/1675368>

2、单元测试的写法

Java代码

```
01. @RunWith(SpringJUnit4ClassRunner.class)
02. @ContextConfiguration(locations = "classpath:spring-test.xml")
03. @Transactional
04. public class GenericDAOTest {
05.
06.     @Autowired
07.     private IBookDAO bookDAO;
08.
09.     @Test
10.     public void testInsert() {
11.         Book book = new Book();
12.         book.setName("thinking in java");
13.         book.setIsbn("111");
14.         bookDAO.insert(book);
15.     }
16.
17. }
```

这里用到了几个注解，@RunWith是为了在spring容器环境下跑这个单元测试类，以支持依赖注入。

@ContextConfiguration是声明spring配置文件的位置。@Transactional注解之后，在单元测试方法中的事务会自动回滚，这个比较方便，这样在前面执行的方法，不会对后面的方法造成影响

这个单元测试类，可以直接在maven里跑起来，让我比较惊喜。之前这样写，在ant里跑没有成功，可能是我没有找到合适的插件的原因

3、除了测试的java代码之外，还有3个资源文件，都是放在src/test/resources下，这些资源文件只在test阶段生效，



21



package阶段不会被打包，也就是专门供测试阶段使用

这个各有利弊，优点是测试的配置文件与开发的配置文件隔离，互不干扰。缺点是配置文件似乎缺少了集中放置的地点，这样如果多个maven工程都需要跑单元测试，要共享测试用配置文件，比较麻烦一点

不过从我个人来看，也算是利大于弊。只是在每个maven项目下，都需要独立的测试相关资源文件，其实也有利于分别修改

另外，可以看到这里的hibernate映射文件，不是和model类放在一个package下，而是放在resources目录下的，这样做可以避免一些潜在的问题，也有利于后续的聚合

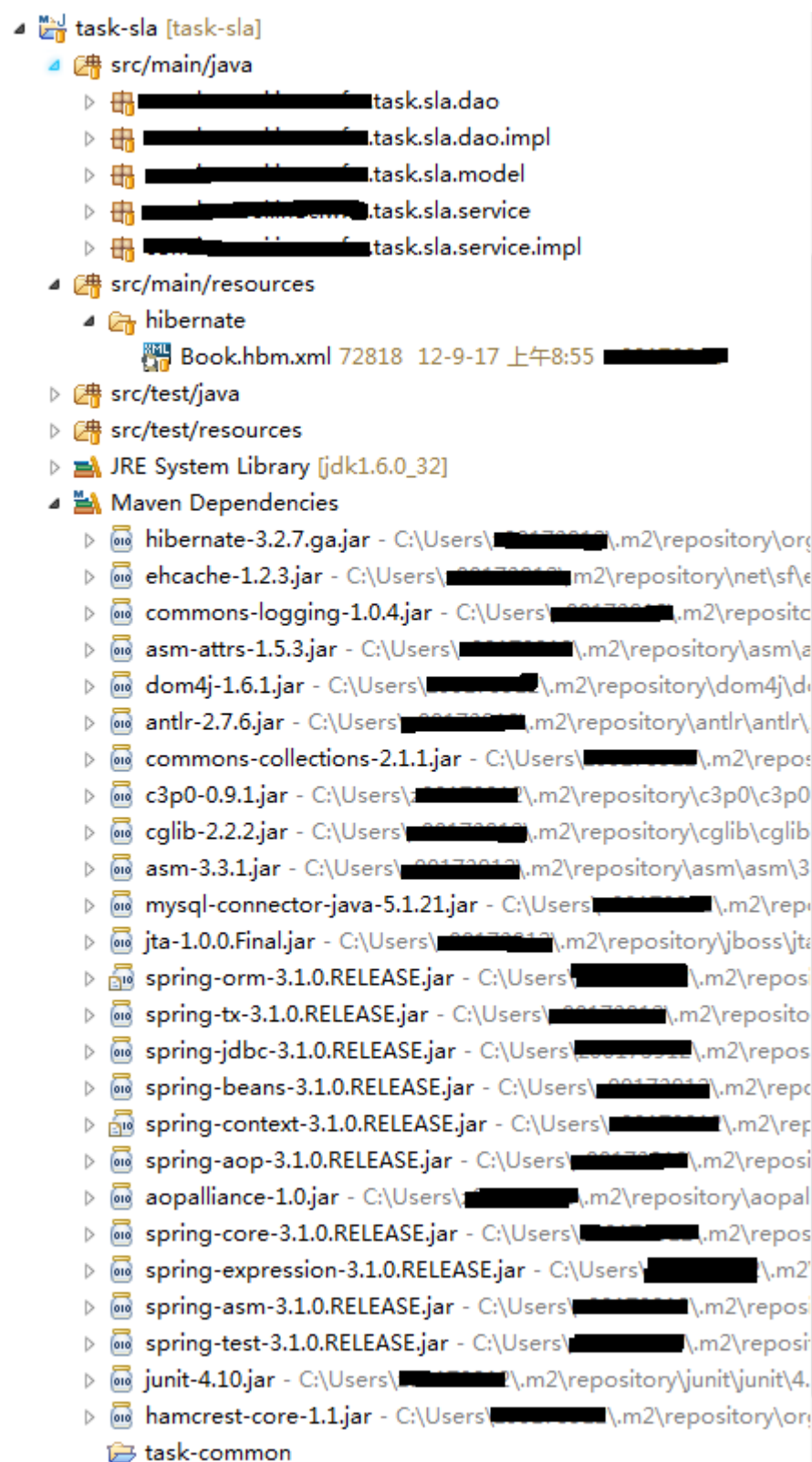
4、pom文件没有什么特别的，只是要引入<scope>为test的junit和spring-test

四、task-sla

该工程依赖task-common（因为用到了GenericDAO），是某一个业务模块的逻辑部分，包含了数据库访问层和业务逻辑层，但是不包括web相关的部分



21



这里没有什么特别要注意的，目录结构和task-common基本一样。比较特别的是可以看到Maven Dependencies里，有一个task-common工程，所以task-common里的任何修改，都可以第一时间在这个工程里体现出来，是比较方便的

关于这个问题，见另外一篇博客：<http://kyfxbl.iteye.com/blog/1679806>

另外就是前面说过的，hibernate的映射文件，应该放在src/main/resources下，而不是与Model类放在一起

五、task-sla-web

这个工程是上述task-sla工程的web层，依赖于task-sla，由于task-sla又依赖task-common，所以这个工程最终会同时依赖task-common和task-sla

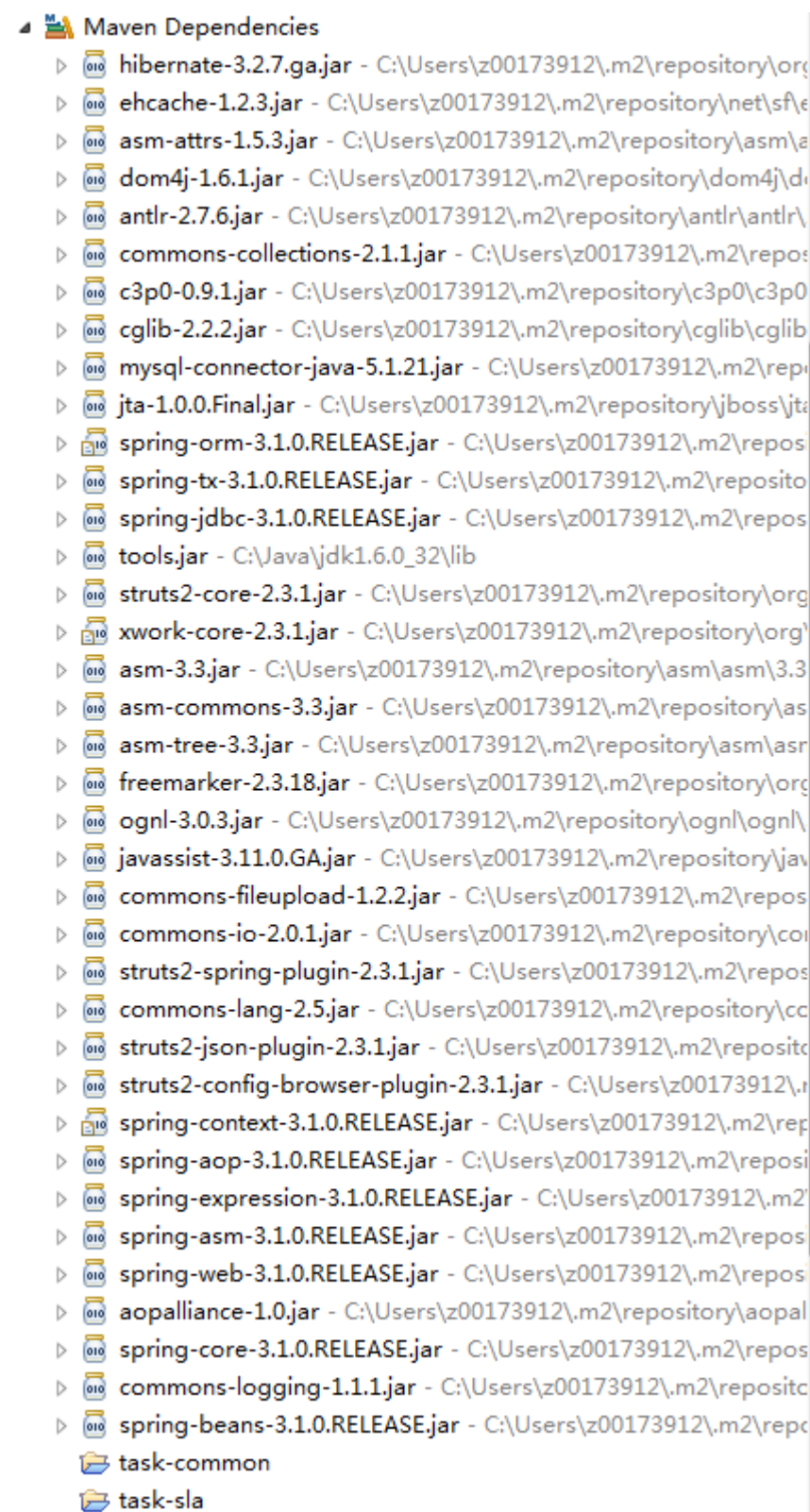


21





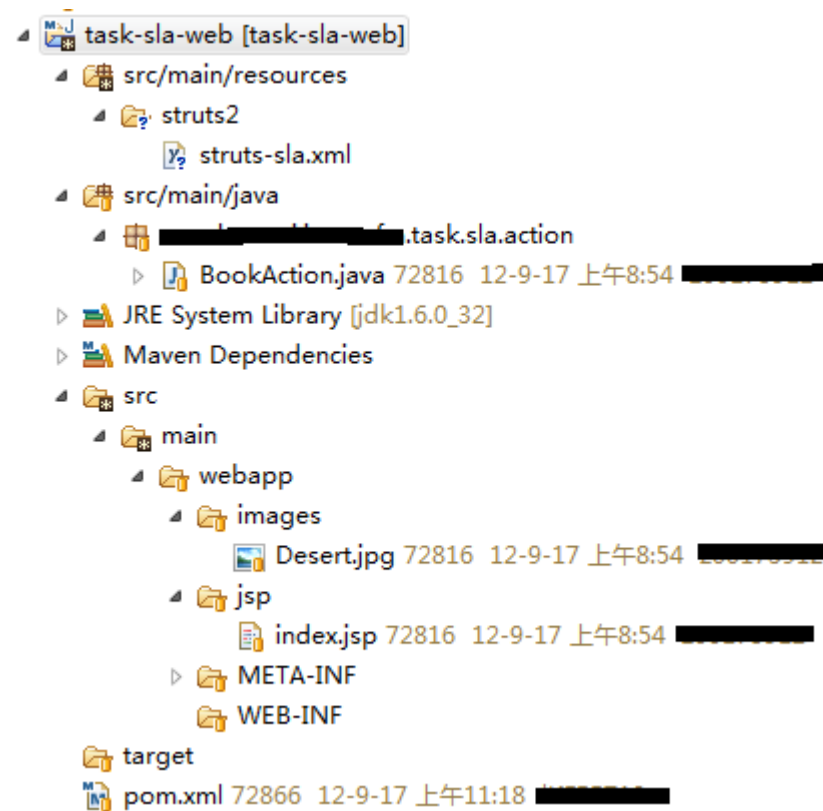
21



然后这个工程里包含了web层的东西，包括Action类、jsp、图片、struts2的配置文件等，这些东西放在web工程里是最合适的



21



这里需要注意2点：

1、这个工程的packaging类型是war，而不是jar。但是最终它不会独立打出war包来，其src/main/webapp里的所有文件，都会被最外围的task-web-dist工程聚合成一个总的war

2、这个工程的WEB-INF目录下，没有web.xml（有也没用，最终会被覆盖）。默认情况下，packaging类型为war的项目，如果没有web.xml，则构建会失败，因此需要在pom里做一个配置

该项目的pom如下，省略了依赖部分：

```
Xml代码
01. <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
02.     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
03.
04.     <parent>
05.         <groupId>com.xxx.task</groupId>
06.         <artifactId>task-aggregator</artifactId>
07.         <version>0.0.1-SNAPSHOT</version>
08.         <relativePath>../task-aggregator</relativePath>
09.     </parent>
10.
11.     <modelVersion>4.0.0</modelVersion>
12.     <artifactId>task-sla-web</artifactId>
13.     <packaging>war</packaging>
```



21

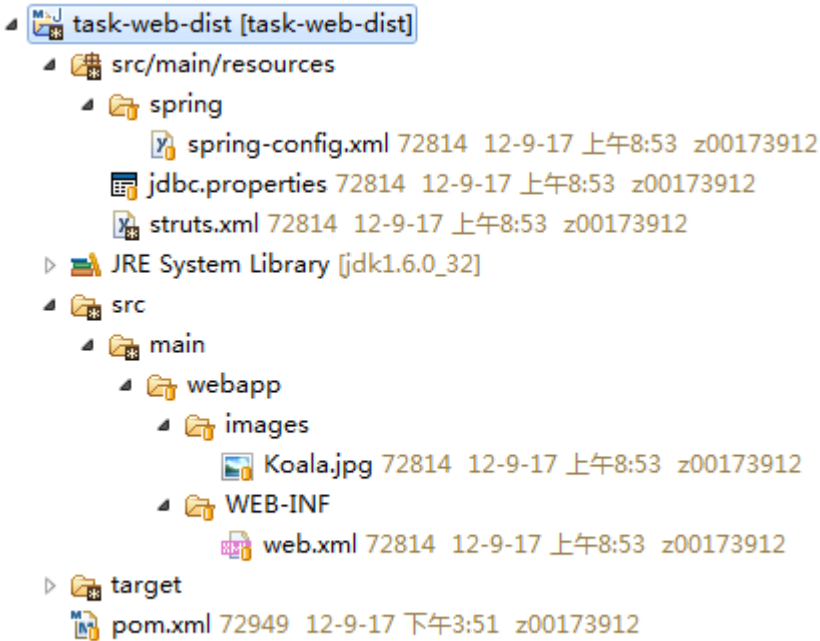


```
14.
15.     <build>
16.         <plugins>
17.             <plugin>
18.                 <groupId>org.apache.maven.plugins</groupId>
19.                 <artifactId>maven-war-plugin</artifactId>
20.                 <configuration>
21.                     <failOnMissingWebXml>false</failOnMissingWebXml>
22.                 </configuration>
23.             </plugin>
24.         </plugins>
25.     </build>
26.
27.     <!-- 配置依赖 -->
28.     <dependencies>
29.         <dependency>
30.             <groupId>org.springframework</groupId>
31.             <artifactId>spring-beans</artifactId>
32.         </dependency>
33.
34.     </dependencies>
35.
36. </project>
```

上面的<failOnMissingWebXml>，就是配置缺少web.xml也不使构建失败

六、task-web-dist

这个工程是最外围的web工程，起到聚合的作用，即把所有的web项目，打成最终的war包。同时，在这个工程里，放置里公共的配置文件，比如struts.xml、ssoconfig.properties等



这个工程的聚合意图十分明显，比如struts.xml

Xml代码

```
01. <?xml version="1.0" encoding="UTF-8"?>
02. <!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN" "ht
03. tp://struts.apache.org/dtds/struts-2.3.dtd">
04. <struts>
05.
06.     <constant name="struts.objectFactory" value="spring" />
07.     <constant name="struts.ui.theme" value="simple" />
08.     <constant name="struts.i18n.encoding" value="UTF-8" />
09.     <constant name="struts.action.extension" value="action" />
10.     <constant name="struts.enable.DynamicMethodInvocation" value="false" />
11.     <constant name="struts.devMode" value="true" />
12.
13.     <include file="struts2/struts-sla.xml" />
14.
15. </struts>
```

提供了项目通用的配置，并把各子项目的struts2配置文件聚合起来。war包中的web.xml也是在这里提供的

下面是该工程的pom，也省略了依赖的配置：

Xml代码

```
01. <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```



21



```
02.      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_
03.      0.xsd">
04.      <parent>
05.        <groupId>com.xxx.task</groupId>
06.        <artifactId>task-aggregator</artifactId>
07.        <version>0.0.1-SNAPSHOT</version>
08.        <relativePath>../task-aggregator</relativePath>
09.      </parent>
10.
11.      <modelVersion>4.0.0</modelVersion>
12.      <artifactId>task-web-dist</artifactId>
13.      <packaging>war</packaging>
14.
15.      <build>
16.
17.        <finalName>task</finalName>
18.
19.        <plugins>
20.
21.          <!-- 合并多个war -->
22.          <plugin>
23.            <groupId>org.apache.maven.plugins</groupId>
24.            <artifactId>maven-war-plugin</artifactId>
25.            <configuration>
26.              <packagingExcludes>WEB-INF/web.xml</packagingExcludes>
27.              <overlays>
28.                <overlay>
29.                  <groupId>com.huawei.inoc.wfm.task</groupId>
30.                  <artifactId>task-sla-web</artifactId>
31.                </overlay>
32.              </overlays>
33.            </configuration>
34.          </plugin>
35.
36.          <!-- 利用cargo启动容器 -->
37.          <plugin>
38.            <groupId>org.codehaus.cargo</groupId>
39.            <artifactId>cargo-maven2-plugin</artifactId>
40.            <version>1.2.3</version>
41.            <configuration>
42.              <container>
43.                <containerId>tomcat7x</containerId>
44.                <home>D:\apache-tomcat-7.0.29</home>
45.              </container>
46.              <configuration>
47.                <type>standalone</type>
48.                <home>${project.build.directory}/tomcat7.0.29</home>
49.                <properties>
50.                  <cargo.jvmargs>
51.                    -Xdebug
52.                    -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=8787
53.                  </cargo.jvmargs>
```



21



```
54.         </properties>
55.     </configuration>
56. </configuration>
57. <executions>
58.     <execution>
59.         <id>cargo-run</id>
60.         <phase>pre-integration-test</phase>
61.         <goals>
62.             <goal>run</goal>
63.         </goals>
64.     </execution>
65. </executions>
66. </plugin>
67.
68. </plugins>
69.
70. </build>
71.
72. </project>
```

这里主要是对maven-war-plugin和cargo-maven2-plugin这2个插件进行了配置，以起到聚合war，以及通过cargo启动容器的作用

关于多war聚合，以及cargo，见另外2篇博客：<http://kyfxbl.iteye.com/blog/1678121>、
<http://kyfxbl.iteye.com/blog/1677608>

七、启动构建

在task-aggregator目录下，执行mvn clean deploy或者mvn clean install，就可启动整个构建过程，并将容器启动起来，跑最终生成的war包

```
[INFO] Building war: D:\workspace_helios\task-web-dist\target\task.war
[INFO] -----
[INFO] Reactor Summary:
[INFO]
[INFO] task-aggregator ..... SUCCESS [0.117s]
[INFO] task-common ..... SUCCESS [3.114s]
[INFO] task-sla ..... SUCCESS [2.117s]
[INFO] task-sla-web ..... SUCCESS [3.249s]
[INFO] task-web-dist ..... SUCCESS [6.633s]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 15.366s
[INFO] Finished at: Mon Sep 17 21:06:11 CST 2012
[INFO] Final Memory: 11M/123M
[INFO] -----
D:\workspace_helios\task-aggregator>
```



21



ggregator [task-aggregat
ommon [task-common]
a [task-sla]
a-web [task-sla-web]
eb-dist [task-web-dist]

(http://dl2.iteye.com/upload/attachment/0073/9522/c5d3d487-6277-

3c19-bf4f-d48b0fe3391b.png)
大小: 5.6 KB

gregator [task-aggregator]
main/java
test/java
System Library [jdk1.6.0_32]

(http://dl2.iteye.com/upload/attachment/0073/9525/28fcfab-0628-

et
s.xml 72949 12-9-17 下午3:51 z0
36ef-aaf2-ca87fb5b2f84.png)
大小: 7.5 KB

> spring-tx-3.1.0.RELEASE.jar - C:\Users\z00173912\m2repository
> spring-jdbc-3.1.0.RELEASE.jar - C:\Users\z00173912\m2repository
> tools.jar - C:\Java\jdk1.6.0_32\lib
> struts2-core-2.3.1.jar - C:\Users\z00173912\m2repository\org
> xwork-core-2.3.1.jar - C:\Users\z00173912\m2repository\org
> asm-3.3.jar - C:\Users\z00173912\m2repository\asm\asm\3.3
> asm-commons-3.3.jar - C:\Users\z00173912\m2repository\asm
> asm-tree-3.3.jar - C:\Users\z00173912\m2repository\asm\asm
> freemarker-2.3.18.jar - C:\Users\z00173912\m2repository\org
> ognl-3.0.3.jar - C:\Users\z00173912\m2repository\ognl\ognl
> javassist-3.11.0.GA.jar - C:\Users\z00173912\m2repository\jav
> commons-fileupload-1.2.2.jar - C:\Users\z00173912\m2repository\com
> commons-io-2.0.1.jar - C:\Users\z00173912\m2repository\com
> struts2-spring-plugin-2.3.1.jar - C:\Users\z00173912\m2repository\org
> commons-lang-2.5.jar - C:\Users\z00173912\m2repository\com
> struts2-json-plugin-2.3.1.jar - C:\Users\z00173912\m2repository

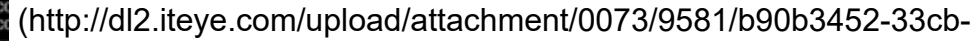
(http://dl2.iteye.com/upload/attachment/0073/9563/55bb6102-cd77-

3d23-9f8b-f66b78e0e4a9.png)
大小: 84.6 KB

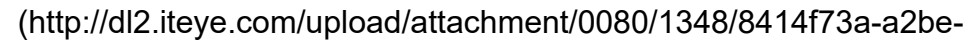
task-web-dist [task-web-dist]
src/main/resources
spring
spring-config.xml 72814 12-9-17 上午8:53 z00173912
jdbc.properties 72814 12-9-17 上午8:53 z00173912
struts.xml 72814 12-9-17 上午8:53 z00173912
JRE System Library [jdk1.6.0_32]
src
main
webapp
images
Koala.jpg 72814 12-9-17 上午8:53 z00173912
WEB-INF
web.xml 72814 12-9-17 上午8:53 z00173912
target
pom.xml 72949 12-9-17 下午3:51 z00173912

(http://dl2.iteye.com/upload/attachment/0073/9576/afe24b98-432a-

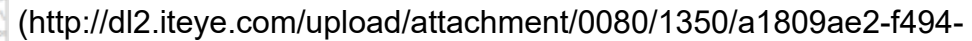
3713-89b5-680efa58f01e.png)
大小: 21.2 KB



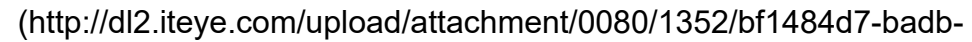
大小: 12.8 KB



大小: 33.5 KB



大小: 69.9 KB



32c3-8c79-4d9627a5788d.png)



21



最近在学习Maven，把一个开源的项目改成maven管理，期间使用到了多项目，从网上查阅了一些资料，主要参考的是<http://kyfxbl.iteye.com/blog/1680045> (<http://kyfxbl.iteye.com/blog/1680045>)，在此把自己的一些心得体会写出来，供大家学习交流。

关于maven的安装，在此就不进行阐述，请参考网上其他教程。

本实例由4个项目组成，其中，aggregator是父工程，同时承担聚合模块和父模块的作用，没有实际代码和资源文件；open-platform-common是公共的java工程；open-platfor-web是公共的web文件，主要包括css、js等；open-bug-m是最终要发布的应用，形成war包。

一、建立一个Maven工程：aggregator

/aggregator

 /src/main/java

 /src/test/java

 pom.xml

此工程主要是父模块，聚合其他工程，没有实际代码和资源文件，最主要的是pom.xml文件，其主要内容如下：

Xml代码 ☆

```
<modelVersion>4.0.0</modelVersion>
<groupId>cn.jess.platform</groupId>
<artifactId>aggregator</artifactId>
<version>0.0.1-SNAPSHOT</version>
<!-- 因为是父工程，因此此处的packaging必须为pom -->
<packaging>pom</packaging>
<name>aggregator</name>

<modules>
  <module>../open-platform-common</module>
  <module>../open-platform-web</module>
  <module>../open-bug-m</module>
</modules>

<!-- 配置部署的远程仓库 -->
<distributionManagement>
  <snapshotRepository>
    <id>nexus-snapshots</id>
    <name>nexus distribution snapshot repository</name>
    <url>http://127.0.0.1:8081/nexus/content/repositories/snapshots</url>
  </snapshotRepository>
</distributionManagement>
```



21



```
<build>
  <pluginManagement>
    <plugins>

      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-resources-plugin</artifactId>
        <version>2.6</version>
        <configuration>
          <encoding>UTF-8</encoding>
        </configuration>
      </plugin>

      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>2.5.1</version>
        <configuration>
          <encoding>UTF-8</encoding>
          <source>1.6</source>
          <target>1.6</target>
        </configuration>
      </plugin>

    </plugins>
  </pluginManagement>
</build>

<dependencyManagement>

  <dependencies>

    <dependency>
      <groupId>com.sun</groupId>
      <artifactId>tools</artifactId>
      <version>1.6.0</version>
      <scope>system</scope>
      <systemPath>${env.JAVA_HOME}/lib/tools.jar</systemPath>
    </dependency>

  </dependencies>

</dependencyManagement>
```

```
</dependencies>
```

```
</dependencyManagement>
```

二、建立一个Maven工程：open-platform-common

此工程主要是项目中使用到的公共java类库，pom文件主要内容如下：

Xml代码 ☆

```
<!-- 由于存在parent工程，因此groupId和version可以省略，直接使用parent工程-->
```

```
<modelVersion>4.0.0</modelVersion>
```

```
<artifactId>open-platform-common</artifactId>
```

```
<!-- 因为此工程要发布到webapp的lib目录下，因此为jar(不知道这样解释对否?) -->
```

```
<packaging>jar</packaging>
```

```
<properties>
```

```
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
```

```
</properties>
```

```
<!-- 指定Maven仓库 -->
```

```
<repositories>
```

```
<!-- my的maven仓库 -->
```

```
<repository>
```

```
<id>myRepository</id>
```

```
<name>local private nexus</name>
```

```
<url>http://127.0.0.1:8081/nexus/content/groups/public/</url>
```

```
<releases>
```

```
<enabled>>true</enabled>
```

```
</releases>
```

```
<snapshots>
```

```
<enabled>>true</enabled>
```

```
</snapshots>
```

```
</repository>
```

```
</repositories>
```

```
<!-- 指定maven plugin仓库 -->
```

```
<pluginRepositories>
```

```
<!-- oschina的maven plugin仓库 -->
```

```
<pluginRepository>
```

```
<id>myPluginRepository</id>
```

```
<name>local private nexus</name>
```

```
<url>http://127.0.0.1:8081/nexus/content/groups/public/</url>
```

```
<releases>
```

```
<enabled>>true</enabled>
```



21





21



```
        </releases>
        <snapshots>
            <enabled>false</enabled>
        </snapshots>
    </pluginRepository>
</pluginRepositories>
<dependencies>
    <!-- 此处的类库根据自己的需要进行添加 -->
</dependencies>
<!-- 用来指定父工程-->
<parent>
    <groupId>cn.jess.platform</groupId>
    <artifactId>aggregator</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <relativePath>../aggregator</relativePath>
</parent>
```

三、建立一个Maven工程：open-platform-web

此工程主要是项目中使用到的公共web文件，pom文件主要内容如下：

Xml代码 ☆

```
<!-- 由于存在parent工程，因此groupId和version可以省略，直接使用parent工程-->
<modelVersion>4.0.0</modelVersion>
<artifactId>open-platform-web</artifactId>
<!-- 因为此工程要发布到webapp应用的根目录下，因此为war(不知道这样解释对否?) -->
<packaging>war</packaging>
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
<!-- 指定Maven仓库 -->
<repositories>
    <!-- my的maven仓库 -->
    <repository>
        <id>myRepository</id>
        <name>local private nexus</name>
        <url>http://127.0.0.1:8081/nexus/content/groups/public/</url>
        <releases>
            <enabled>true</enabled>
        </releases>
        <snapshots>
            <enabled>true</enabled>
```



21



```
        </snapshots>
    </repository>
</repositories>
<!-- 指定maven plugin仓库 -->
<pluginRepositories>
    <!-- oschina的maven plugin仓库 -->
    <pluginRepository>
        <id>myPluginRepository</id>
        <name>local private nexus</name>
        <url>http://127.0.0.1:8081/nexus/content/groups/public/</url>
        <releases>
            <enabled>true</enabled>
        </releases>
        <snapshots>
            <enabled>false</enabled>
        </snapshots>
    </pluginRepository>
</pluginRepositories>
```

```
<parent>
    <groupId>cn.jess.platform</groupId>
    <artifactId>aggregator</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <relativePath>../aggregator</relativePath>
</parent>
</project>
```

注意：此工程的WEB-INF目录下必须包含web.xml文件，否则在执行mvn时会报错

四、建立一个Maven工程：open-bug-m：

此工程是最终要发布的应用，其依赖于open-platform-common和open-platform-web，因此在pom文件中要加入这两个工程的依赖，pom文件内容如下所示：

Xml代码 ☆

```
<groupId>open-bug-m</groupId>
<artifactId>open-bug-m</artifactId>
<packaging>war</packaging>
<name/>
<description/>
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
```



21



```
<parent>
  <groupId>cn.jess.platform</groupId>
  <artifactId>aggregator</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <relativePath>../aggregator</relativePath>
</parent>
<!-- 指定Maven仓库 -->
<repositories>
  <!-- my的maven仓库 -->
  <repository>
    <id>myRepository</id>
    <name>local private nexus</name>
    <url>http://127.0.0.1:8081/nexus/content/groups/public/</url>
    <releases>
      <enabled>true</enabled>
    </releases>
    <snapshots>
      <enabled>true</enabled>
    </snapshots>
  </repository>
</repositories>
<!-- 指定maven plugin仓库 -->
<pluginRepositories>
  <!-- oschina的maven plugin仓库 -->
  <pluginRepository>
    <id>myPluginRepository</id>
    <name>local private nexus</name>
    <url>http://127.0.0.1:8081/nexus/content/groups/public/</url>
    <releases>
      <enabled>true</enabled>
    </releases>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
  </pluginRepository>
</pluginRepositories>
<dependencies>
  <dependency>
    <groupId>cn.jess.platform</groupId>
    <artifactId>open-platform-common</artifactId>
```



21



```
<version>0.0.1-SNAPSHOT</version>
<type>jar</type>
</dependency>
<dependency>
  <groupId>cn.jess.platform</groupId>
  <artifactId>open-platform-web</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <type>war</type>
</dependency>
<!-- 此处的类库根据自己的需要进行添加 -->
```

```
</dependencies>
<build>
  <finalName>open-bug</finalName>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-war-plugin</artifactId>
      <version>2.3</version>
      <configuration>
        <packagingExcludes>WEB-INF/web.xml</packagingExcludes>
        <overlays>
          <overlay>
            <groupId>cn.jess.platform</groupId>
            <artifactId>open-platform-web</artifactId>
          </overlay>
        </overlays>
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.codehaus.cargo</groupId>
      <artifactId>cargo-maven2-plugin</artifactId>
      <version>1.2.3</version>
      <configuration>
        <container>
          <containerId>tomcat7x</containerId>
          <home>F:\apache-tomcat-7.0.42(x64)</home>
        </container>
      </configuration>
    </plugin>
  </plugins>
</build>
```



21



```
<type>existing</type>
<home>F:\apache-tomcat-7.0.42(x64)</home>
<properties>
  <cargo.jvmargs>
    -Xdebug          -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=8787
  </cargo.jvmargs>
</properties>
</configuration>
</configuration>
<executions>
  <execution>
    <id>cargo-run</id>
    <phase>pre-integration-test</phase>
    <goals>
      <goal>run</goal>
    </goals>
  </execution>
</executions>
</plugin>
</plugins>
</build>
```

关于maven-war-plugin和cargo-maven2-plugin的使用方法请参考网上其他使用教程。
所有上述四个工程准备就绪后，执行mvn install就可对工程项目进行部署。



21



Answer1331 (/Answer1331) 2017-08-15 13:31

回复

1楼

(/Answer1331)请问多个Web模块可以共享一些配置文件吗，比如log4j.properties

Maven POM文件、多模块以及依赖关系



EvelynHouseba

2014年02月20日 15:57



14377

一、POM.XML 1、ProjectObject Model：项目对象模型 2、基本项： project：pom.xml的顶级元素。groupId：指出创建这个工程的组织或团队的唯一标识。plug...

(<http://blog.csdn.net/EvelynHouseba/article/details/19558841>)

Maven学习八：多模块依赖



lfsfxy9

2013年11月30日 20:37



14850

使用Maven以后，可以更方便的进行构件化开发，一般项目中存在多模块，它们的关系，包括父子关系以及依赖关系，都可以通过pom文件的配置来实现的。1. 父子关系 一个多模块项目通过一个父POM...

(<http://blog.csdn.net/lfsfxy9/article/details/17046029>)

Maven的使用《二》之依赖、继承、聚合



Mr_li13

2016年04月18日 16:42



6697

问题？Maven的使用《二》之依赖、继承、聚合 前面已经讲到了环境的搭建(环境搭建链接)，这里就不重复了。下面的主要是对依赖、继承、聚合详细说明。这里用的编译软件是eclipse，要装maven...

(http://blog.csdn.net/Mr_li13/article/details/51178646)

3.将maven项目jar纳入maven仓库，Mave项目依赖另外一个Maven项目的案例

1 若想让maven项目依赖另外一个maven项目，被依赖的项目要在maven仓库中有相应的jar包，所以要对依赖的项目执行mvninstall命令。 2 新建第二个...



toto1297488504

2014年10月18日 09:43



1595

(<http://blog.csdn.net/toto1297488504/article/details/40208233>)

Maven--多模块依赖实例解析（五）



StubbornPotatoes

2013年09月30日 14:42



10620

这是最后一篇，做一个多模块依赖的正式例子，简单的讲一下单元测试、覆盖率以及发布和站点的部署。只想讲讲和Maven相关的东西，因此，注重看pom文件，里面具体的代码就不实现了，下面是我项目骨架： ...

(<http://blog.csdn.net/StubbornPotatoes/article/details/12194391>)

程序猿不会英语怎么行？英语文档都看不懂！

软件工程出身的英语老师，教你用数学公式读懂天下英文→



Maven编译多子项目依赖



u011579138 2016年05月03日 11:58 14395

早在某公司实习的时候就听闻过Maven，只是听大神简单的介绍过，自己没有实习做过。之前做Android 的时候主要是用gradle（不过真心不熟），大概有点知道Maven的作用，这次来深圳某公司实习， ...

(<http://blog.csdn.net/u011579138/article/details/51303178>)

Maven多项目依赖配置



qq_15237993 2017年03月09日 10:14 624

niweiwei Maven多项目依赖配置 博客分类： Maven 最近在学习Maven，把一个开源的项目改成maven管理，期间使用到了多项目，从网上查...

(http://blog.csdn.net/qq_15237993/article/details/60955408)

Maven--多模块依赖实例解析（五）



hybaym 2016年03月04日 16:48 1998

《Maven--搭建开发环境（一）》 《Maven--构建企业级仓库（二）》 《Maven—几个需要补充的问题（三）》 《Maven—生命周期和插件（四）》 《Maven--多...

(<http://blog.csdn.net/hybaym/article/details/50803897>)

【Maven】——优化依赖



u012654963 2017年04月09日 16:16 805

上篇说到传递依赖，其优点我们有目共睹，其为我们提高了很多效率，减少了很多工作量。但现实是把双刃剑，完美中总会有所不足。Maven的传递依赖给我们带来的点点不足便是：我们本需要依赖的一些jar包，可能通...

(<http://blog.csdn.net/u012654963/article/details/69817615>)

Maven多项目依赖配置



qq584852076 2015年06月09日 14:01 1165

转载： <http://niweiwei.iteye.com/blog/1965760> 最近在学习Maven，把一个开源的项目改成maven管理，期间使用到了多项目，从网上查阅了一些资料，主要参考的是ht...

(<http://blog.csdn.net/qq584852076/article/details/46426027>)



21





程序猿不会英语怎么行？英语文档都看不懂！

软件工程出身的英语老师，教你用数学公式读懂天下英文→



Maven建立父子关系项目工程，建立依赖关系结构



Java使用maven创建父子工程项目，构建项目之间关系。便于对开发人员对项目工程进行分模块、分业务管理...

 u010786396 2017年11月10日 12:49  103

(<http://blog.csdn.net/u010786396/article/details/78497872>)



解决Maven项目相互依赖/循环依赖/双向依赖的问题

很多时候随着项目的膨胀，模块会越来越多，如果设计上 稍有不慎就会出现模块之间相互依赖的情况。这对于使用M...

 Leolu007 2016年11月08日 10:42  6245

(<http://blog.csdn.net/Leolu007/article/details/53079875>)



Maven详解之聚合与继承

 wanghantong 2014年08月07日 23:06  63233

Maven的聚合与继承，如何在项目中正确的使用聚合与继承，来提高项目的可读性和可用性。如何做到项目的依赖管理和插件的正确管理？...

(<http://blog.csdn.net/wanghantong/article/details/36427411>)



Maven工程聚合,多个Java工程、web工程聚合

最近写个自己的小项目,牵扯到maven工程聚合问题,网上看了大  qq_32588349 2016年07月17日 23:08  14687
量资料研究了下,成功了,按照自己的理解简单粗暴的记录

下。。。一、模块结构粗略画了个草图表示下现有模块之间的关系二、模块作用及配置文件 to...

(http://blog.csdn.net/qq_32588349/article/details/51934100)

maven聚合和继承(一次操作多个项目)

 u011306224 2017年04月08日 02:23  393

1、聚合假设有项目A和项目B，我们想一次性构建两个项目，而不是到两个模块的目录下分别执行mvn命令。maven聚合解决了该问题。 这时候我们要创建另外一个项目ALL，然后通过该模块构建整个项目的...

(<http://blog.csdn.net/u011306224/article/details/69664211>)



21





21



程序员不会英语怎么行？

老司机教你一个数学公式秒懂天下英语



Maven整合SSM的最基本聚合项目



chenhaotao 2017年11月29日 11:26 165

打算学习springboot和springcloud，在此之前先尝试搭建一个最简洁的SSM项目，对比之后跟能感受springboot的强大和便捷。项目说明：sclipse+maven+SSM，只是...

(<http://blog.csdn.net/chenhaotao/article/details/78663383>)

maven聚合工程的创建和聚合工程的打包



millery22 2015年09月23日 11:21 18984

第一步：创建父工程millery-manage，如图：右击空白处，new创建新maven工程：这里跳过默认的骨架，使用自动义的骨架 这里父工程必须使用pom打包方式 第...

(<http://blog.csdn.net/millery22/article/details/48677643>)

Maven+myeclipse 创建聚合项目



cuiyaonan2000 2014年03月13日 15:32 3589

Maven+myeclipse 创建聚合项目 首先:在myeclipse中 file-->new-->other,在上输入maven,点击如图中的选项 然后next 显示如下 点击n...

(<http://blog.csdn.net/cuiyaonan2000/article/details/21172711>)

maven 项目之间的依赖



y666666y 2017年04月17日 15:28 874

父项目的pom.xml文件的相对路径。默认值为../（同一工作空间下另一个项目的根目录）pom.xml。maven首先从当前构建项目开始查找父项目的pom文件，然后从本地仓库，最有从远程仓库。Rela...

(<http://blog.csdn.net/y666666y/article/details/70211815>)

maven 项目拆分配置文件为单独的项目（即maven项目如何依赖另一个项目的配置...

在学习dubbo 分布式框架的时候，需要把一个大的项目分割成多个服务，例如：用户服务模块（包括权限、用户登录、用户增删）、公共服务模块（包括文件上传等），这些服务模块之间可能需要依赖公共的配置文件，例...



u011244682 2017年07月27日 17:52 2159

(<http://blog.csdn.net/u011244682/article/details/76213224>)