Maven3 实战笔记 06 聚合

刘岩

Email:suhuanzheng7784877@163.com

1. Maven 聚合的概念

   聚合概念是由来已久，比如我们需要 2 个项目协调合作才能完成一个大的、完整的业务场景，这个时候就需要构建 2 个项目，但是呢，想一次性就构建这 2 个项目，而不是分别构建 2 个项目后再合在一起整。这样我们就需要一个比较特殊的项目了，就是所谓的聚合项目，这个项目没有别的目的，就是仅仅为了聚合多个项目模块用的。

2. 新的项目模块

   在介绍聚合项目前，先利用 IDE 建立一个新的项目模块——MavenAccount-persist，负责注册系统的持久层业务，按照 Maven 的项目规范，我们建立项目如下。

实体 POJO 如下

```
package pojo;

/**
 * 注册用户实体
 * @author liuyan
 */
public class Account {

    private String id;
    private String name;
```

```
    private String password;
    private String email;


//省去setter和getter


}
```

业务接口

```
package dao;
import pojo.Account;

/**
 * 业务接口
 * @author liuyan
 */
public interface AccountDAO {

    public boolean save(Account account);

    public boolean update(Account account);

    public Account query();

    public boolean delete(String id);

}
```

接口实现类

```
package dao.impl;

import pojo.Account;
import dao.AccountDAO;

/**
 * 业务实现类
 * @author liuyan
 */
public class AccountDAOImpl implements AccountDAO {

    @Override
    public boolean delete(String id) {
```

```
        System.out.println("删除记录");
        return true;
    }

    @Override
    public Account query() {
        System.out.println("查找记录");
        return null;
    }

    @Override
    public boolean save(Account account) {
        System.out.println("保存记录");
        return true;
    }

    @Override
    public boolean update(Account account) {
        System.out.println("更新记录");
        return true;
    }

}
```

之前咱们说过，Java 具体技术不是我们这个学习笔记系列的重点，所以我只是输出一些简单信息，并没有和任何的持久层框架结合起来。咱们权且当做完成了这个注册系统的持久层模块吧。

Spring 配置文件

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
     http://www.springframework.org/schema/aop
     http://www.springframework.org/schema/aop/spring-aop-2.5.xsd"
```

```
    xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:p="http://www.springframework.org/schema/p">

    <bean id="accountDAO" class="dao.impl.AccountDAOImpl">
    </bean>

</beans>
```

下面我们来看这个单独模块的 pom.xml 内容，虽然和 MavenAccount-email（http://suhuanzheng7784877.iteye.com/blog/1066917）有很多冗余，但是我还是想把它完整的粘贴下来

```
<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/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.liuyan.account</groupId>
    <artifactId>MavenAccount-persist</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <properties>
        <springversion>2.5.6</springversion>
        <junitversion>2.5.6</junitversion>
    </properties>


    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>${springversion}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-beans</artifactId>
            <version>${springversion}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>${springversion}</version>
```

```
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context-support</artifactId>
            <version>${springversion}</version>
        </dependency>
        <dependency>
            <groupId>javax.mail</groupId>
            <artifactId>mail</artifactId>
            <version>1.4.1</version>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.7</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>com.icegreen</groupId>
            <artifactId>greenmail</artifactId>
            <version>1.3.1b</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <resources>
            <resource>
                <directory>src/main/resource</directory>
            </resource>
        </resources>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-source-plugin</artifactId>
                <version>2.1.1</version>
                <executions>
                    <execution>
                        <id>buildSource</id>
                        <goals>
                            <goal>jar-no-fork</goal>
                        </goals>
                        <inherited>false</inherited>
                        <configuration>
```

```
                    </configuration>
                </execution>
            </executions>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <configuration>
                <target>1.5</target>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-resources-plugin</artifactId>
            <configuration>
                <target>UTF-8</target>
            </configuration>
        </plugin>
    </plugins>
  </build>
</project>
```

3. 聚合项目

以上模块作为一个单独的项目模块来说其实没什么问题，pom 配置文件写得那么多也无可厚非，但是上面的持久层模块势必要和咱们之前的邮件模块协同合作，那么问题就来了，配置冗余，很多配置内容都是重复的。比如两个项目都用到了 spring，都用到了 junit 依赖，那么是不是在一个地方配置好了，大家统一用它就行了，不用大张旗鼓的再配置一遍吧。还有一个问题就是假如说邮件模块使用的 Spring 版本是 2.5.6，而持久层模块使用的版本是 3.0.1，这就造成了一个大系统，不同模块之间使用的第三方辅助包的版本不一致，那么有可能每一个单独的模块构建成功，到集成测试的时候死活就 build 不过去！Maven 的聚合机制就解决了这一问题。如上所说，我们应该建立一个新 Maven 项目，专门负责聚合这些模块用的。这个项目就一个 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/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.liuyan.account</groupId>
    <artifactId>MavenAccount-aggregator</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>pom</packaging>
    <modules>
        <module>../MavenAccount-email</module>
        <module>../MavenAccount-persist</module>
    </modules>
</project>
```

<packaging>里面的内容必须是 pom 类型

```
    <modules>
        <module>../MavenAccount-email</module>
        <module>../MavenAccount-persist</module>
    </modules>
```

里面就是聚合模块需要聚合一起构建的项目（或者可以说是文件夹）的名称，../代表聚合项目与这 2 个模块属于同级文件夹。

在命令行将文件夹 cd 到此聚合项目下，执行如下命令

```
mvn clean install
```

之后两个模块就一起构建了。

```
[INFO] ------------------------------------------------------------------------
[INFO] Building MavenAccount-aggregator 0.0.1-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.4.1:clean (default-clean) @ MavenAccount-aggrega
tor ---
[INFO]
[INFO] --- maven-install-plugin:2.3.1:install (default-install) @ MavenAccount-a
ggregator ---
[INFO] Installing E:\Genuitec\ws\MavenAccount-aggregator\pom.xml to C:\Users\liu
yan\.m2\repository\com\liuyan\account\MavenAccount-aggregator\0.0.1-SNAPSHOT\Mav
enAccount-aggregator-0.0.1-SNAPSHOT.pom
```

```
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] MavenAccount-email ..... SUCCESS [6.553s]
[INFO] MavenAccount-persist ..... SUCCESS [1.498s]
[INFO] MavenAccount-aggregator ..... SUCCESS [0.062s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 8.300s
[INFO] Finished at: Sun Jun 05 12:51:01 CST 2011
[INFO] Final Memory: 13M/31M
[INFO] ------------------------------------------------------------------------
```

4. 总结

我们这里仅仅介绍了集合模块一起构建的方式，当然如果模块少，一个一个分别 build 成功后再集成也无可厚非，看项目具体情况而定。这次我们留了一个问题，就是这两个项目模块使用的依赖包版本不一致怎么办，因为 pom.xml 是分别声明的依赖，这个问题我们留在了 Maven 继承机制解决。