

Nell'articolo di oggi vedremo una delle caratteristiche più interessanti di maven, ovvero quella di poter strutturare un progetto in moduli, componenti del progetto principale detto parent. In particolare definiremo un progetto composto da una interfaccia grafica realizzata in flex e un back-end che espone servizi soa basati su spring ws. Per le caratteristiche di spring-ws rimando alla sezione dedicata.

Iniziamo con il creare il progetto parent tramite la direttiva base

```
mvn archetype:create -DgroupId=it.finazzo.esempio -DartifactId=progettoModulare
```

Adesso dobbiamo configurare i moduli all'interno del nostro progetto. Per farlo è sufficiente modificare il tag di packaging specificando pom al posto di jar e indicando i moduli tramite il tag modules.

```
<modules>
<module>../front-end</module>
<module>../back-end</module>
</modules>
```

Creiamo il progetto relativo al front-end realizzato in flex. In questo caso ci avvaliamo di [flexmojos](http://flexmojos.org), un set di plugin realizzati da Sonatype per consentire in maven la compilazione, l'ottimizzazione e il testing di componenti flex. Mister Sonatype ha messo a disposizione una serie di archetipi per soddisfare le nostre esigenze: la creazione di una libreria, la creazione di una applicazione flex e la creazione di un progetto progetto suddiviso in moduli swc, swf e flex. Nel nostro caso ci avvaliamo del secondo archetipo e scegliamo la versione stable che è , nel momento in cui scrivo, la 3.9.

```
mvn archetype:generate
-DarchetypeRepository=http://repository.sonatype.com/content/groups/flexgroup
-DarchetypeGroupId=org.sonatype.flexmojos
-DarchetypeArtifactId=flexmojos-archetypes-application -DarchetypeVersion=3.8
```

-DgroupId=it.finazzo.progettoModulare -DartifactId=front-end

Generato il progetto occorre modificare il pom per includere il repository maven di Sonatype per consentire il download di tutti i componenti necessari alla compilazione. Potete includere il repository all'interno del pom file dello specifico progetto o nel file settings.xml per renderlo disponibile per tutti i progetti.

```
<repositories>
<repository>
<id>flex-mojos-repository</id>
<url> http://repository.sonatype.org/content/groups/flexgroup/</url>
</repository>
</repositories>
```

Infine specifichiamo il path del flash player stand alone per poter eseguire i test tramite la variabile flashPlayer.command

```
<properties>
<flashPlayer.command>pathflashplayer_10_sa_debug.exe</flashPlayer.command>
</properties>
```

Creiamo il progetto del back-end realizzato in spring-ws 2.0. Usiamo la direttiva che abbiamo usato in uno dei precedenti articoli per generare il nostro back-end.

```
mvn archetype:create -DarchetypeGroupId=org.springframework.ws
-DarchetypeArtifactId=spring-ws-archetype -DarchetypeVersion=2.0.2.RELEASE
-DgroupId=it.finazzo.progettoModulare -DartifactId=back-end.
```

All'interno del progetto back-end specifichiamo la dipendenza dal progetto front-end tramite il tag dependencies

```
<dependency>
<groupId>it.finazzo.progettoModulare</groupId>
<artifactId>front-end</artifactId>
<version>1.0-SNAPSHOT</version>
<type>swf</type>
</dependency>
```

e includiamo il plugin flexmojos-maven-plugin responsabile di copiare gli swf generati nella root del nostro back-end.

```
<plugin>
<groupId>org.sonatype.flexmojos</groupId>
<artifactId>flexmojos-maven-plugin</artifactId>
<version>3.5.0</version>
<executions>
<execution>
<goals>
<goal>copy-flex-resources</goal>
</goals>
</execution>
</executions>
</plugin>
```

Per verificare il corretto funzionamento lanciamo la direttiva mvn package sul progetto parent. I moduli verranno compilati e assemblati nell'ordine dichiarato, ottenendo il seguente output:

```
[INFO] -----
[INFO] Reactor Summary:
[INFO] -----
[INFO] progettoModulare ..... SUCCESS [0.779s]
[INFO] front-end Flex ..... SUCCESS [16.670s]
[INFO] back-end Spring-WS Application ..... SUCCESS [11.676s]
```

[Qui](#) trovate i file necessari per fare il vostro progetto modulare.

[Prossimo](#)