Apache Maven

Apache Maven predstavlja alat koji pojednostavljuje kreiranje, testiranje i paketiranje projekata. Pruža preporuke gde će se nalaziti različiti delovi projekta kao što su *source* kod, *test* kod i konfiguracioni fajlovi. Takođe, kako mnogi projekti zavise od drugih projekata i *open source framework*-a kako bi funkcionisali, a pojedinačno rešavanje tih zavisnosti (*dependencies*) ručno je mukotrpno, *Maven* nudi rešenje da se sve te zavisnosti navode u eksternom *pom.xml* fajlu. Na taj način će se automatski skinuti svi fajlovi od kojih zavisi projekat na kojem se radi. Bitno je napomenuti da se u *pom.xml* fajlu navodi samo **šta** je potrebno za projekat, ali ne i **kako** do toga doći.

Instalacija Apache Maven

U cilju instalacije *Apache Maven* alata potrebno je ispratiti sledeći niz koraka[1]:

1. Raspakovati *Maven* instalacioni arhiv na proizvoljnoj lokaciji na disku
2. Napraviti novu sistemsku promenljivu `M2_HOME` i dodeliti joj vrednost koja predstavlja punu putanju do direktorijuma u kojem je *Maven* arhiva raspakovana
3. U sistemsku promenljivu `PATH` dodati `%M2_HOME%/bin` kako bi se *Maven* mogao koristiti iz komandne linije

Dodatna podešavanja se mogu uneti u *settings.xml* fajl koji se kreira u `c:\Users\<<user_name>>\.m2` folderu. U .m2 folderu se nalazi *repository* folder u koji se smeštaju sve biblioteke koje *Maven* skida sa interneta. Ovo, kao i mnoge druge stvari se može promeniti izmenom *settings.xml* fajla.

Maven Dependency Management

Projekti tipično zavise od nekih biblioteka. Da bi se recimo iskoristila *hibernate-core* biblioteka potrebno bi bilo da se ode na stranicu sa koje bi se skinuo *JAR* fajl i zatim stavio u *lib* folder projekta ili dodao na *classpath*. Problem koji bi se tu javio bi bio da taj *JAR* fajl zavisi od nekih drugih biblioteka. Onda bi bilo potrebno da se pronađu te nove biblioteke i da se dodaju u projekat. Drugi problem bi bio ako bi se verzija *JAR* fajla promenila i tada bi ceo prethodni postupak morao da se ponovi. Da bi se ovakvi problemi sprečili *Maven* obezbeđuje deklarativni *dependency management*. Ovim pristupkom, sve potrebne zavisnosti (eksterne biblioteke) se deklarišu u eksternom *pom.xml* fajlu. *Maven* onda automatski skida te biblioteke i njihove funkcionalnosti se mogu koristiti u projektu. *Maven dependencies* su obično arhive sa *JAR, WAR, EAR* i *ZIP* ekstenzijama. Svaka od tih biblioteka se može identifikovati tzv. *Maven* koordinatama:

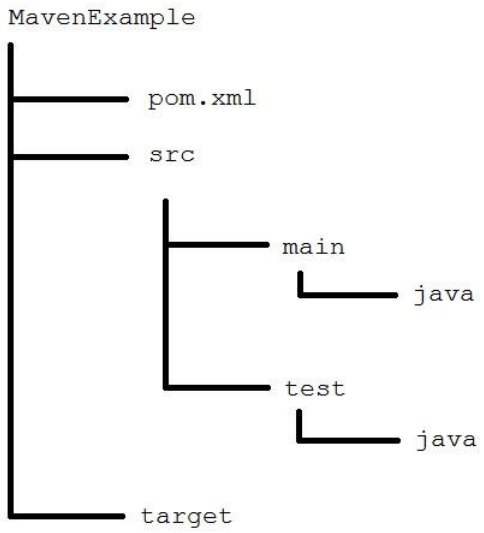
[1] Ukoliko se *Apache Maven* skriptovi koriste u okviru *Eclipse* okruženja, potrebno je instalirati *Maven* plug-in gde će se u *Eclipse* ići na **Help → Install New Software …** i sa adrese **http://download.eclipse.org/technology/m2e/releases** skinuti potreban plug-in.

1. `groupId`: identifikator organizacije ili grupe koja je odgovorna za biblioteku (npr. `org.hibernate`)
2. `artifactId`: identifikator artifakta koji generiše projekat (mora biti jedinstven među artifaktima koji pripadaju istom `groupId`)
3. `version`: predstavlja verziju biblioteke (npr. `4.2.2.Final`)
4. `type`: predstavlja ekstenziju generisanog artifakta

Ukoliko postoje biblioteke koje nisu javno dostupne a koje se koriste u projektu, postoji mogućnost da se *Maven* komandama one dodaju. Takođe, *Maven* obezbeđuje tzv. *scope* u kojem će biblioteke biti dostupne (npr. *JUnit JAR* je potreban samo tokom faze testiranja).

Maven projekat

Organizacija jednog *Maven* projekta prikazana je na slici 1.1.



Slika 1.1 - Primer *Maven* projekta

Komponente projekta sa slike su:

- Koreni folder projekta `MavenExample` (obično ime projekta odgovara `artifactId` iz *pom.xml* fajla)
- `src` folder sadrži sve artifakte koji su u vezi sa projektom
- `src/main/java` folder sadrži *Java source* kod
- `src/test/java` folder sadrži *Java unit test* kod
- `target` folder sadrži generisane artifakte kao što su `.class` fajlovi
- *pom.xml* fajl nalazi se u korenom folderu projekta i sadrži sve konfiguracione informacije

Projekat može sadržati i dodatne foldere u zavisnosti šta je za projekat potrebno (npr. foldere koji će sadržati dodatne konfiguracione fajlove, resurse itd).

pom.xml

Fajl *pom.xml* je jedini obavezan u *Maven* projektu. Nalazi se u korenom folderu projekta. Počinje `project` elementom unutar koga se dalje nalaze `modelVersion`, `groupId`, `artifactId`, `version`, `packaging` i drugi elementi. Primer jednog jednostavnog *pom.xml* fajla dat je u listingu 1.1.

```
<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>rs.ac.uns.ftn.informatika</groupId>
	<artifactId>MavenExample</artifactId>
	<version>1.0.0-SNAPSHOT</version>
	<packaging>jar</packaging>
	<name>Primer Maven projekta</name>
	<url>http://blabla.xyz</url>

	<properties>

		<junit.version>4.11</junit.version>

	</properties>

	<dependencies>


		<dependency>


			<groupId>junit</groupId>
			<artifactId>junit</artifactId>
			<version>${junit.version}</version>
			<scope>test</scope>

			<exclusions>


				<exclusion>


					<groupId> org.hamcrest</groupId>
					<artifactId>hamcrest</artifactId>

				</exclusion>

			</exclusions>

		</dependency>

	</dependencies>
```

```
</project>
```

Listing 1.1 - Primer *pom.xml* fajla

Pored prvih nekoliko navedenih elemenata u *pom.xml* fajlu koji su ranije opisani, prvi novi element je `packaging` koji *Mavenu* signalizira da je potrebno napraviti *JAR* arhivu projekta. Sledeći je `dependencies` element unutar koga se navode biblioteke od kojih projekat zavisi. Svaka zavisnost se pojedinačno navodi unutar `dependency` elementa, a informacije koje su potrebne da se navedu su *Maven* koordinate (`groupId`, `artifactId` i `version`), dok `scope` element nije obavezan (u navedenom primeru na listingu 1.1 `scope` ima vrednost *test* što označava da se JUnit biblioteka koristi samo u fazi testiranja projekta), kao i `exclusions` element koji označava koje povezane biblioteke sa navedenom bibliotekom ne treba uključiti u projekat. Ukoliko se često menjaju verzije biblioteka pogodno je informacije o verzijama držati na jednom mestu, a pomoću tzv. *placeholdera Maven* će odgovarajuće vrednosti ubaciti tamo gde je označeno. Za ovakve slučajeve može se koristi `properties` element unutar koga se navode elementi koji čuvaju vrednosti koje su bitne, a zatim se one očitavaju unutar `${<<naziv_elementa>>}`.

Maven životni ciklus

Proces generisanja artifakata podrazumeva nekoliko koraka i zadataka koji se moraju izvršiti. Primeri tih zadataka uključuju kompajliranje koda, pokretanje unit testova i pakovanje artifakata. Maven koristi koncept ciljeva (*goals*) da predstavi te granularne zadatke. Ciljevi se pakuju u *plug-inove* koji predstavljaju kolekciju sačinjenu od jednog ili više ciljeva. Ciljevi se mogu pokrenuti unošenjem komandi sa sledećom sintaksom:

```
mvn plugin_identifikator:goal_identifikator
```

Maven prati ustanovljeni niz koraka koji se izvršavaju u istom redosledu nezavisno od artifakta koji se pravi. Postoje tri ugrađena životna ciklusa:

1. *Default*: barata fazama kompajliranja, pakovanja i deployovanja *Maven* projekata
2. *Clean*: barata brisanjem privremenih fajlova i generisanih artifakata iz `target` foldera
3. *Site*: barata generisanjem dokumentacije

Svaki životni ciklus ima svoje faze:

1. *Validate*: proverava da li u projektu postoje greške i da li su sve biblioteke dostupne
2. *Compile*: kompajlira kod
3. *Test*: pokreće unit testove
4. *Package*: pakuje kompajliran kod u neku od arhiva
5. *Install*: instalira arhivu na lokalni repozitorijum. Tada je arhiva dostupna svakom projektu koji se nalazi na toj mašini
6. *Deploy*: smeša arhivu na udaljeni repozitorijum svima na korišćenje

U svakoj fazi se izvršavaju određeni zadaci i svaka faza je povezana sa jednim ili više ciljeva. Faze delegiraju zadatke svojim ciljevima koje izvršavaju plug-inovi.

Maven Archetypes

Kako se *Maven* projekti ne bi pravili ručno od nule, mogu se iskoristiti *Maven archetypes* koji će izgenerisati početni izgled projekta. Potrebno je uneti sledeću komandu:

```
mvn archetype:generate
```

Posle toga je potrebno uneti broj šablona projekta iz liste koji se generiše kao i *Maven* koordinate. Alternativno se mogu direktno uneti svi potrebni parametri.

Literatura

[1] Apache, *Maven*, https://maven.apache.org/

[2] Balaji Varanasi, Sudha Belida, *Introducing Maven*, Apress 2014