Java Web Development

Modul 2

Termin 3

Sadržaj

Maven

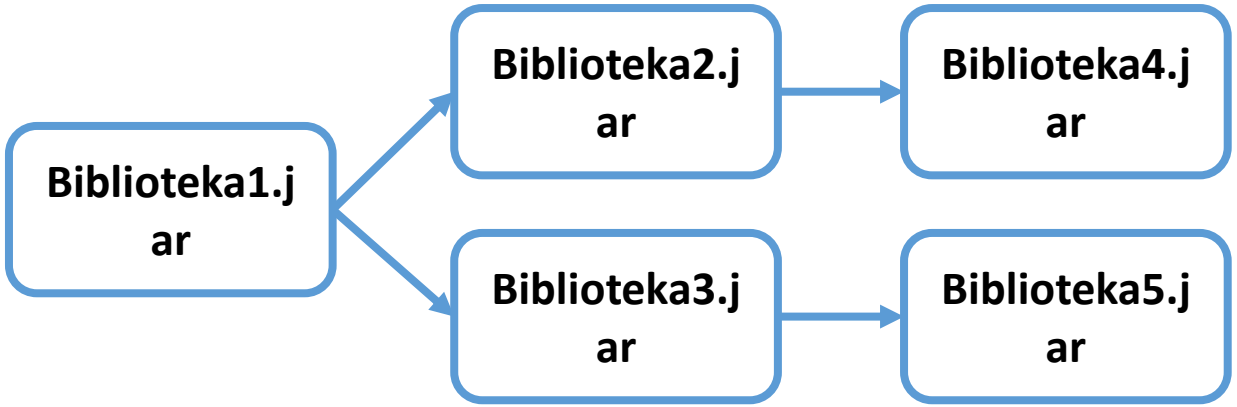
Uvod

- Za kreiranje Spring aplikacije koristiće se Maven alat.
- Maven je ***build tool*** alat (https://maven.apache.org/users/index.html) i ima za cilj olakšavanje razvoja kompleksnih projekata.
- Alat za **izgradnju** tj. konstrukciju projekta kompleksnih aplikacija.
- Korišćenje
  - nazavisan *stand alone* alat koji se poziva preko komandne linije. Neophodno je alat preuzeti sa https://maven.apache.org/
  - pozvati iz Eclipse softvera oslanjajući se na Maven Eclipse plug-in (već integrisan u novijim verzijama Eclipse softvera)

Maven

Dependency management - bez Maven

- Kod klasičnog projekta bez Mavena kada bi bilo neophodno koristiti neku biblioteku neophodno bi bilo fizički ubaciti biblioteku u projekat i dodati je u *build path* projekta.
- U slučaju da ubačena biblioteka zavisi od nekih drugih biblioteka neophodno bi bilo pribaviti i ubaciti i ostale tražene biblioteke.
- Proces ponavljati sve dok više nema zavisnih biblioteka



Maven

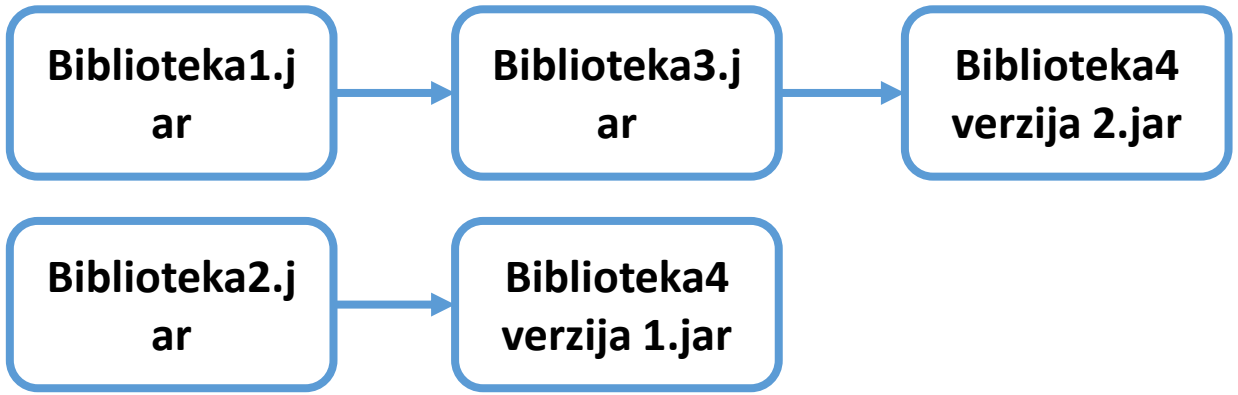
Dependency management - sa Maven

- Maven omogućva *Dependency management* tj. automatsko upravljanje dependency-ima za korišćene biblioteke.
- Kod projekta sa Maven kada bi bilo neophodno koristiti neku biblioteku neophodno bi bilo samo navesti zavisnost projekta od te biblioteke u **pom.xml** fajlu, a Maven će sam naći tu biblioteku, pruzeti je, ubaciti u projekat, dodati je u *build path* projekta.
- Maven će ponoviti isti postupak za sve biblioteke od kojih preuzeta bibliteka zavisi, i ponoviti isti process za novopreuzete bibliteke, sve dok više ne bude zavisnosti.
- Proces ponavljati sve dok više nema zavisnih biblioteka

Maven

Dependency management - sa Maven

- Maven će detektovati i obavestiti nas o situacije da u projektu treba da se preuzmu različite verzije iste biblioteke, što nije moguće.



Maven

Životni ciklus izgradnje veb aplikacije

- Prethodno kada ste razvijali veb aplikaciju sa Servletima bilo je neophodno:
  - **kompajlirati Java koda veb aplikacije tj. dobiti class fajlove od Java fajlova**
  - **testirati delove veb aplikacije izvršavanjem Unit testova**
  - **zapakovati class fajlove i ostale veb resurse (HTML, CSS, slike i druge fajlove) u arhivu, čija je ekstenzija war**
  - **ubaciti war arhivu u veb kontejner koji je bio Tomcat tj. izvršiti *deployment* veb aplikacije**
  - Startovati Tomcat, a on će izvršiti kompajlirani Java kod deplojovane veb aplikacije
- Svi koraci koji su boldovani predstavlju ***Build life-cycle*** veb aplikacije
- Glavni zadatka Maven alata je da izvršava *Build life-cycle* (https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html)
- Maven *Build life-cycle* predstavlja niz koraka tj. faza koje se izvršavaju svaki put kada se builduje Maven aplikacija

Maven

Životni ciklus izgradnje veb aplikacije

- Mogu se izvršiti svi koraci ili se može izvršiti do određenog koraka.
- Posle svakog uspešnog izvršenog koraka ide se na sledeći
  - *validate* – proverava tj. validira da li je projekat korektan, da li su tu svi neohodni delovi projekta. Ako je sve u redu ide
  - *compile* – pretvara izvorni u izvršni kod.
  - *test* – testiranje aplikacije, pokreću se Unit testovi. Ako testovi prođu ide
  - *package* – od kompajliranog koda, u zavisnosti od tipa aplikacije, kreira se paket tipa *war* ili *jar*.
  - *verify* – pokreću se integracioni tetovi
  - *install* – instalacija paketa u lokalni repozitorijum
  - *deploy* – instalacija paketa u udaljeni repozitorijum
- Zašto su neophodni svi ovi koraci?

Maven

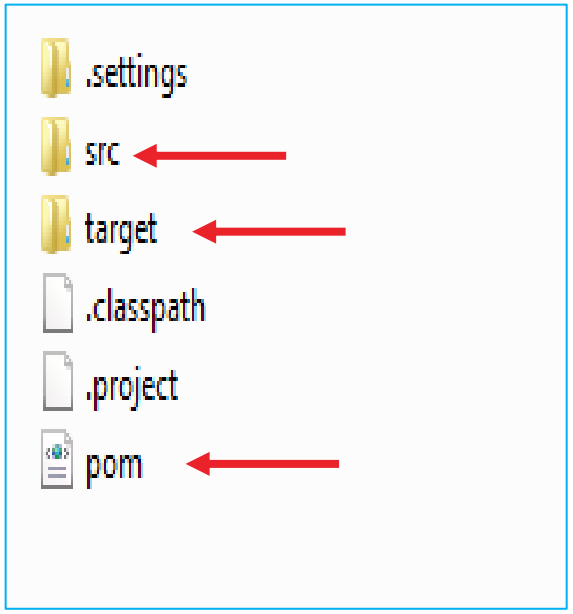
Convention over Configuration

- U Maven projektu se poštuje princip *Convention over Configuration* koji se još naziva i *coding by convention*
- Omogućava dobijanje određenog željenog ponašanja rada aplikacije bez potrebe da se pišu konfiguracioni fajlovi
- Ideja je da se konfiguracija projekta ne radi eksplictno, već da postoji dogovor/konvencija po kojoj se od programera očekuje samo da se klase za određenu funkcionalnost nalaze u tačno predefinisanim folderima u projektu, da se one nazivaju po nekoj konvenciji i da je na taj način projekat iskonfigursan adekvatno.
  - Ukoliko je neophodno izmeniti nešto od podrazumevanih dogovora, dostupno je dodatno konfigurisanje projekta

Maven projekat

Struktura projekta

- Maven koristi specifičnu strukturu projekta i ta struktura **MORA** biti ispoštovana - *Standard Directory Layout* (https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html)
- U folderu *src* se nalazi izvorni kod (šta mi programiramo), dok se
- U folderu *target* nalazi sve što je rezultat Maven *Build life-cycle*.
- Fajl pom.xml sadrži konfiguraciju Maven projekta.



Maven projekat

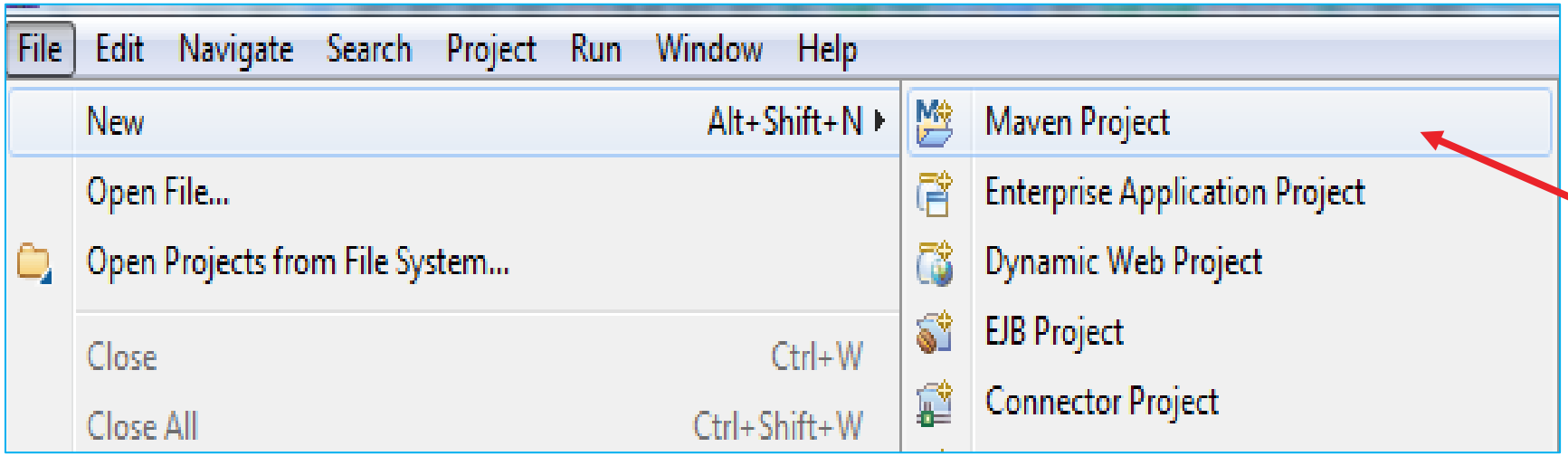
Struktura projekta

- Folder src sadrži predefinisane podfoldere
  - **src/main/java** – se nalazi izvorni kod aplikacije
  - **src/main/resources** – se resursi koji nisu Java fajlovi, npr. property fajlovi, konfiguracioni fajlovi
  - **src/main/webapp** – se nalazi **web.xml** fajl (deployment descriptor), folder **WEB-INF**, **statički resursi**, slike, CSS fajlovi, JavaScript fajlovi, biblioteke, itd. Takođe, tamo se nalazi i izvorni kod za kreiranje prezentacionog sloja aplikacije (fajlovi: .jsp, .jspf, .jspx, .ftl, .ftlh, .ftlx, .html).
    - Spring može da koristi različite Template Engine za kreiranje prezentacionog sloja MVC aplikacije. Za kreiranje dinamičkih HTML stranica može se koristiti JavaServer Pages (.jsp), FreeMarker (.ftl), Thymeleaf (.html),…
  - **src/test/java** – se nalaze Unit testovi izvorni kod aplikacije

Maven projekat

Kreiranje projekta

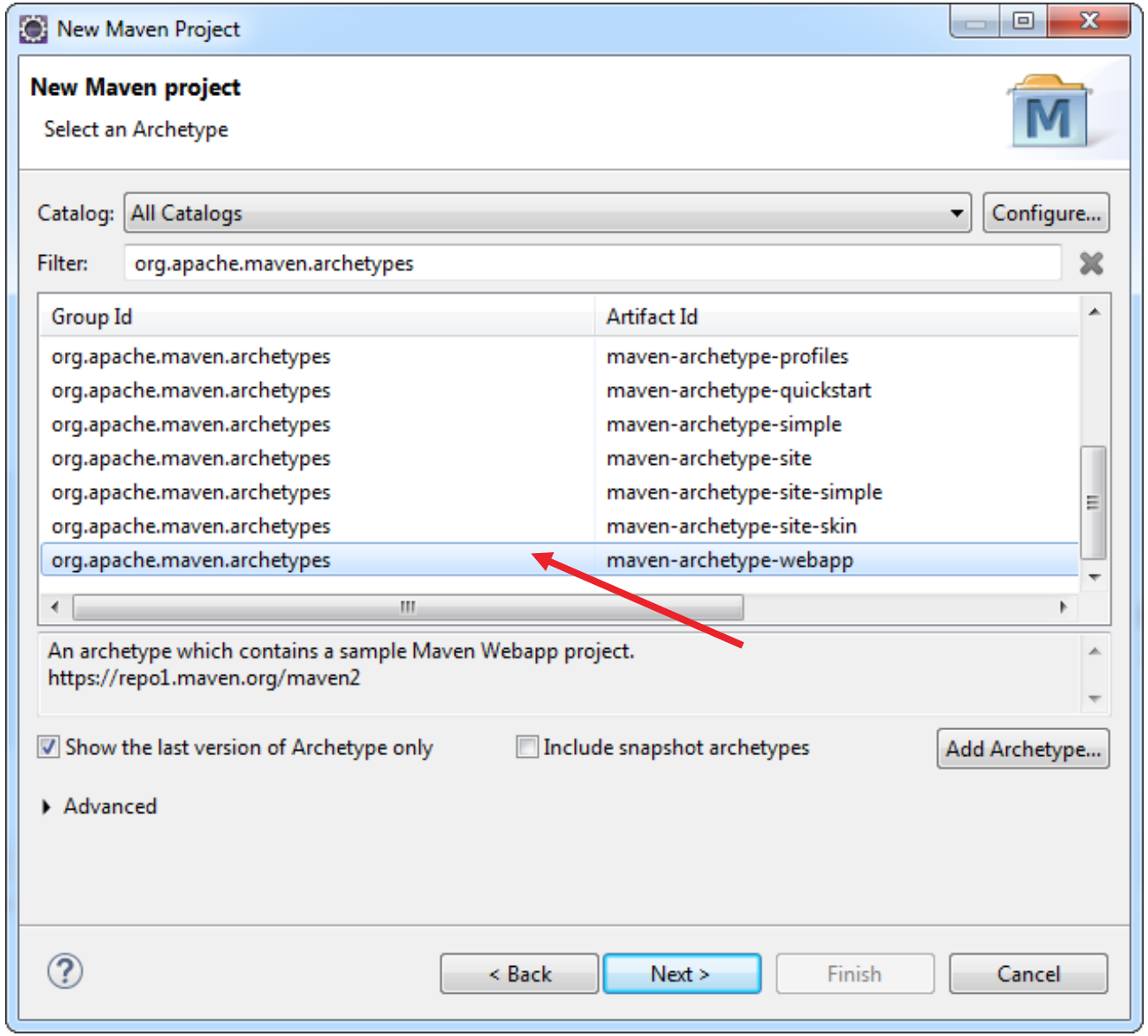
- Iz *Java EE* perspective, kliknite *File->New-> Maven Project*
- Otvoriće se novi prozor u kome samo klik *Next*.



Maven projekat

Kreiranje projekta

- U prikazanom prozoru treba da se odabere tip Maven arhitekture koji odgovara veb projektu.
- Odabrati za *Group Id* vrednost *org.apache.maven.archetypes* a za *Artifact Id* vrednost *maven-archetype-webapp*, pa *Next*.



Maven projekat

Group Id i Artifact Id

- *Group Id* označava grupu projekta, najčešće nešto vezano za organizaciju.
- Grupa *maven.archetypes* predstavlja konfuguracije Maven projekta koja se koristi za kreiranje određenog tipa aplikacije.
- *Artifact Id* prestavlja osnovnu gradivnu jedinicu u Maven alatu.

Maven projekat

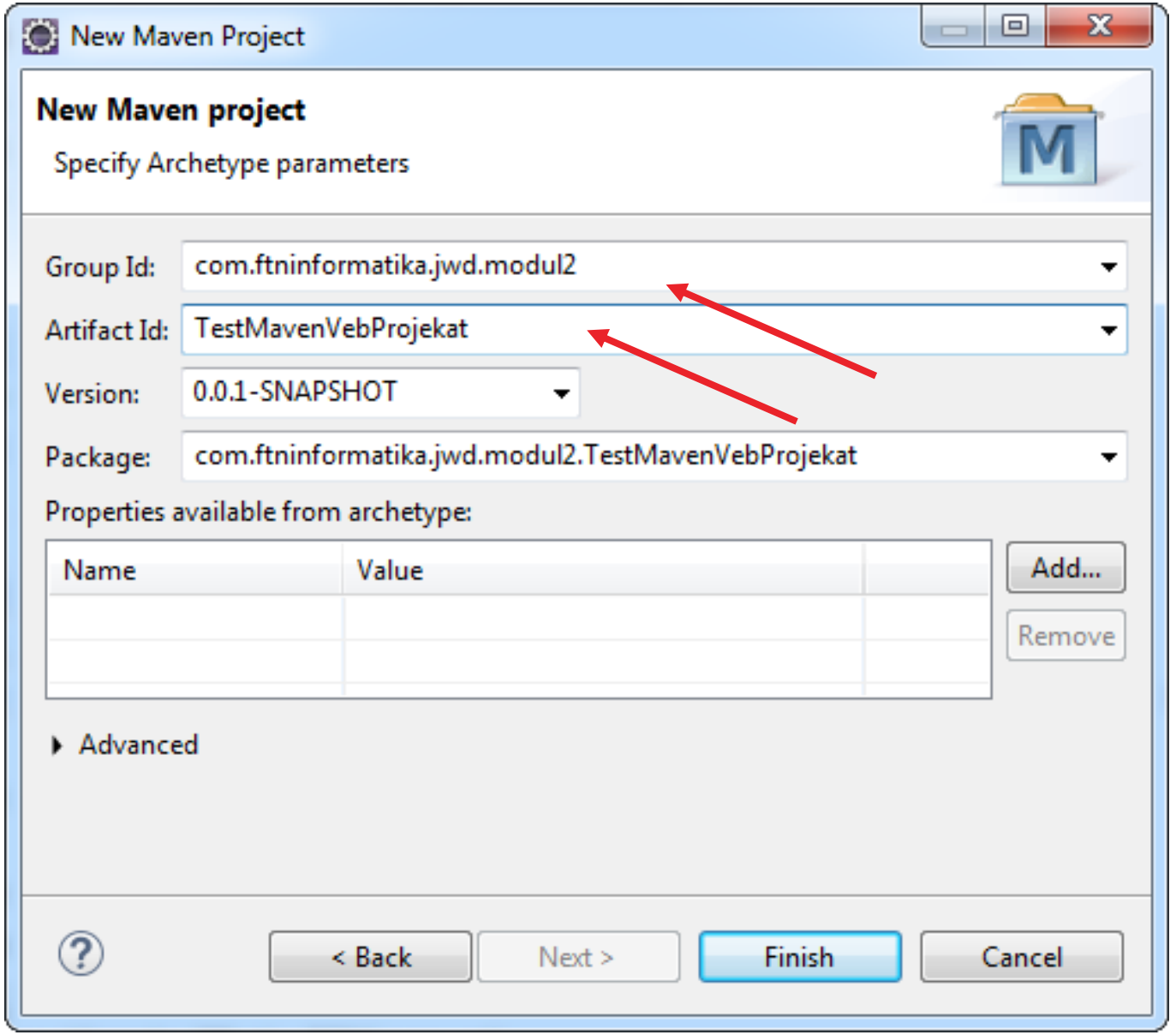
Kreiranje projekta

- U novom prozoru treba da se unesu vrednosti za *Group Id*, *Artifact Id* i *version* za projekat koji se kreira.
- Pomenute vrednosti imaju za cilj da se za **projektnu deliverablu** - *project deliverable* (jar/war/ear …) definiše jedinstveni identitet u repozitorijumu.
- **Projekat koji se kreira predstavlja jedan Maven artifakt, a drugi projekti od kojih kreirani projekat će zavisiti predstavlju druge Maven artifakte**

Maven projekat

Kreiranje projekta

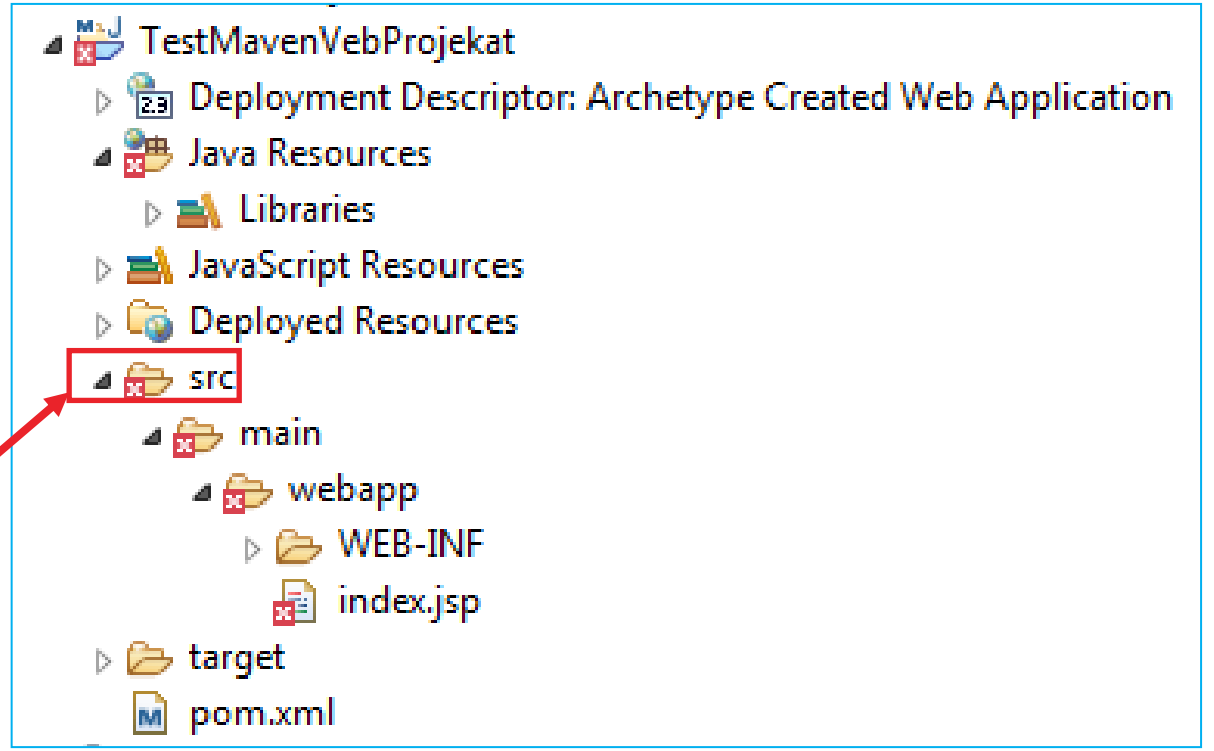
- Za *Group Id* unesite *com.ftninformatika.jwd.modul2*
- Za *Artifact Id* unesite *TestMavenVebProjekat*
- Za verziju neka ostane predložena vrednost
- Maven će predložiti naziv korenskog paketa koji se koristi u projektu, nastaje spajanjem vrednosti *Group Id* i *Artifact Id*.
- Klik na *Finish*.



Maven projekat

Prostorni raspored foldera

- Projekat ima ikonicu M
- Maven će preuzeti neophodne stvari sa internet, pogledati desni donju ugao. Sačekajte dok se proces preuzimanja ne završi
- Obrišite index.jsp fajl
- Potrebno je kreirati ostale foldere koji nedostaju a u skladu su sa predefinisanim prostornim rasporedom foldera u Maven projektu

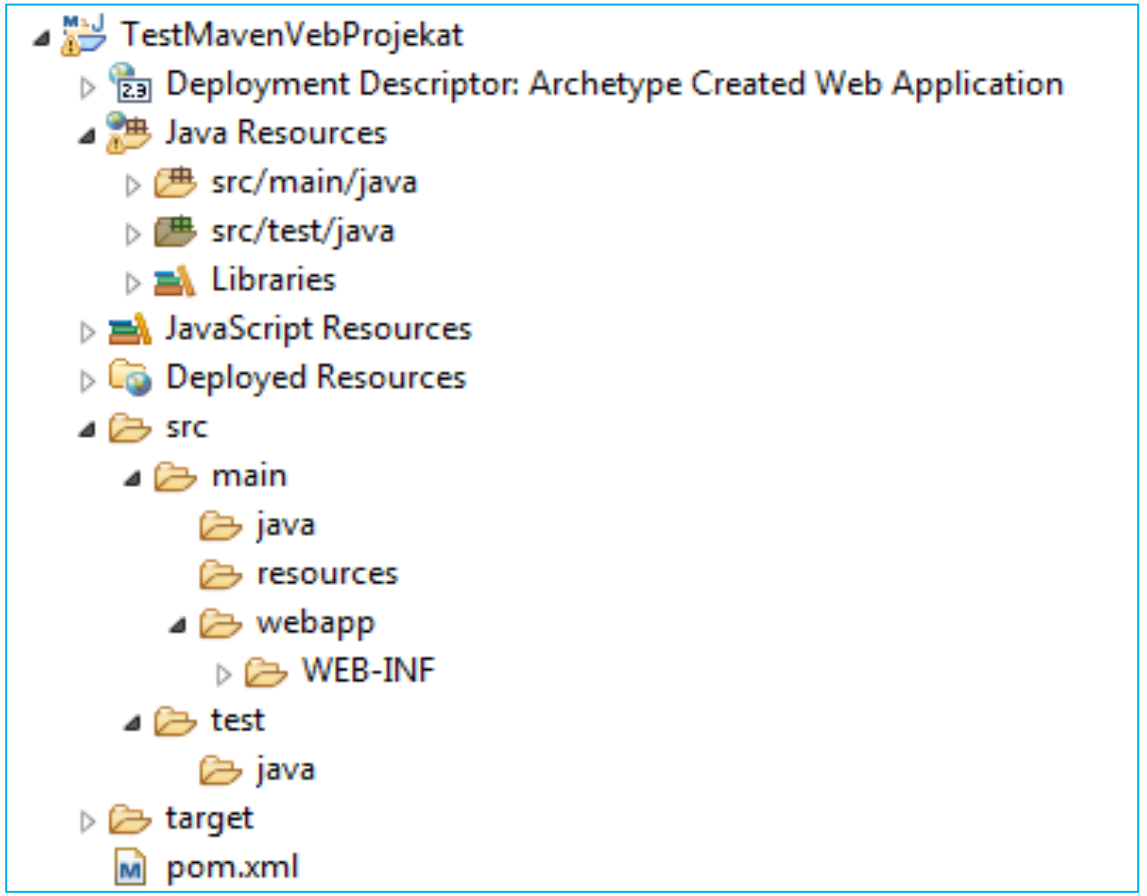


- **src/main/java**
- **src/main/resources**
- **src/test/java**

Maven projekat

Prostorni raspored foldera

- Primetite da folder *src* postoji u gornjem delu i u donjem delu projekta.
- U gornjem delu se nalazi sve što je ubačeno u **Build Path** projekta
  - Src u folderu *Java Resources* kao folder za Java pakete
- U donjem delu projekta se nalazi kao fizička reprezentacija foldera na disku.
- U gornjem delu se kreiraju fajlovi koji će se uvesti kao Java resursi za projekat (svi Java fajlovi se kreiraju ovde), dok se u donjem delu kreiraju obični fajlovi koji se samo nalaze na toj lokaciji na disku.



- Može se folder iz donjeg dela dodati u gornji deo sa
  Desni klik->Build Path->Use as SourceFolder

Maven projekat

pom.xml

- **<modelVersion>** - predstavlja verziju pom modela koji se koristi
- **Group Id, Artifact Id, version** – uneti kroz process kreiranja projekta
- **<packaging>** - označava da će Maven *Build life-cycle* u fazi *package* kreirati projektnu deliverablu kao **war** arhiva (jar/war/ear…)
- **<name>** - opisno ime projekta
- **<url>** - link do dokumentacije projekta
- **<properties>** - imenovane varijable koje se mogu koristiti u ostatku fajla sa ${imePropetija}

```
<?xml version="1.0" encoding="UTF-8"?>

<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.
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.ftninformatika.jwd.modul2</groupId>
  <artifactId>TestMavenVebProjekat</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>

  <name>TestMavenVebProjekat Maven Webapp</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
  </properties>
```

Maven projekat

pom.xml

- **<dependencies>** - navode se biblioteke od kojih projekat zavisi.
- Za svaku zavisnost se definiše jedinstveni identitet (**<groupId>**, **<artifactId>**, **<version>**) i opseg (**<scope>**) kojim se navodi za koji deo *Build life-cycle* se koristi ta zavisnost.
  - Scope može da se izostavi, tada zavisnost važi za sve faze *Build life-cycle.*
- Trenutno je navedena zavisnost od **JUnit** biblioteke. Za konkretnu JUnit biblioteku se navodi opseg *test*, što znači da se JUnut biblioteka neće koristiti za narednu fazu koja dolazi posle *test* faze tj. za fazu *package*.

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.11</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Maven projekat

pom.xml

- **<build>** - opisuju se podešavanja vezana za bildovanje deliverable
- **<finalName>** - ime deliverable
- **<pluginManagement> <plugins>** - sekcija za priključcima, koji se sve priključci koriste za bildovanje projekta u projektnu deliverablu
- **<plugin>** - priključak se identifikuje sa **<groupId>**, **<artifactId>** i **<version>**. Za njega se može postaviti dodatna konfiguracija sa **<configuration>**.

```
<build>
  <finalName>TestMavenVebProjekat</finalName>
  <pluginManagement><!-- lock down plugins versions
    <plugins>
      <plugin>
        <artifactId>maven-clean-plugin</artifactId>
        <version>3.1.0</version>
      </plugin>
      <!-- see http://maven.apache.org/ref/current/
      <plugin>
        <artifactId>maven-resources-plugin</artifac
        <version>3.0.2</version>
      </plugin>
      <plugin>
        <artifactId>maven-compiler-plugin</artifact
```