



Osnove web programiranja

Maven, Spring

Termin 2

Sadržaj

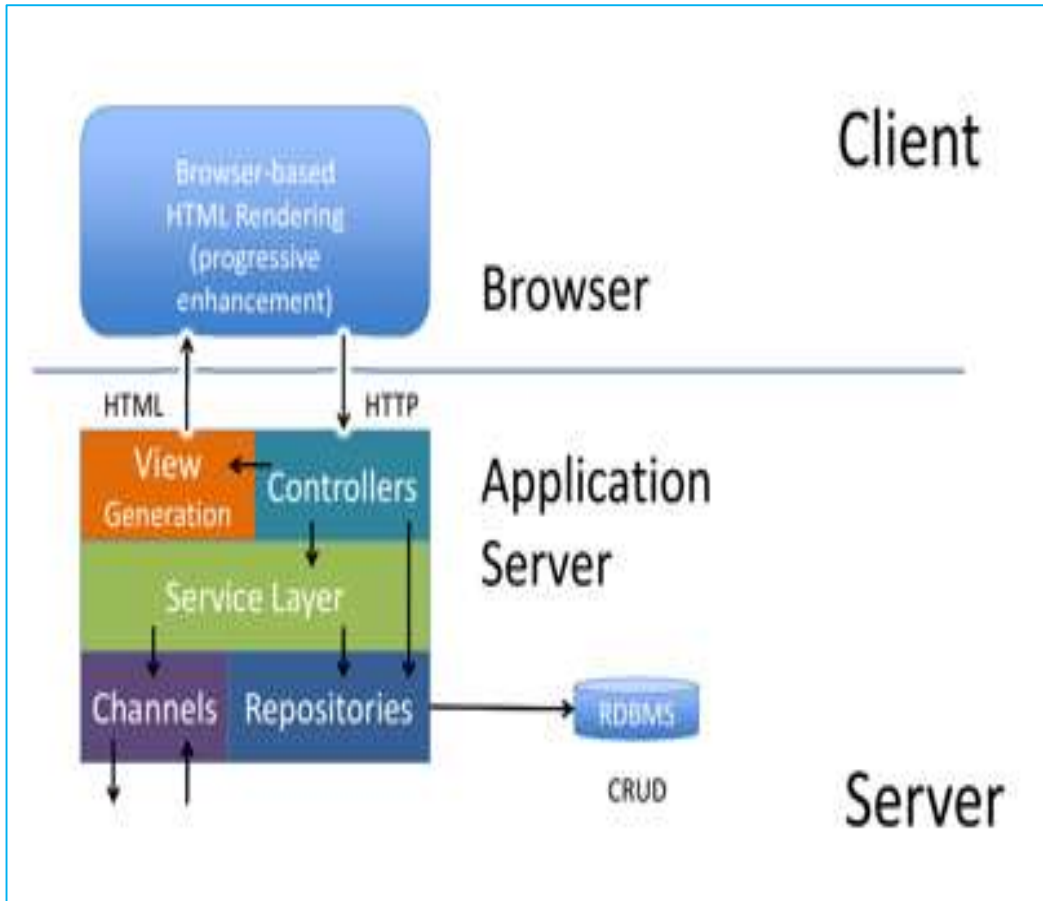
1. Arhitektura veb aplikacija
2. Spring uvod
3. Maven projekti
4. Maven Veb projekti
5. Kreiranje novog Maven veb projekta

Dodatno:

1. Inversion of control (IoC)
2. Convention over Configuration
3. Maven Dependency management

Arhitektura veb aplikacija

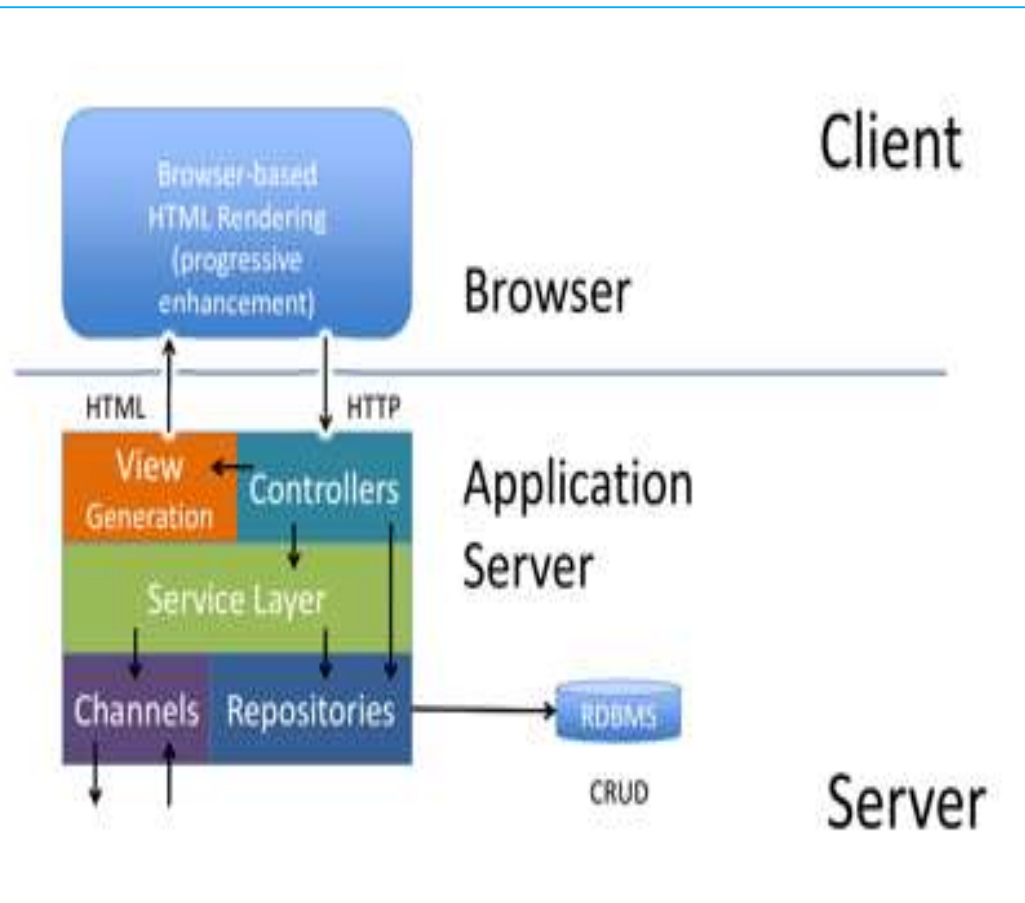
MVC



- Klijent – Server aplikacija, gde komuniciraju po HTTP protokolu
- Klijent - brauzerska aplikacija, šalje zahtev za resursom i po dobijanju odgovora prikazuje dobijeni resurs od Servera
- Server prihvata HTTP zahteve, obrađuje ih i vraća HTTP odgovor
- klasična veb aplikacija
- MVC aplikacija u kojoj se HTML kod generiše na serverskoj stani i isporučuje klijentu
- HTML generisana stanice ne sadrži Java kod
- MVC – Model View Controler, Univerzalan šablon (ne koristi se samo u web aplikacijama)
- Design pattern koji se zasniva na sinhronom radu tri komponente: Model, View i Controller

Arhitektura veb aplikacija

MVC-Scenario upotrebe



- Server na osnovu URL zaključuje koji to resurs korisnik traži i poziva odgovarajući kontroler
- Kontroler izvršava akciju za traženi resurs. Obradjuje HTTP zahtev, izvršava neku poslovnu logiku, inicira obradu podataka, koristi razne servise da bi dobavio neke podatke iz baze ili nešto ažurirao u bazi, kontroler popunjava podatke u Model.
- Kontroler zatim Model proseđuje Pogledu koji će dinamički generisti HTML stranicu, a Server će tako tako generisanu HTML stranicu vratiti Klijentu

Arhitektura veb aplikacija

MVC sa Servletima

- Ako bi se MVC veb aplikacija kompletno razvijala u Servlet tehnologiji. Servleti su bili kontroleri, a pogled su bili JSP stranice.
- Korišćenjem Servlet tehnologije se **standardna Java aplikacija proširivala podrškom za Servlete** tako da se dobije veb aplikacija.
 - Neophodno je bilo **ubaciti dodatne Java biblioteke** i implementirati odgovarjuće Servlet Java klase
- Cilje je da se MVC veb aplikacija razvije **korišćenjem Spring framework** (radni okvir)
- Koja je razlika između korišćenja biblioteka iz prvog slučaja i korišćenja radnog okvira?

Arhitektura veb aplikacija

MVC sa Servletima Vs MVC sa Spring radnim okvirom

- Koja je razlika između korišćenja biblioteka iz prvog slučaja i korišćenja radnog okvira?
- Prvi slučaj, biblioteka se ubacuje u **naš** projekat
 - na odgovarajućem mestu se poziva funkcionalnost te biblioteke, a ona vraća neki rezultat
 - **kontrola je na našoj strani, mi pozivamo biblioteku**
- Drugi slučaj, kada se koristi radni okvir
 - u projektu se programiraju komponente koje će se ugraditi u već predefinisanu aplikaciju (aplikacija kao da već postoji samo je treba doraditi)
 - **kontrola nije više na našoj strani, radni okvir poziva našu komponentu**
 - radni okvir implementira učestale mehanizme, nizove koraka, koji se uvek izvršavaju u aplikaciji.
 - kada se dođe do konkretnog koraka koji treba da se izvrši, tada je neophodno za radni okvir isprogramirati i u njemu ugraditi odgovarajuću komponentu, koja će izvršiti odgovarajuću funkcionalnost

Arhitektura veb aplikacija

Inversion of Control(IoC)

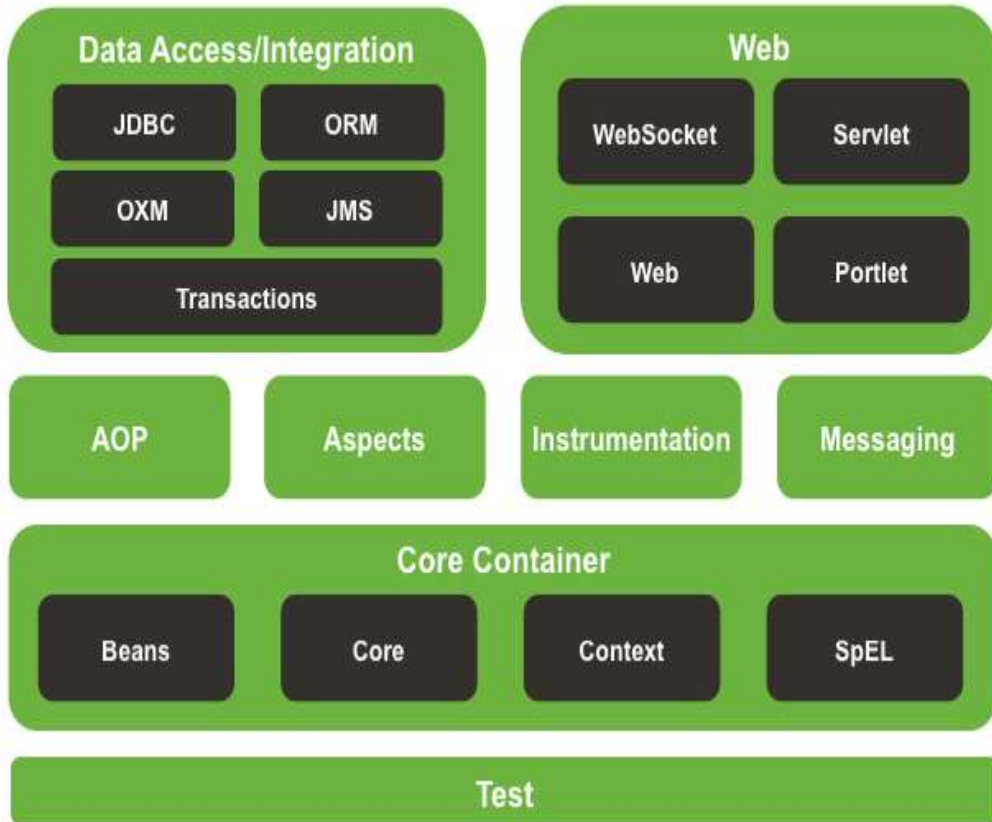
- Princip koji se koristi kod Spring radnog okvira, da programer kreira komponente koje Spring radni okvir poziva po potrebi zove se inverzija kontrole (**Inversion of Control**) ili može se čak i zvati **Hollywood Principle**
- U knjizi *Head First Design pattern* se **Hollywood Principle** princip objašnjava *With the Hollywood Principle, we allow low-level components to hook themselves into a system, but the high-level components determine when they are needed, and how. In other words, the high-level components give the low-level components a “don’t call us, we’ll call you” treatment.*

Spring uvod

Moduli radnog okvira



Spring Framework Runtime



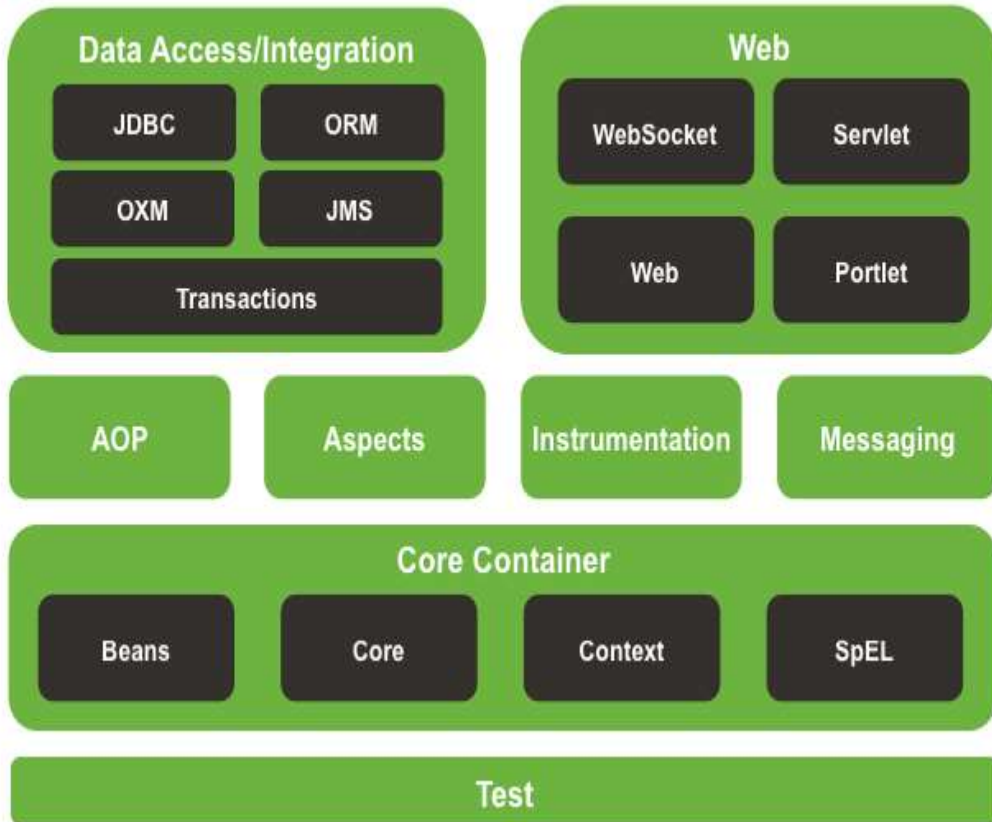
- Spring ima široko polje primene. Koristi se za izradu: veb aplikacija (klasična MVC arhitektura ili nova REST arhitektura), desktop aplikacija, mobilnih aplikacija, itd.
- Spring je radni okvir za izradu Java aplikacija
- Spring je podeljen na module, svaki modul zadužen za neku funkcionalnost.
- Spring Core modul implementira osnovne mehanizme rada Spring
- Razvoj Spring aplikacija se svodi na korišćenje Spring Core modula i onih modula koji su potrebni za konkretan tip aplikacije čija je funkcionalnost neophodno implementirati

Spring uvod

Moduli radnog okvira



Spring Framework Runtime



- Spring veb aplikacije zahtevaju uključenje modula Web.
- Ako je potreba perzistencija podataka uljučuje se Data Access/Integration modul.
- Više o Springu na <https://docs.spring.io/spring/docs/5.0.0.RC2/spring-framework-reference/overview.html>
- Za kreiranje Spring aplikacija sve biblioteke za odabrane module se mogu uvezati ručno, ali to nije praktično

Maven projekti

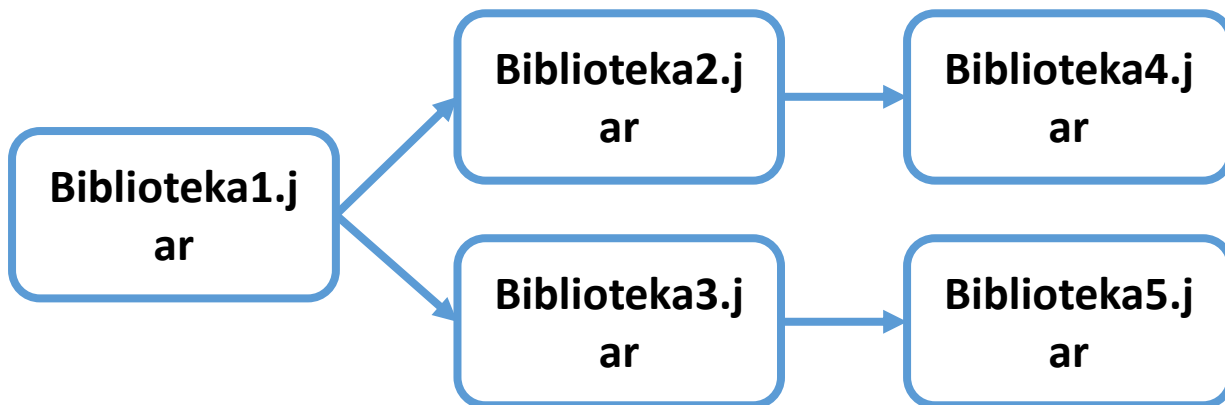
Uvod

- 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
 - nazavan *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 projekti

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 projekti

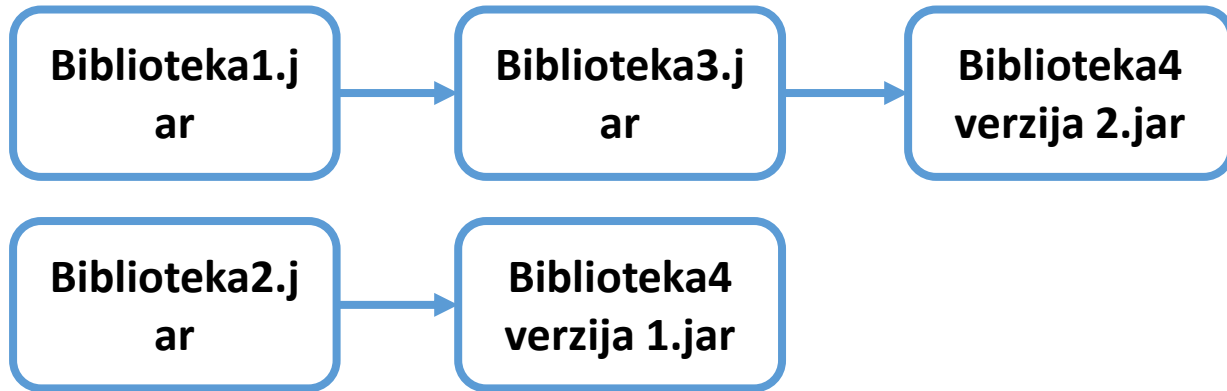
Dependency management - sa Maven

- Maven omogućava *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 biblioteka zavisi, i ponoviti isti process za novopreuzete biblioteke, sve dok više ne bude zavisnosti.
- Proces ponavljati sve dok više nema zavisnih biblioteka

Maven projekti

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 projekti

Životni ciklus izgradnje veb aplikacije

- Prethodno kada se razvijala veb aplikacija 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 predstavljaju ***Build life-cycle*** veb aplikacije
- Glavni zadatak 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 projekti

Životni ciklus izgradnje veb aplikacije

- Mogu se izvršiti svi koraci ili se može izvršiti do određenog koraka.
- Posle svako 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 testovi
 - *install* – instalacija paketa u lokalni repozitorijum
 - *deploy* – instalacija paketa u udaljeni repozitorijum
- Zašto su neophodni svi ovi koraci?

Maven projekti

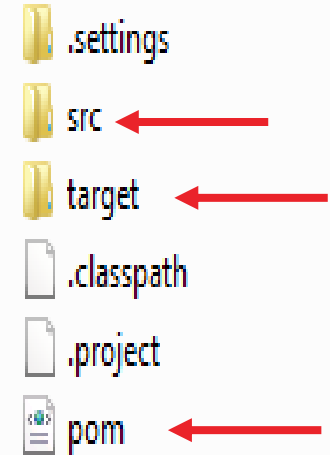
Convention over Configuration

- 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 eksplicitno, 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 projekti

Predefinisani prostorni raspored foldera

- 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 Veb projekti

Predefinisani prostorni raspored foldera

- 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, .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

Kreiranje novog Maven veb projekta

Java EE perspectiva

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

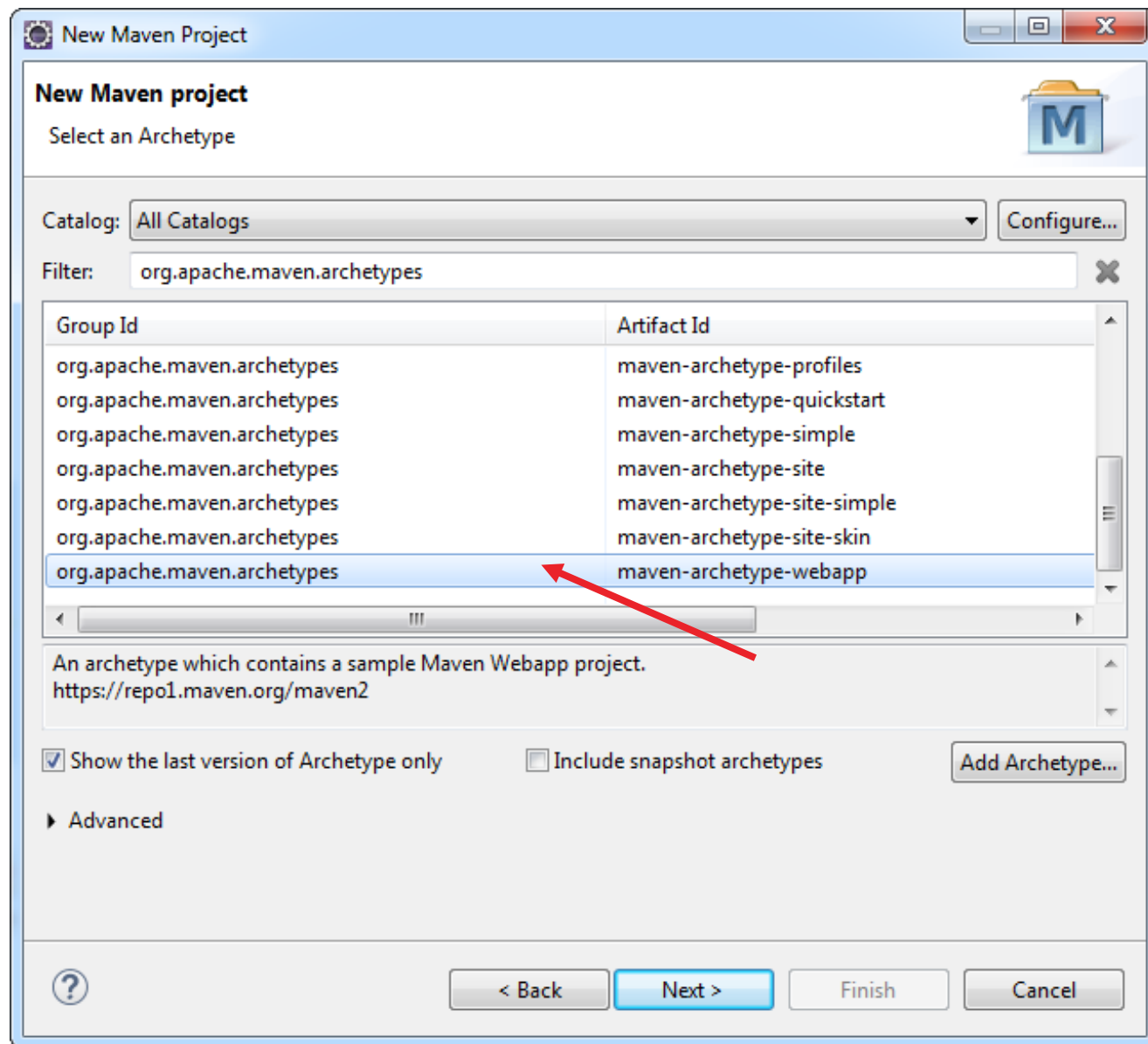


**Primer01-Kreiranje novog
Maven veb projekta**

Kreiranje novog Maven veb projekta

Java EE perspektiva

- 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*.



Kreiranje novog Maven veb projekta

Group Id i Artifact Id

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

Kreiranje novog Maven veb projekta

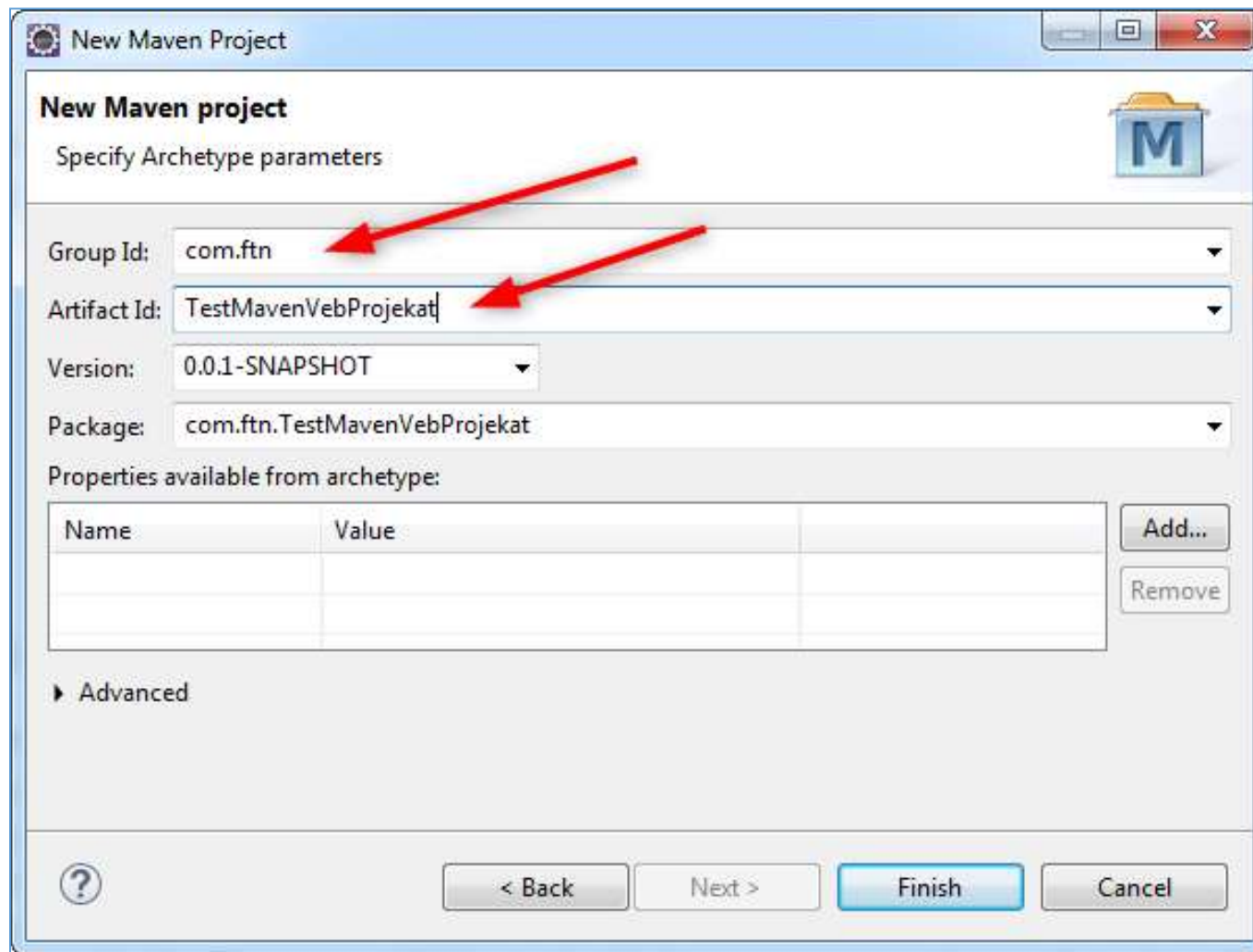
Java EE perspectiva

- 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 predstavljaju druge Maven artifakte**

Kreiranje novog Maven veb projekta

Java EE perspectiva

- Za *Group Id* unesite *com.ftn*
- 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*.



New Maven Project

Specify Archetype parameters

Group Id:

Artifact Id:

Version:

Package:

Properties available from archetype:

Name	Value

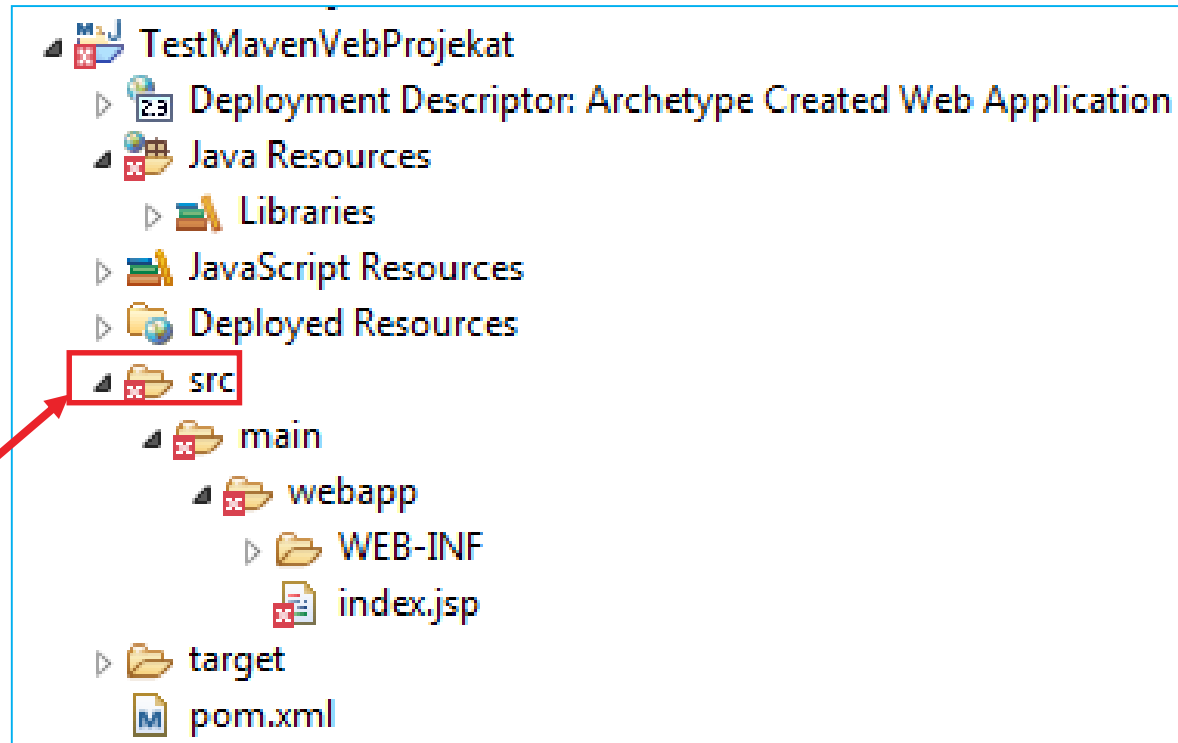
Advanced

< Back Next > Finish Cancel

Kreiranje novog Maven veb projekta

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

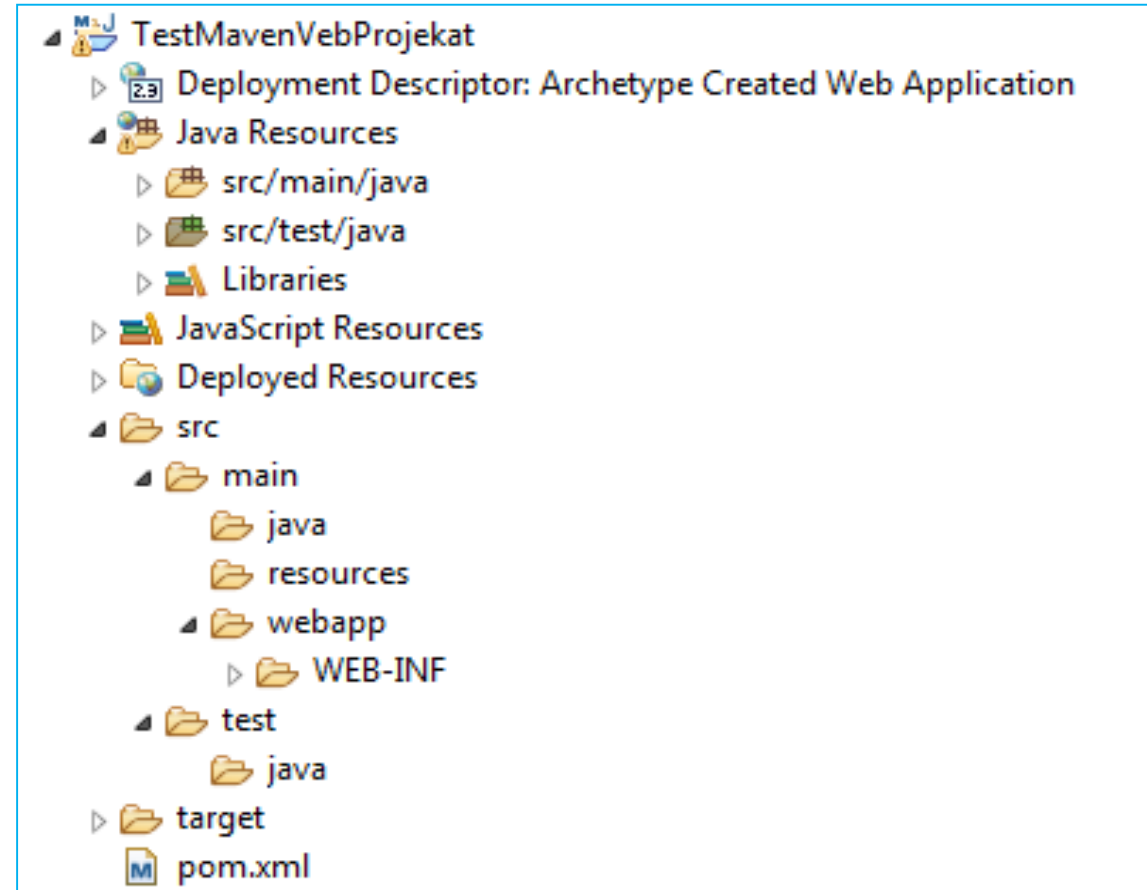


- **src/main/java**
- **src/main/resources**
- **src/test/java**

Kreiranje novog Maven veb projekta

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

Kreiranje novog Maven veb projekta

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

  <groupId>com.ftn</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>
```

Kreiranje novog Maven veb projekta

Objašnjenje 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 JUnit 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>
```

Kreiranje novog Maven veb projekta

Objašnjenje 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</artifactId>
      <version>3.0.2</version>
    </plugin>
    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.0</version>
    </plugin>
  </plugins>
</build>
```

Dodatno

Inversion of control

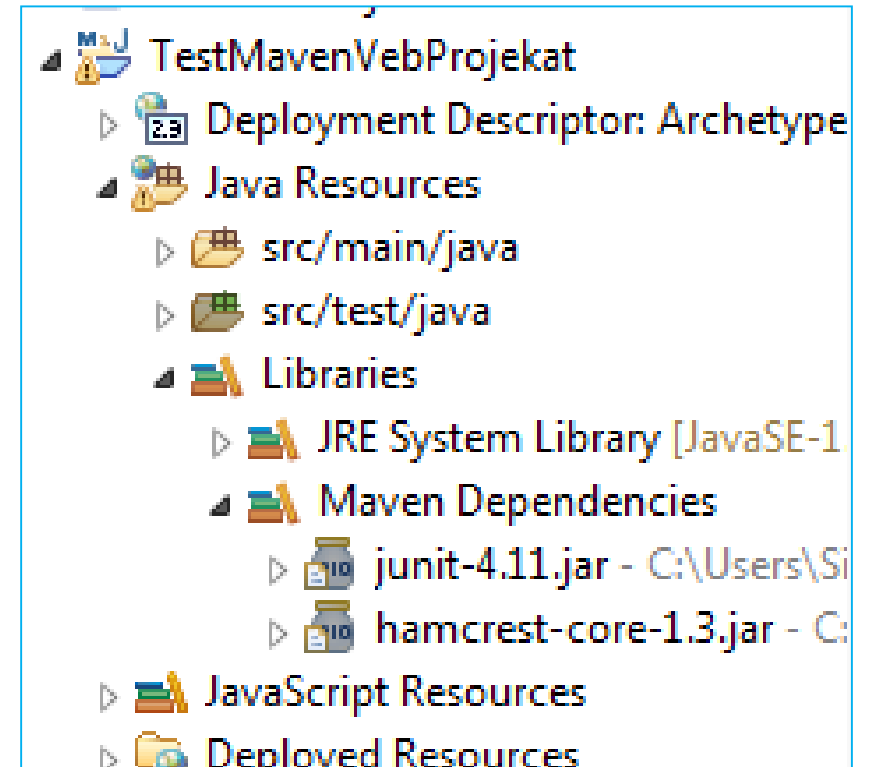
Inversion of control (IoC)

- According to the paper written by Martin Fowler(<https://martinfowler.com/bliki/InversionOfControl.html>), inversion of control is the principle where the control flow of a program is inverted: instead of the programmer controlling the flow of a program, the external sources (framework, services, other components) take control of it. It's like we plug something into something else.

Maven Dependency management

Maven Dependencies

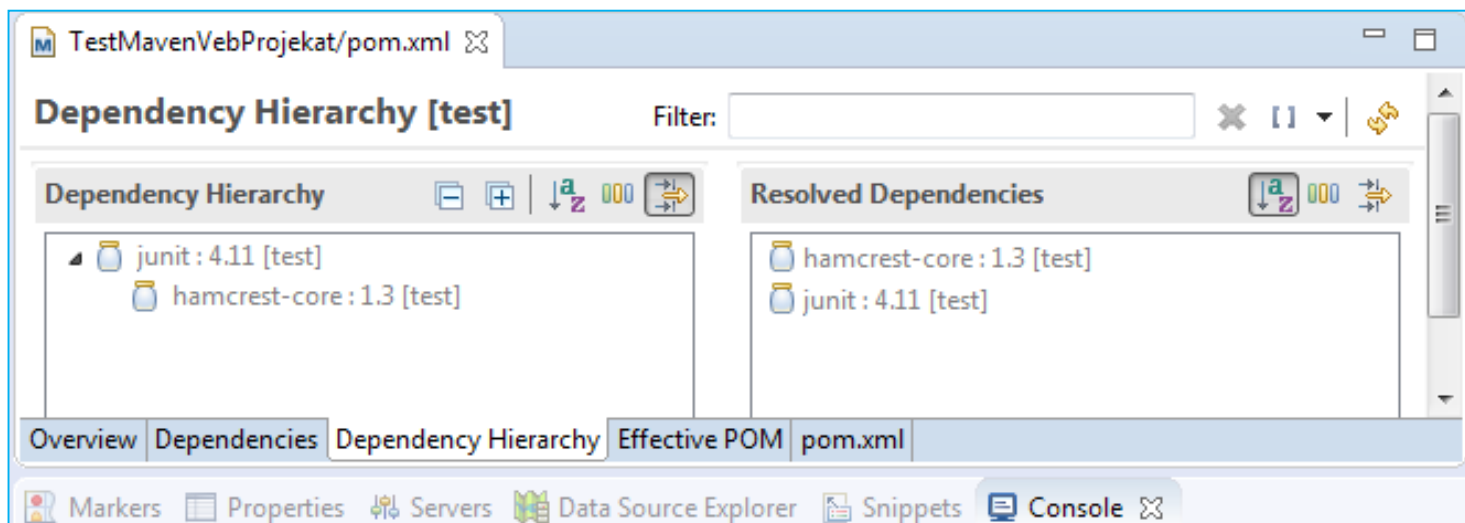
- Zavisnosti se mogu videti u okviru dela *Libraries->Maven Dependencies*
- Tu se može videti *junit* jar arhiva jer je ona navedena u pom.xml u tagu **<dependencies>**. Pored nje postoji i *hamcrest-core* jar arhiva koja je neophodna za rad JUnit biblioteke.
- Postojanje *hamcrest-core* jar arhive je dobar primer *Maven Dependency management* u akciji



Maven Dependency management

Effective POM i Dependency Hierarchy

- Pored taba *pom.xml* postoji i tab *Effective POM*. *Effective POM* se dobija kada se na definisani *pom.xml* dodaju pomovi svi projektnih zavisnosti, i od zavisnosti pomovi njihovih zavisnosti,... to spoji u jedan fajl.
- Tab *Dependency Hierarchy* na jednostavan način vizuelno prikazuje zavisnosti u projektu



Maven Dependency management

Greška zbog index.jsp

- U projektu je prethodno postojala greška jsp fala u *index.jsp*

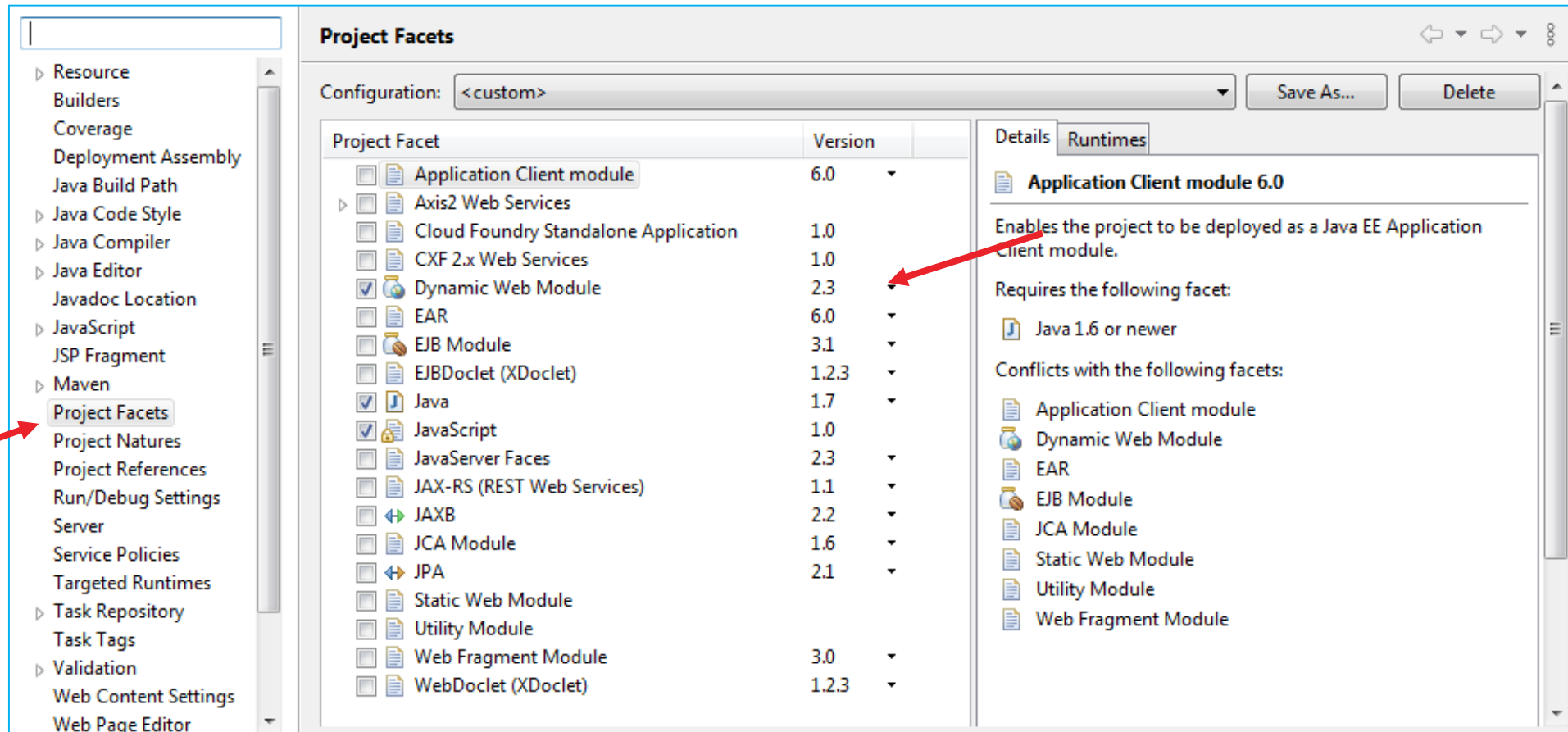
The superclass "javax.servlet.http.HttpServlet" was not found on the Java Build Path

- Da bi se greška uspešno rešila a da se **ne obriše** fajl *index.jsp* neophodno je u pom.xml fajlu navesti zavisnosti koje bi omogućile rad sa JSP tehnologijom.

Maven Dependency management

Greška zbog index.jsp

- Prvo proveriti u Project Facet podešavanje projekta (Desni klik na projekat *Properties->Project Facets->Dynamic Web Module*).



Maven Dependency management

Greška zbog index.jsp

- Zatim u tagu **<dependencies>** neophodno bi bilo dodati zavisnost od Servleta i JSP tehnologije, tako da ta zavisnost odgovara serverskoj specifikaciji navedenoj u Project Facet.
- Desni klik na projekat pa **Maven->Update Project ...**

```
<dependencies>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <version>2.3</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>javax.servlet.jsp</groupId>
    <artifactId>jsp-api</artifactId>
    <version>2.1</version>
    <scope>provided</scope>
  </dependency>
</dependencies>
```