Distribuirano Programiranje

Java Spark

Aleksandar Kartelj

Ukratko

1. Paralelna paradigma koristi konkurentna izračunavanja u cilju ubrzavanja i najčešće ima deljenu memoriju među jedinicama izračunavanja.
   - Da bi deljena memorija imala smisla (da bi bila dovoljno brza), fizičke udaljenosti među jedinicama izračunavanja (procesorima) moraju biti male što znaci da su one obično smeštene na jednom mestu (klasteri, mejnfrejmovi...)

2. Distribuirana paradigma koristi fizički "razbacane" jedinice izračunavanja i te jedinice nemaju deljenu memoriju u klasičnom smislu.
   - Jedinice (čvorovi, eng. nodes) su nezavisnije i izbegava se komunikacija osim kada se radi nekakvo sazimanje podataka.

- Jedinice su u distribuiranoj paradigmi obicno računari, a ne procesori kao sto je to slučaj kod paralelne.

Ukratko nastavak

- Model izračunavanja koji ćemo primenjivati je vid paralelizma nad podacima (Data paralelism) se može svesti na dve faze:

1. Svaki čvor primenjuje istu operaciju nad podskupom sveukupnih podataka
   (ovo liči na map funkciju višeg reda u Haskell-u)
2. Nakon što je operacija mapiranja primenjena radi se sažimanje rezultata među čvorovima
   (ovo liči na fold/reduce funkciju višeg reda u Haskell-u)

Neretko se ceo koncept ovakvog izračunavanja naziva **map-reduce**.

Radna platforma

- Radi se pod Java Spark platformom
- Moguća su dva režima izvršavanja:
  1. Lokalni u kojem jezgra procesora glume čvorove
  2. Pravi distribuirani u kojem se radi na mnogo većem broju čvorova: (zahteva povezivanje sa HDFS (Hadoop Distributed File System))
- Od ostalih stvari poželjna je Maven podrška za Eclipse
- Rad sa Lambda izrazima koji su podržani počev od Java verzije 8
- Cilj prvog tročasa je da svima proradi platforma!!!

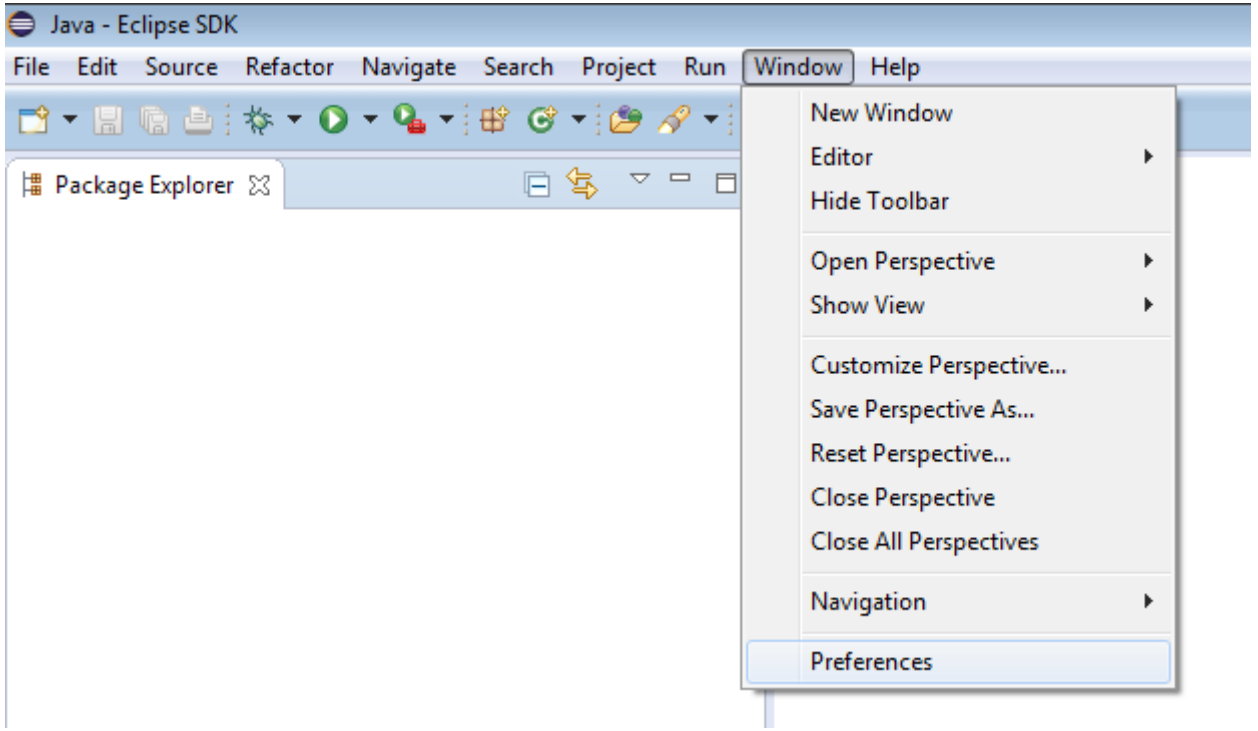
„Bolno“ podešavanje svega na fakultetskim računarima

- Da bismo podesili razvojno okruženje u Eclipse alatu potrebno je da prođemo kroz niz koraka:

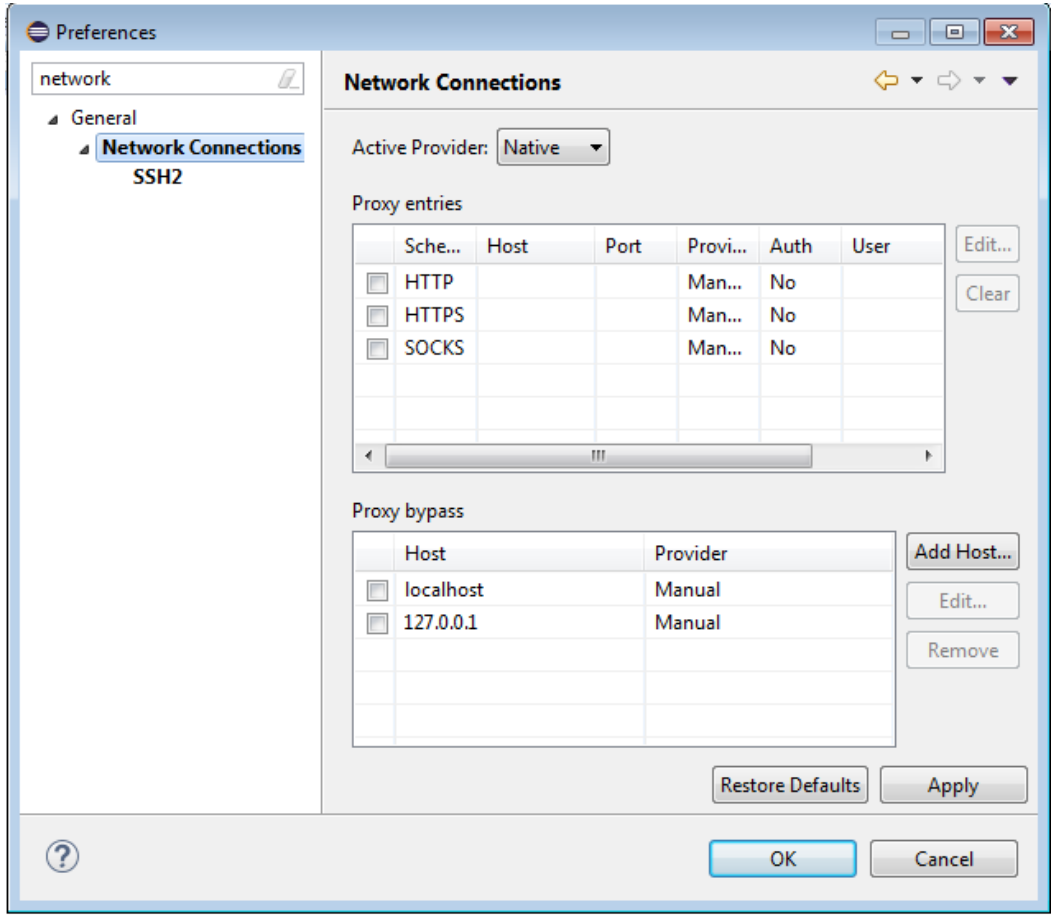
1. Eclipse podešavanje proxy-ja
2. Pravljenje maven projekta
   A. Pravljenje settting.xml ako ne postoji
   B. Postavljanje proxy-ja u settings.xml-u
   C. Postavljanje autoupdate podešavanja za maven
   D. Pravljenje projekta čarobnjak (wizard)
3. Dodavanje podrške za Spark u maven projekat
   A. Podešavanje verzije Jave
4. Izvršavanje jednostavnog programa u lokalnom okruženju

- Korak 1) i 2B) mogu biti preskočeni u slučaju da se ne radi na fakultetskim računarima odnosno ako ne postoji proxy.
  Korak 2A) može biti preskočen ako postoji settings.xml u maven repozitorijumu.
  3A) može biti preskočeno ako je već podešana verzija 8 Jave.

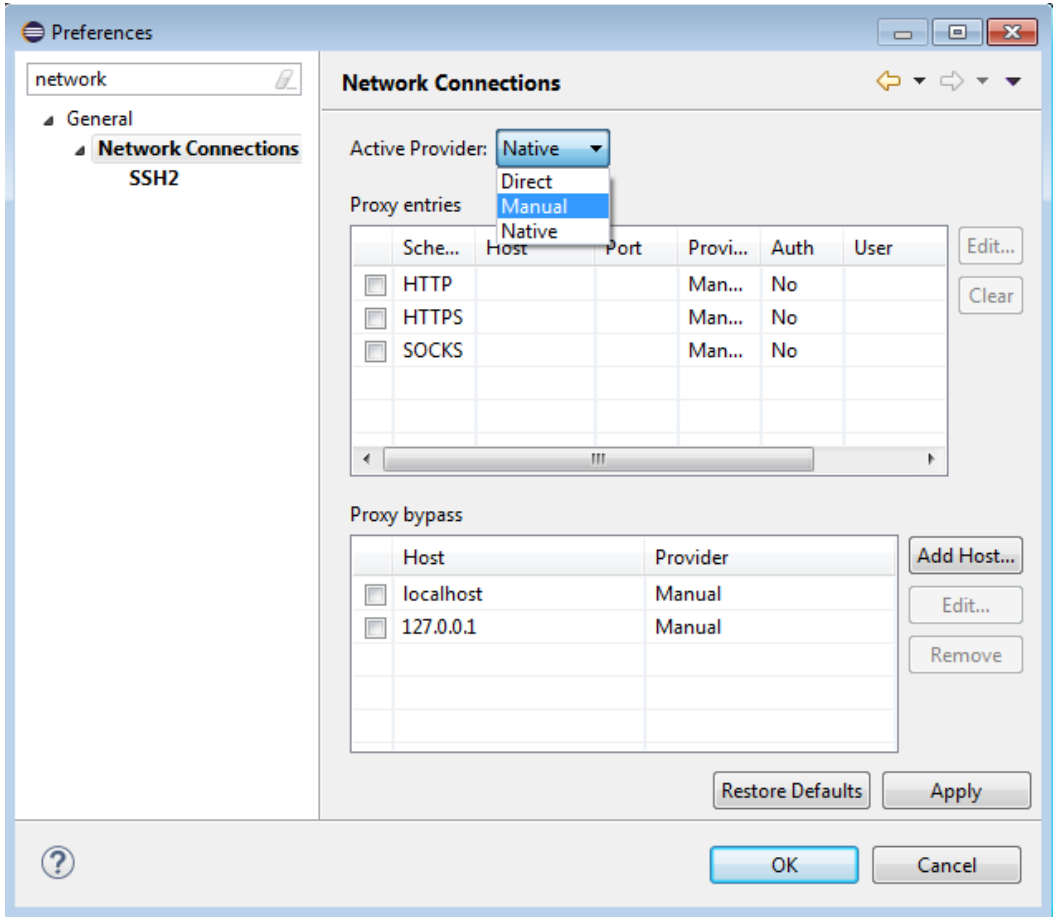
1. Eclipse podešavanje proxy-ja



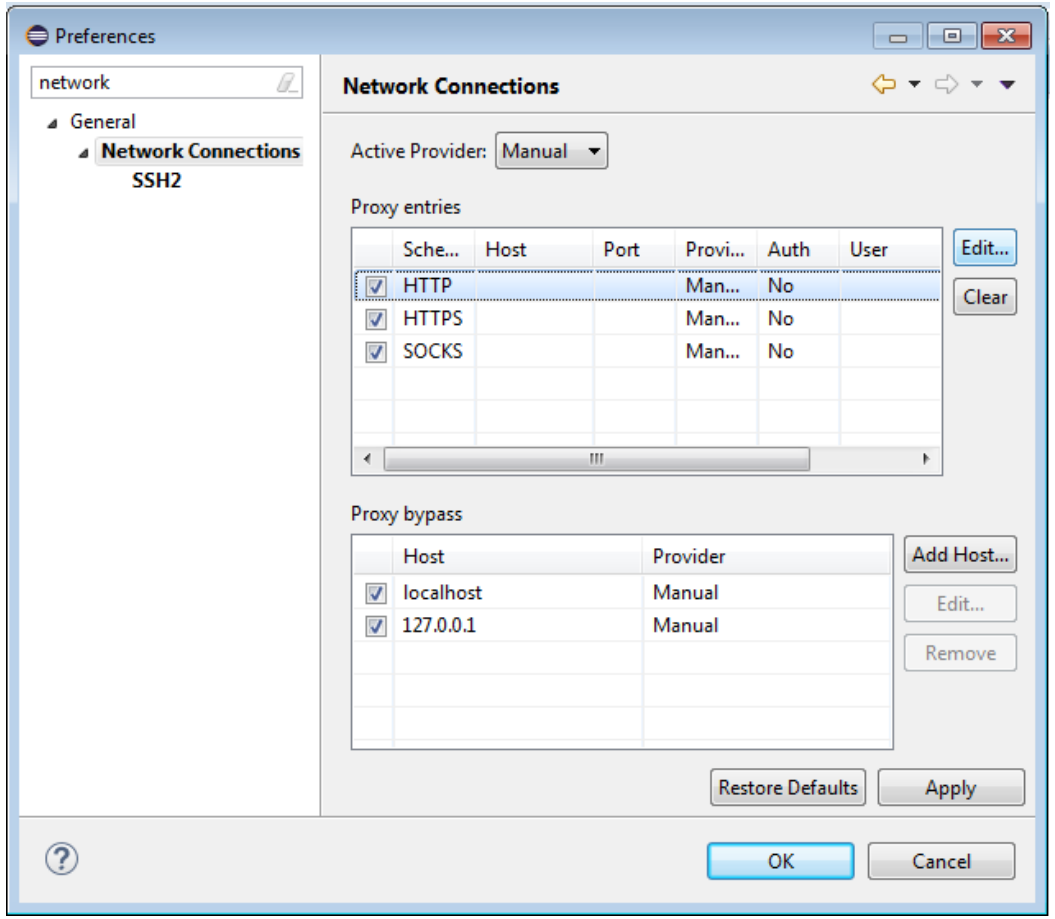
1. Eclipse podešavanje proxy-ja



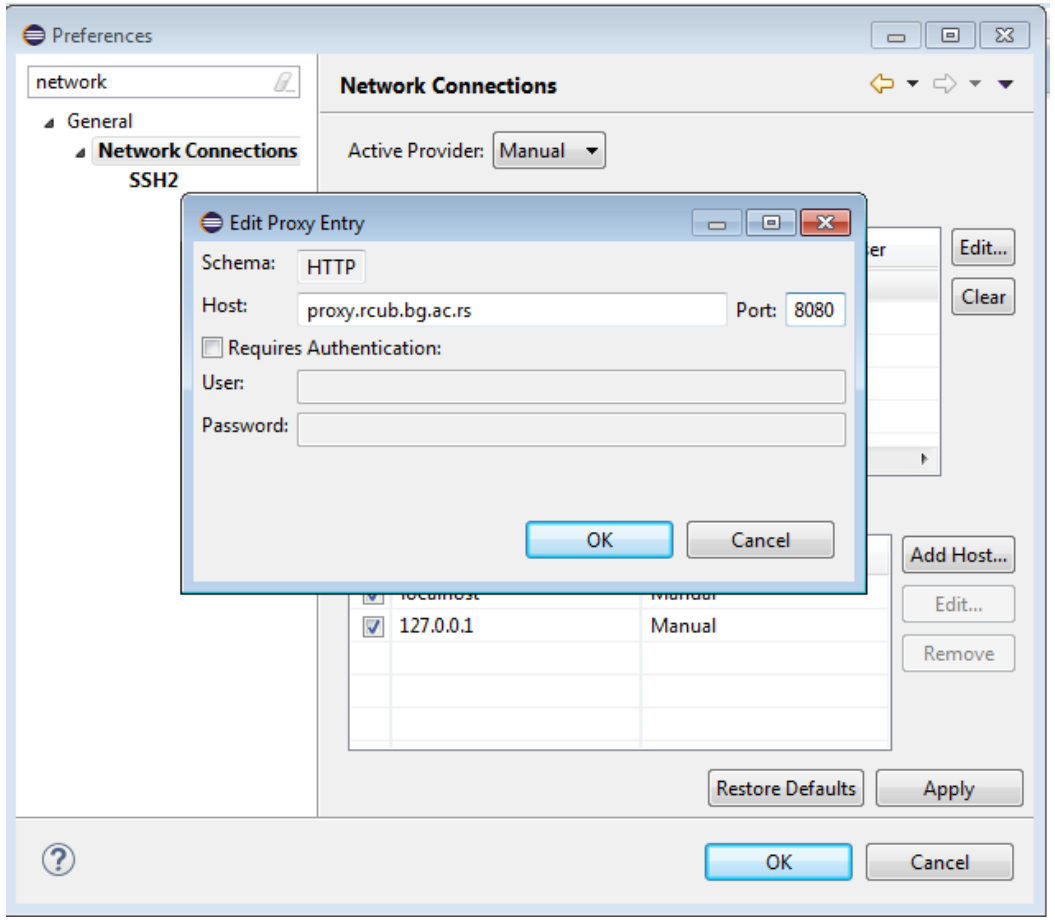
1. Eclipse podešavanje proxy-ja



1. Eclipse podešavanje proxy-ja



1. Eclipse podešavanje proxy-ja



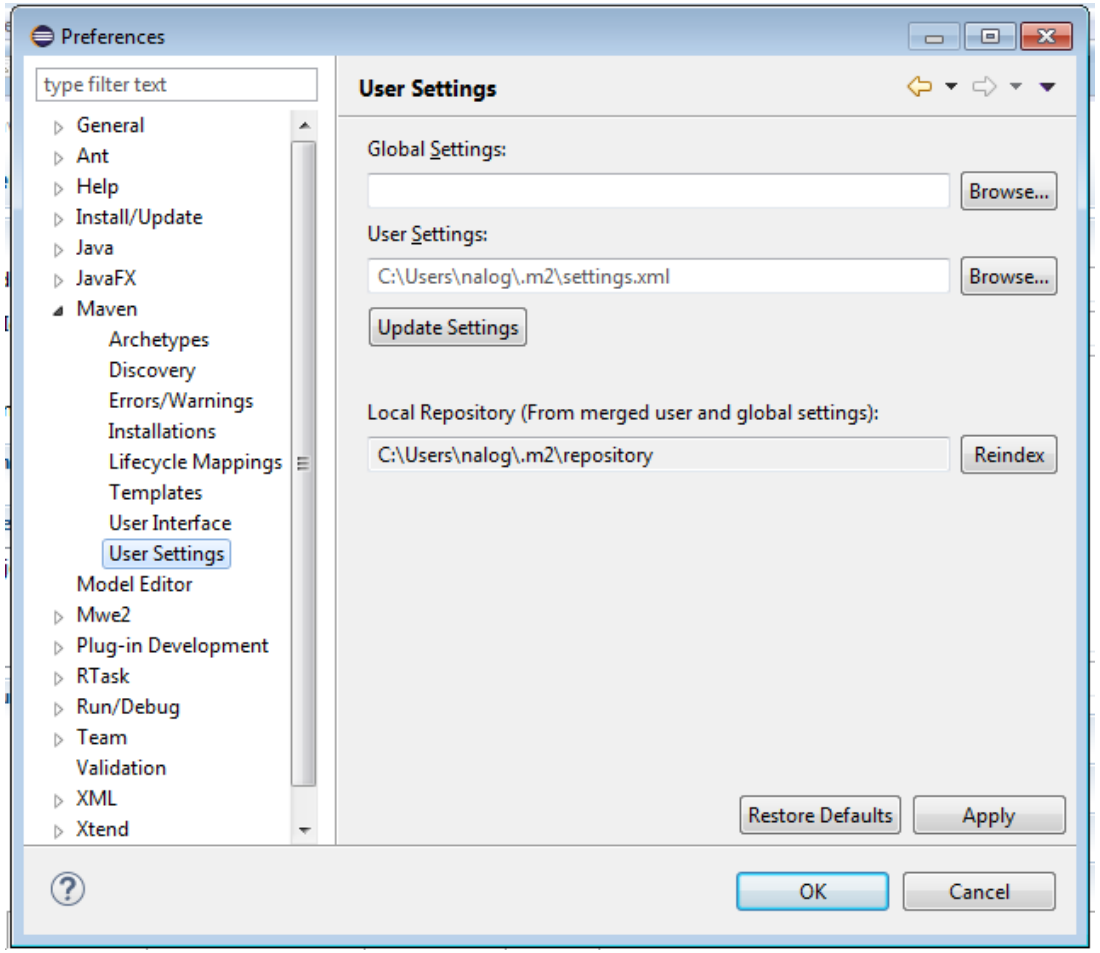
2. Pravljenje maven projekta

- Šta je Maven?
- Maven je alat koji se koristi pri razvoju aplikacija u cilju lakše integracije sa već postojećim bibliotekama klasa.
- Za nas su najbitnije sledeće funkcionalnosti:

1. Pravljenje gotovih šablona aplikacija
2. Ubacivanje potrebnih biblioteka za rad sa Spark-om i svih njihovih zavisnosti jednostavnim konfigurisanjem pom.xml datoteke

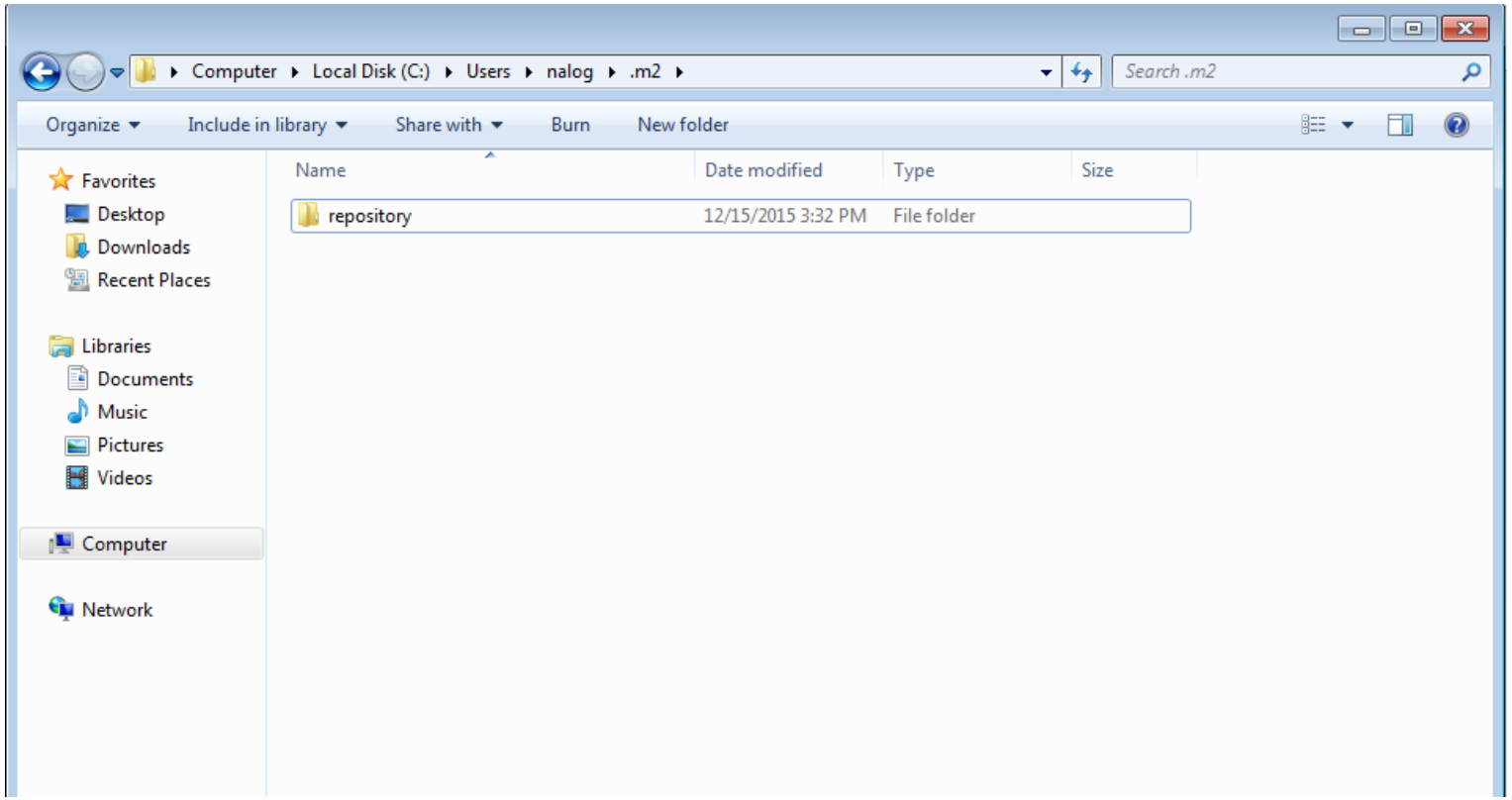
2A. Pravljenje maven projekta – Pravljenje settings.xml datoteka

- Lokacija settings.xml datoteke se vidi na slici.



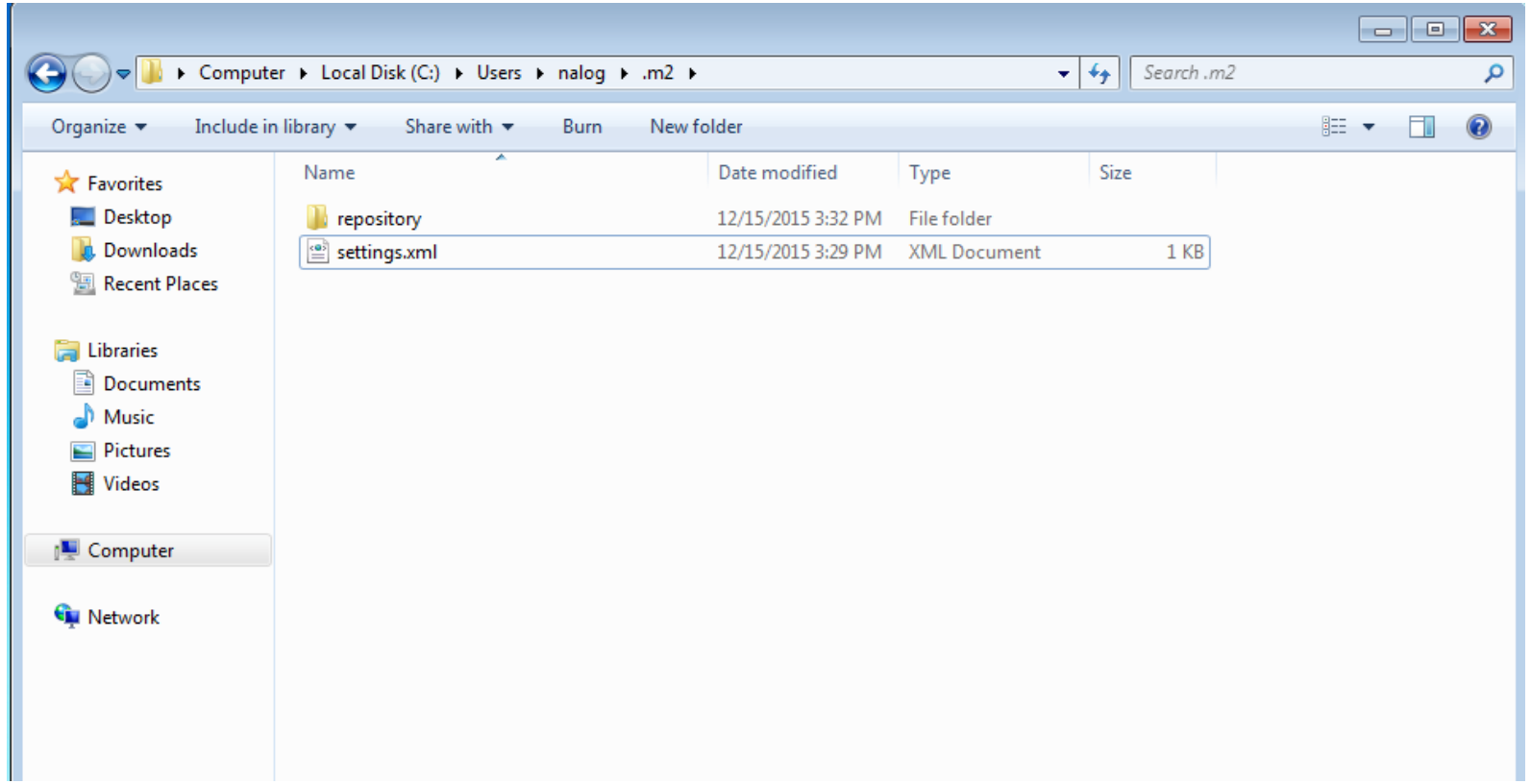
2A. Pravljenje maven projekta – Pravljenje settings.xml datoteka

- Ovo se radi samo ukoliko u korisničkom home direktorijumu ne postoji settings.xml u suprotnom preskočite 1A deo.



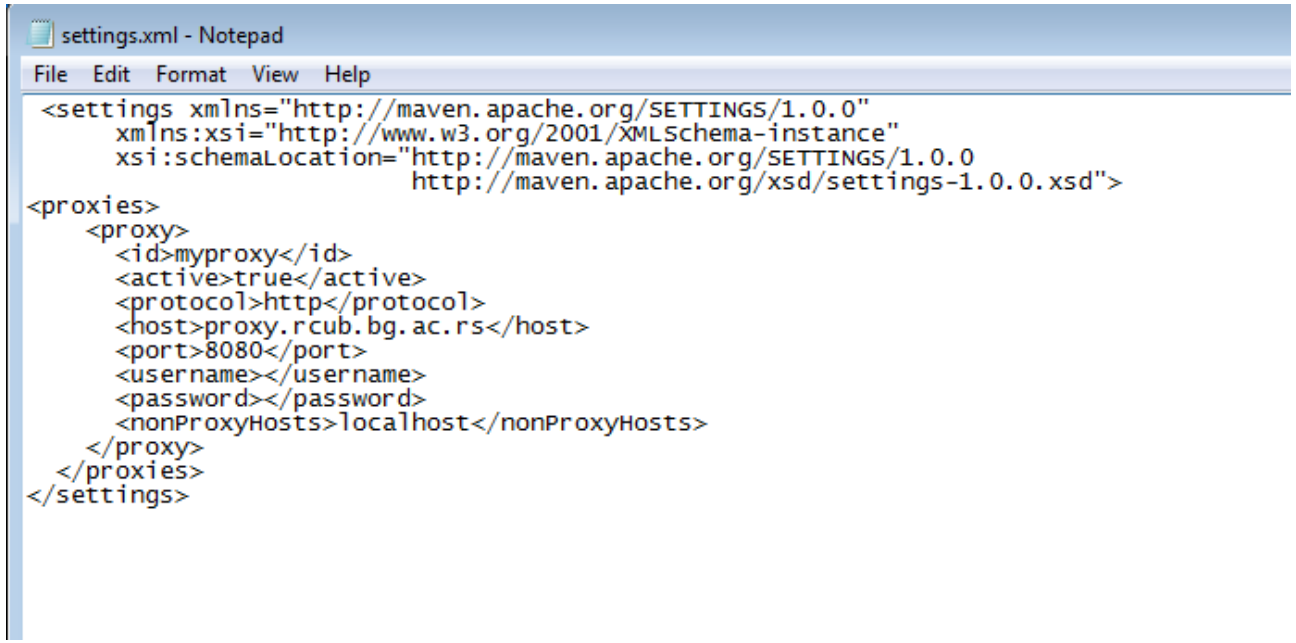
2A. Pravljenje maven projekta – Pravljenje settings.xml datoteka

- Napravite datoteku sa nazivom settings.xml kao na slici u .m2 direktorijumu.



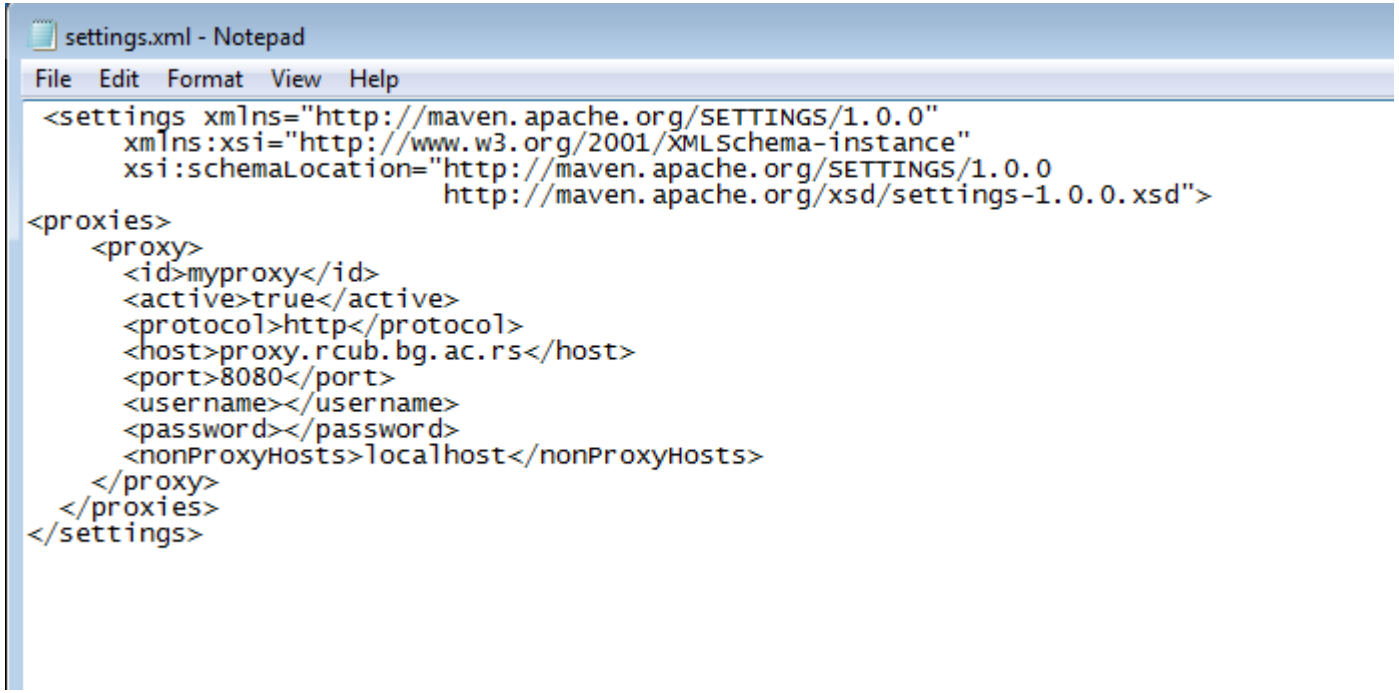
2A. Pravljenje maven projekta – Pravljenje settings.xml datoteka

- Otvorite ga tekstualnim editorom i dodajte sadržaj kao ispod samo bez proxies segmenta. Dakle samo otvoreni i zatvoreni settings tagovi.
- Ovako podešen settings.xml možete preuzeti odavde.



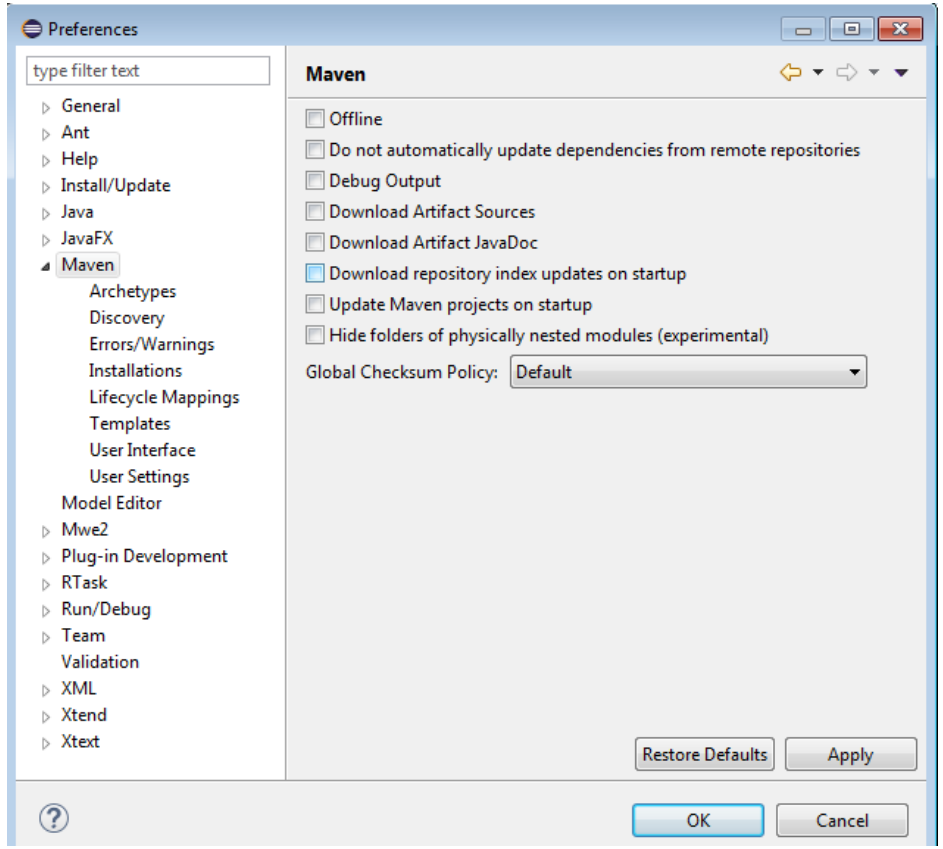
2B. Pravljenje maven projekta – Postavljanje proxy-ja u settings.xml-u

- Ako na Internet idete preko proxy-ja onda dodajte sav sadržaj koji je na slici. Naravno, umesto proxy-ja sa slike upišite adekvatne informacije.

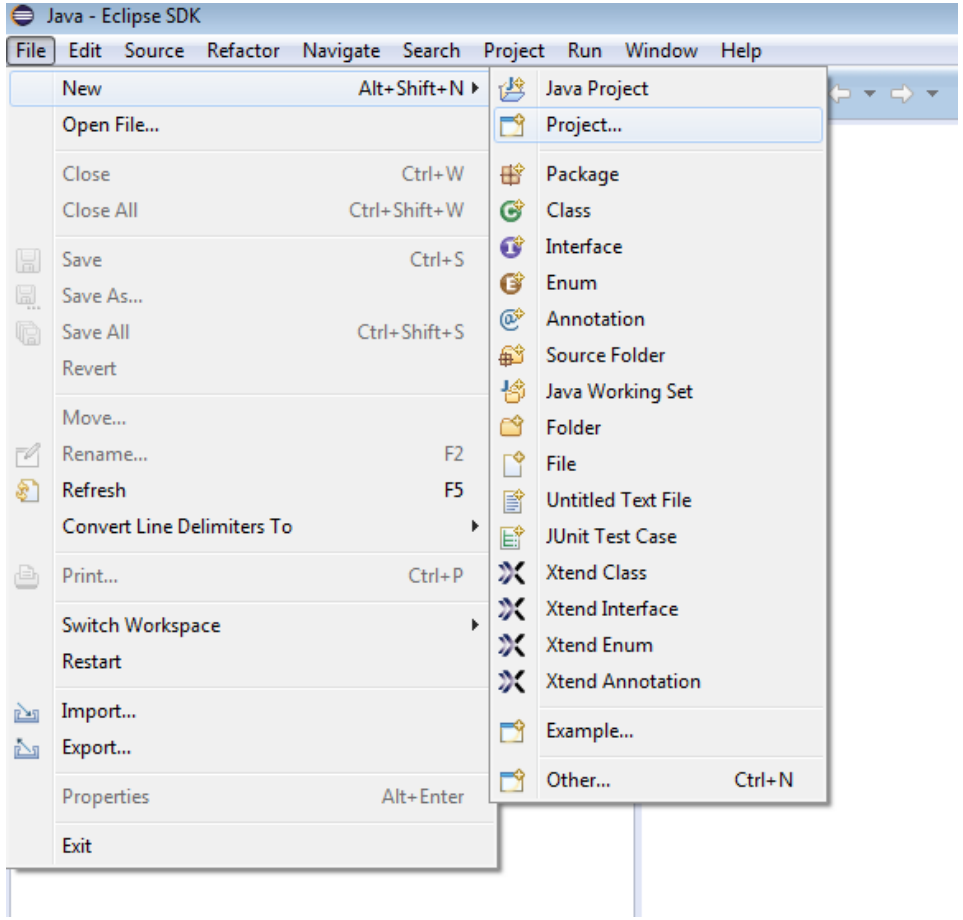


2C. Pravljenje maven projekta – Postavljanje autoupdate za maven

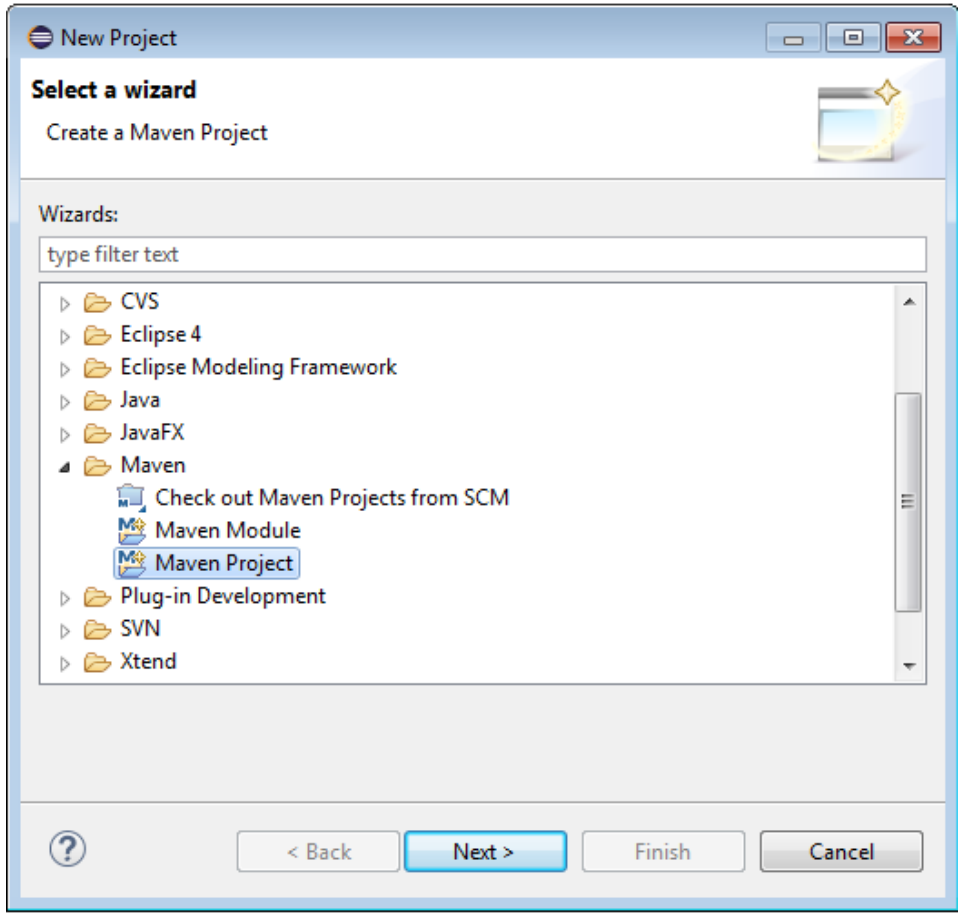
- Maven podešavanja u Window->Preferences treba da izgledaju ovako:



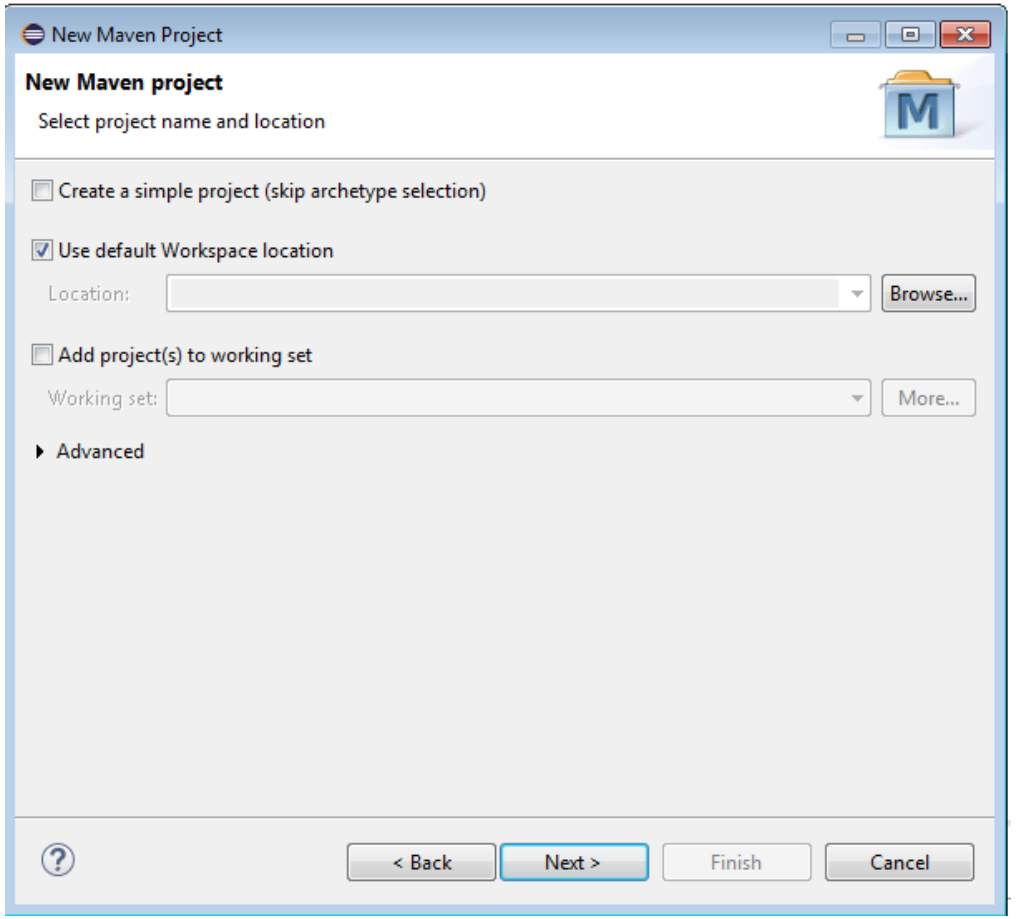
2D. Pravljenje maven projekta – čarobnjak (wizard)



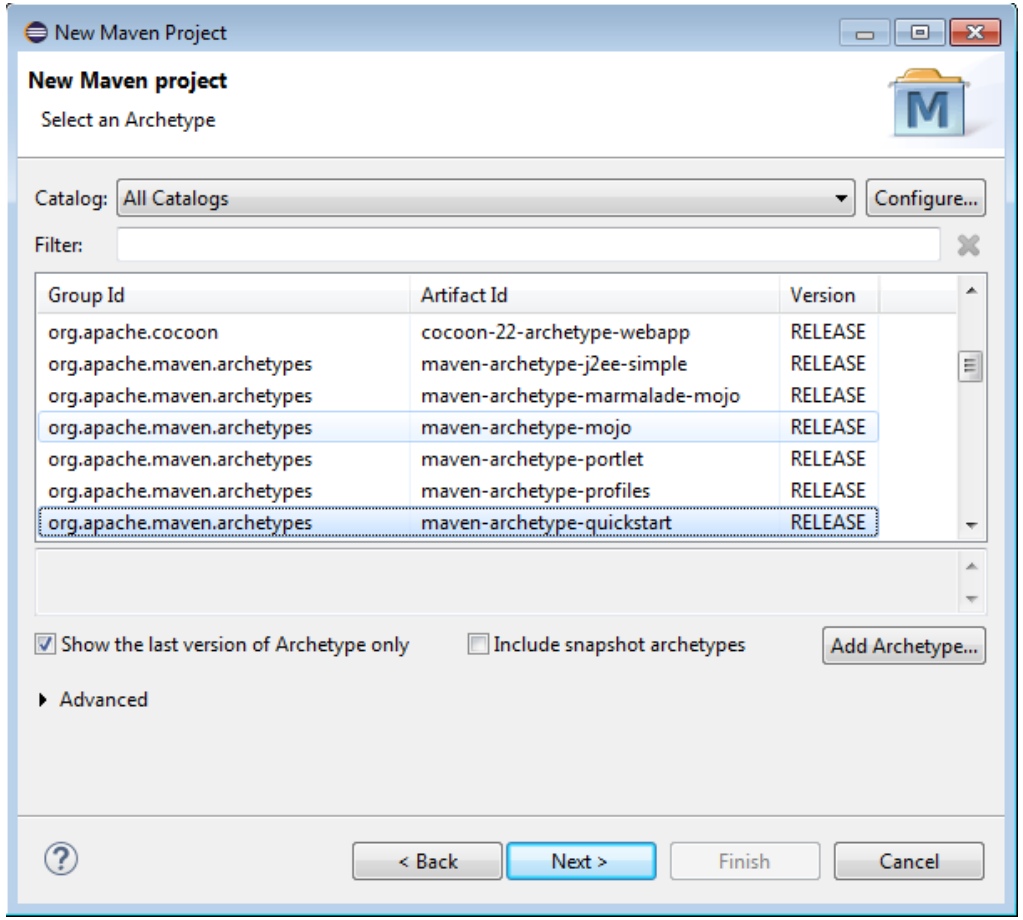
2D. Pravljenje maven projekta – čarobnjak (wizard)



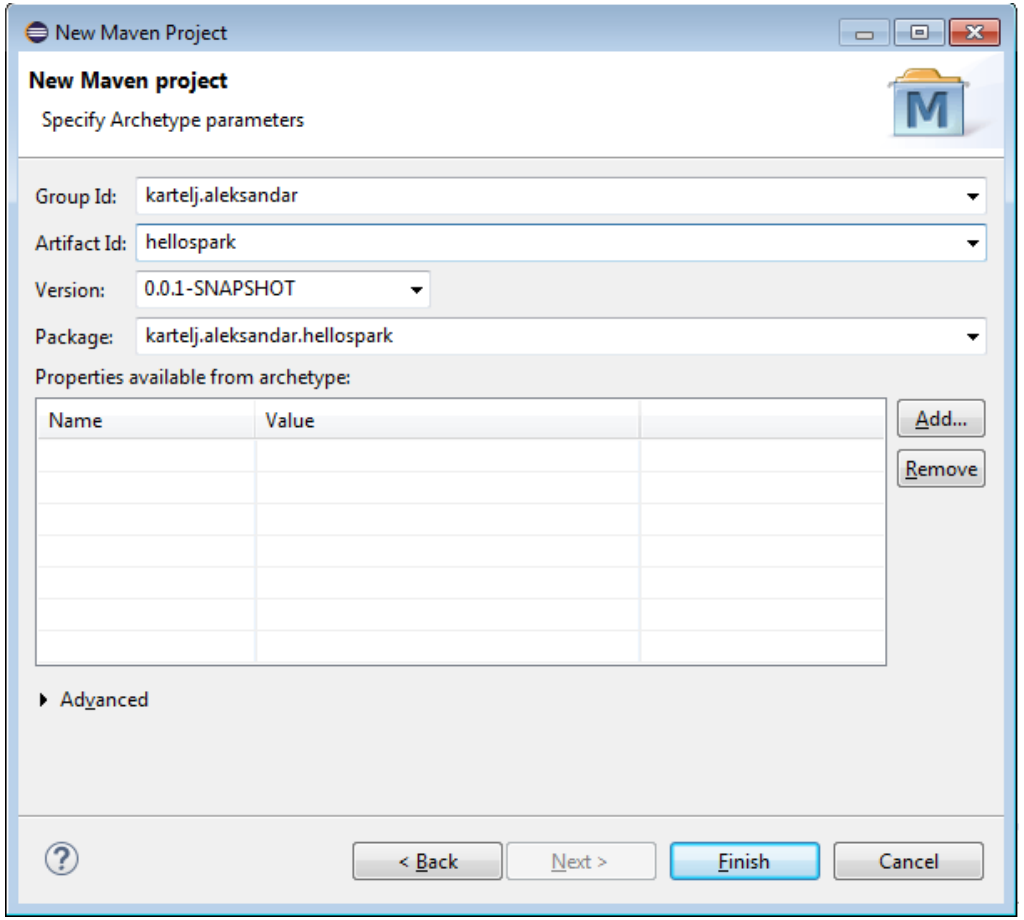
2D. Pravljenje maven projekta – čarobnjak (wizard)



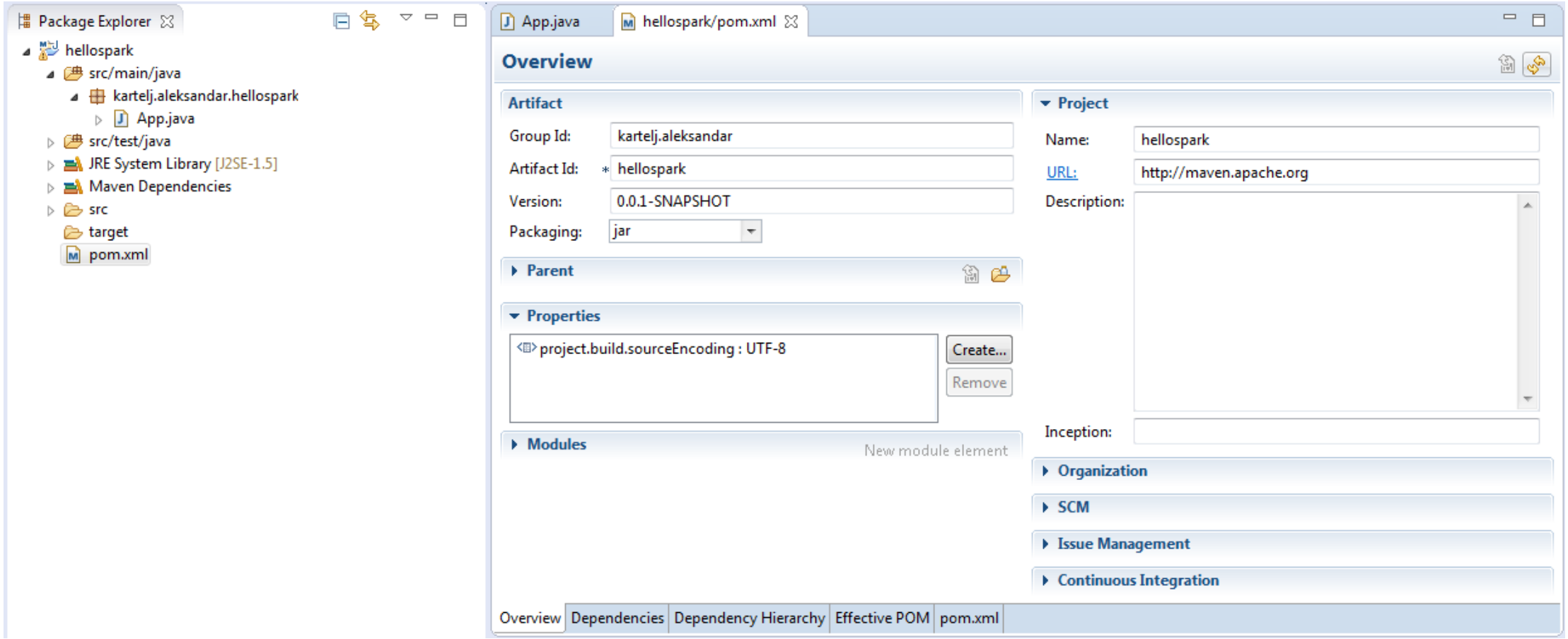
2D. Pravljenje maven projekta – čarobnjak (wizard)



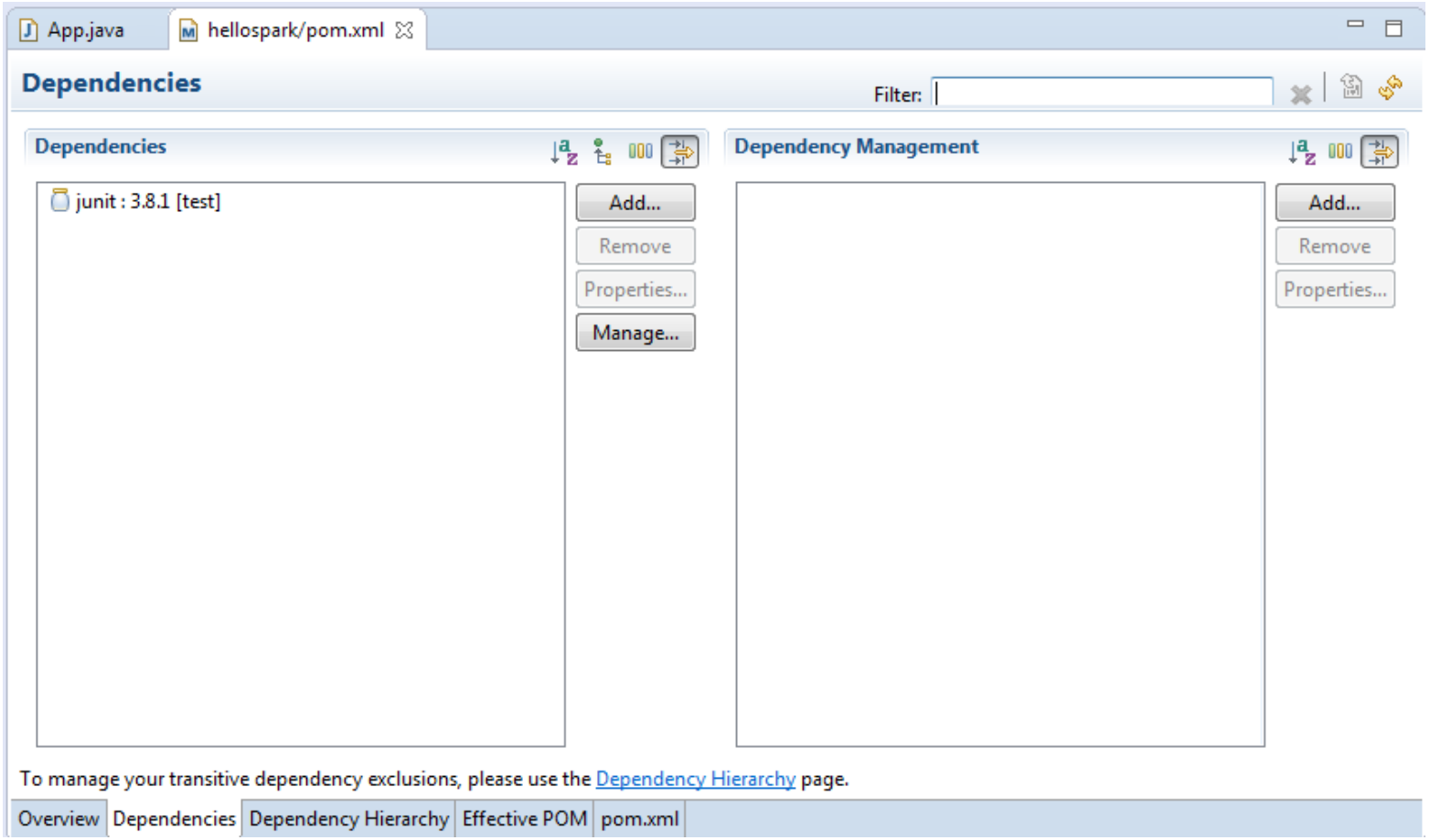
2D. Pravljenje maven projekta – čarobnjak (wizard)



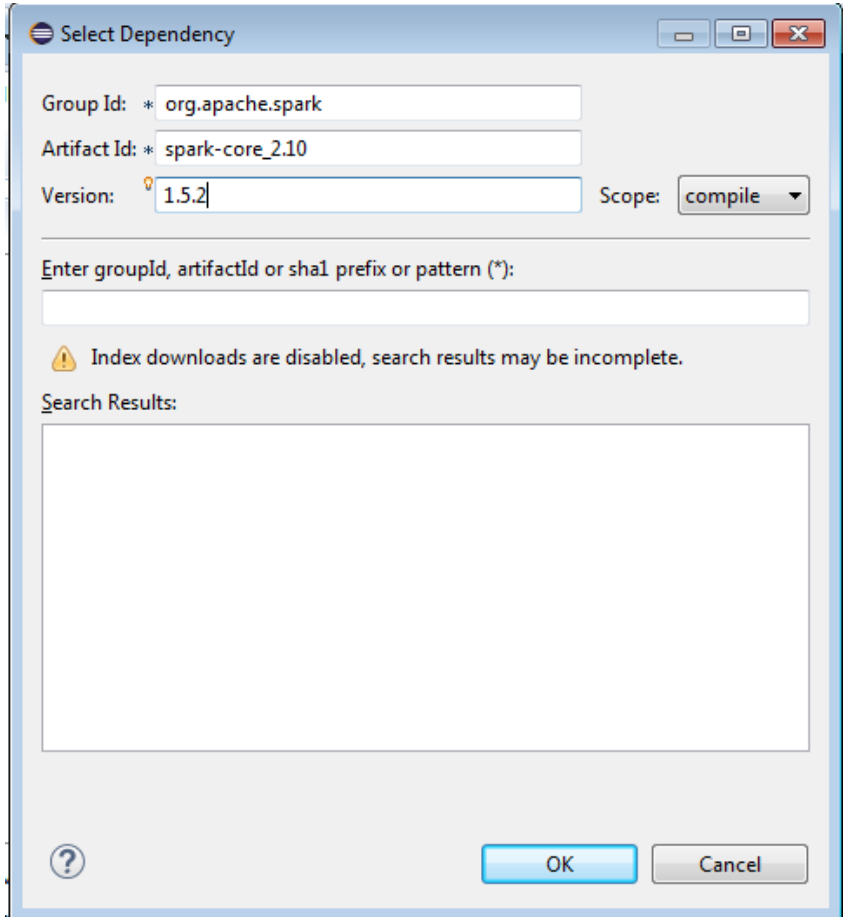
2D. Pravljenje maven projekta – Izgled projekta nakon pravljenja



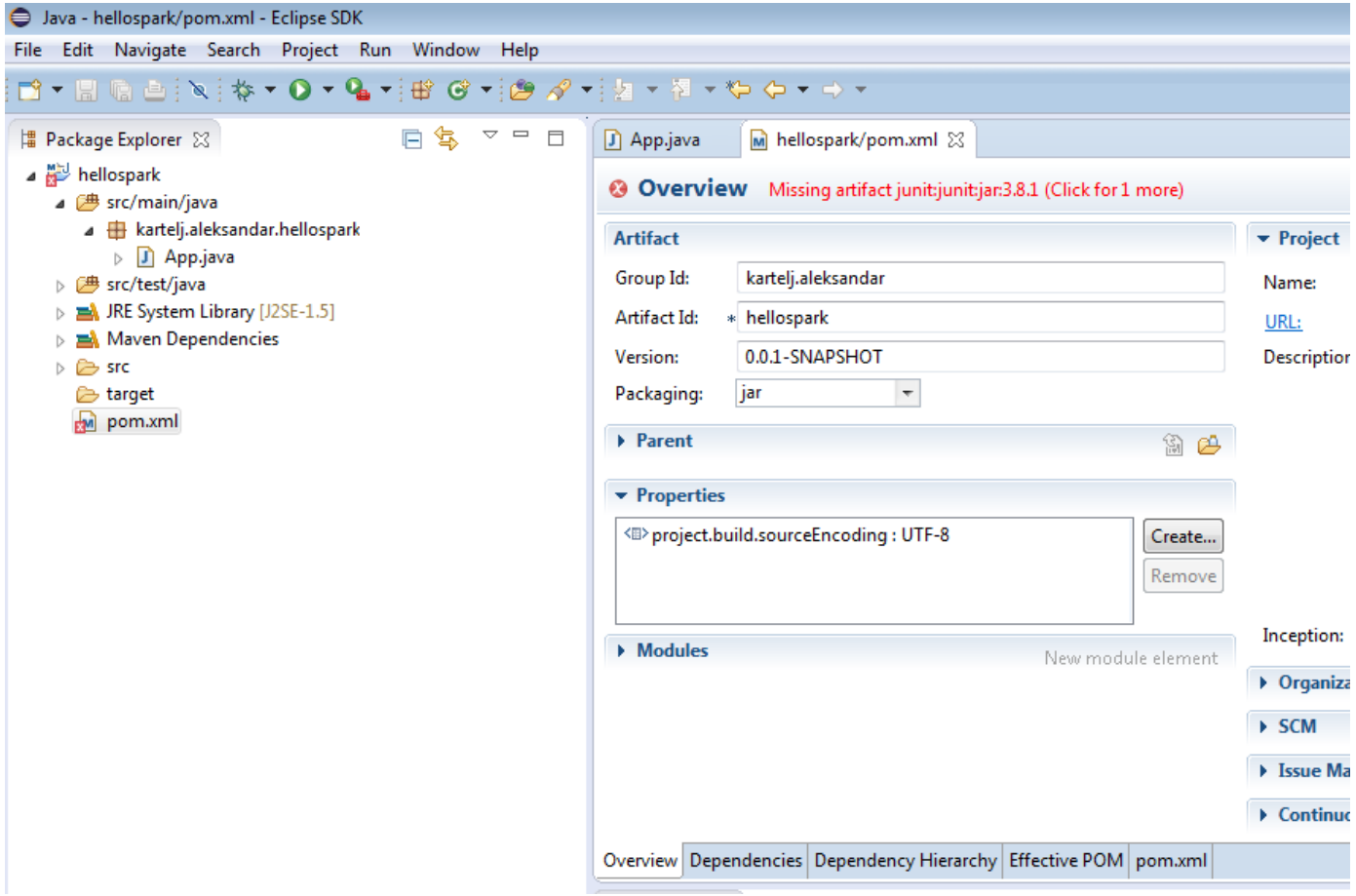
3. Dodavanje podrške za Spark u maven projekat



3. Dodavanje podrške za Spark u maven projekat

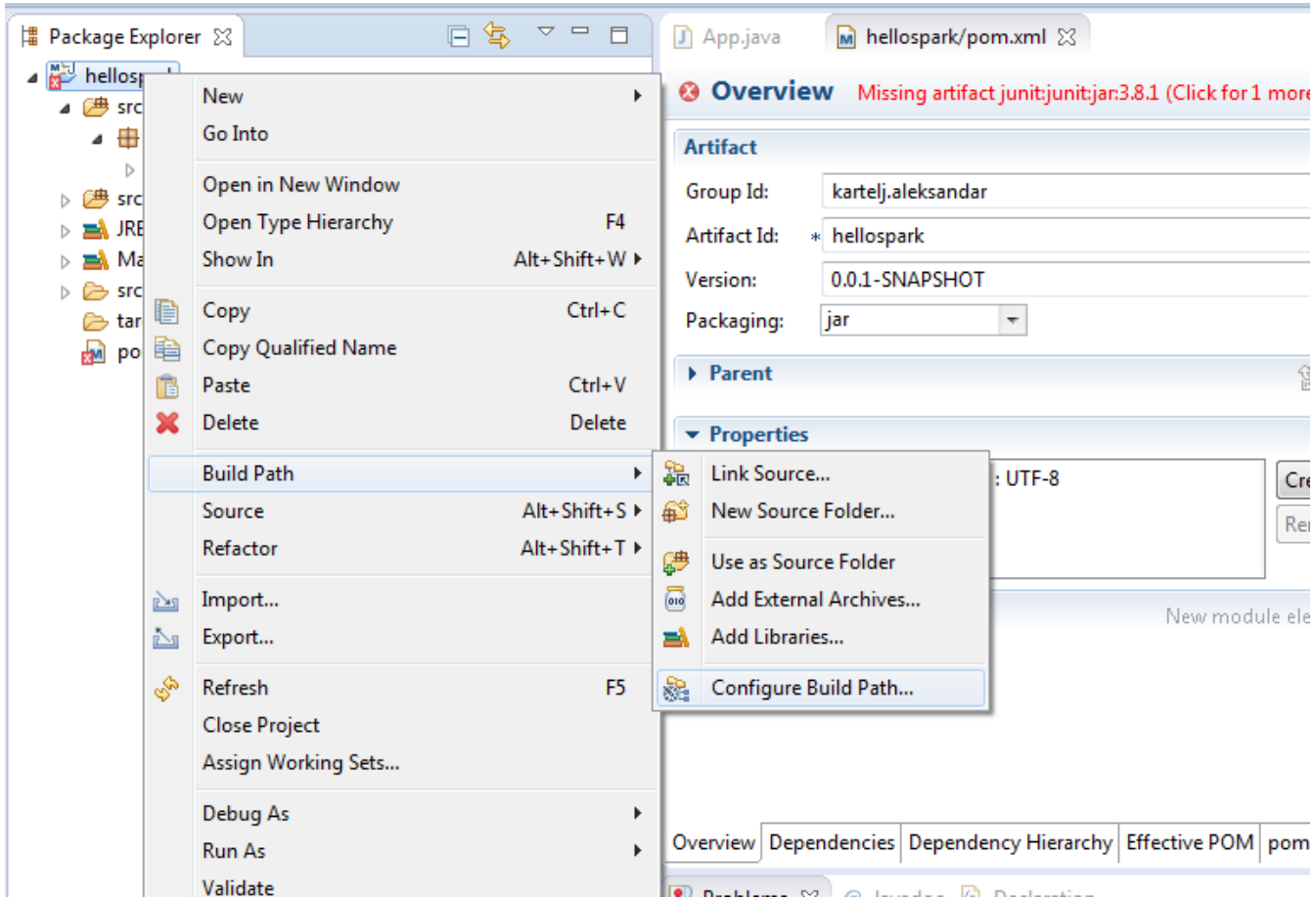


3. Dodavanje podrške za Spark u maven projekat

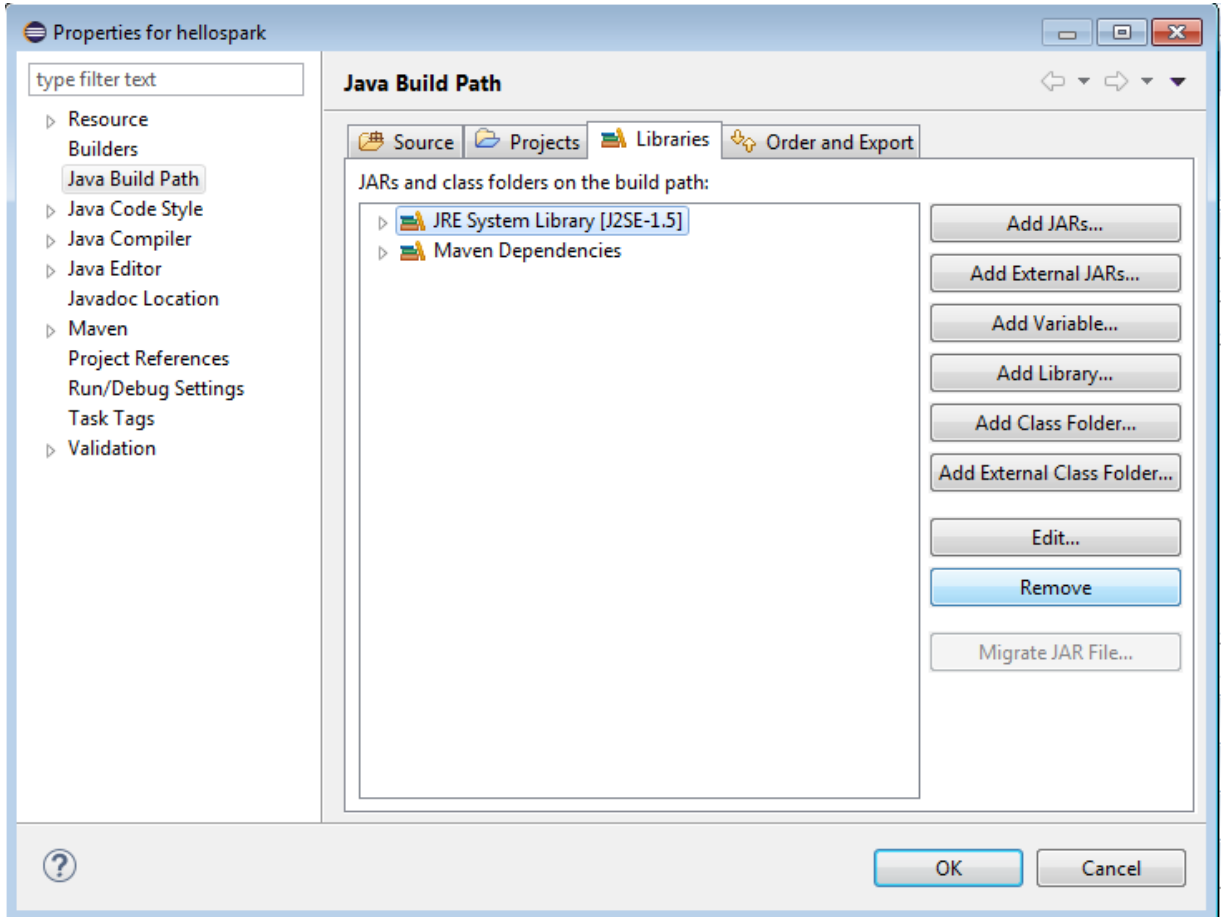


Crveni zbog pogrešne verzije Jave. Postavljen J2SE-1.5 a treba verzija 8.

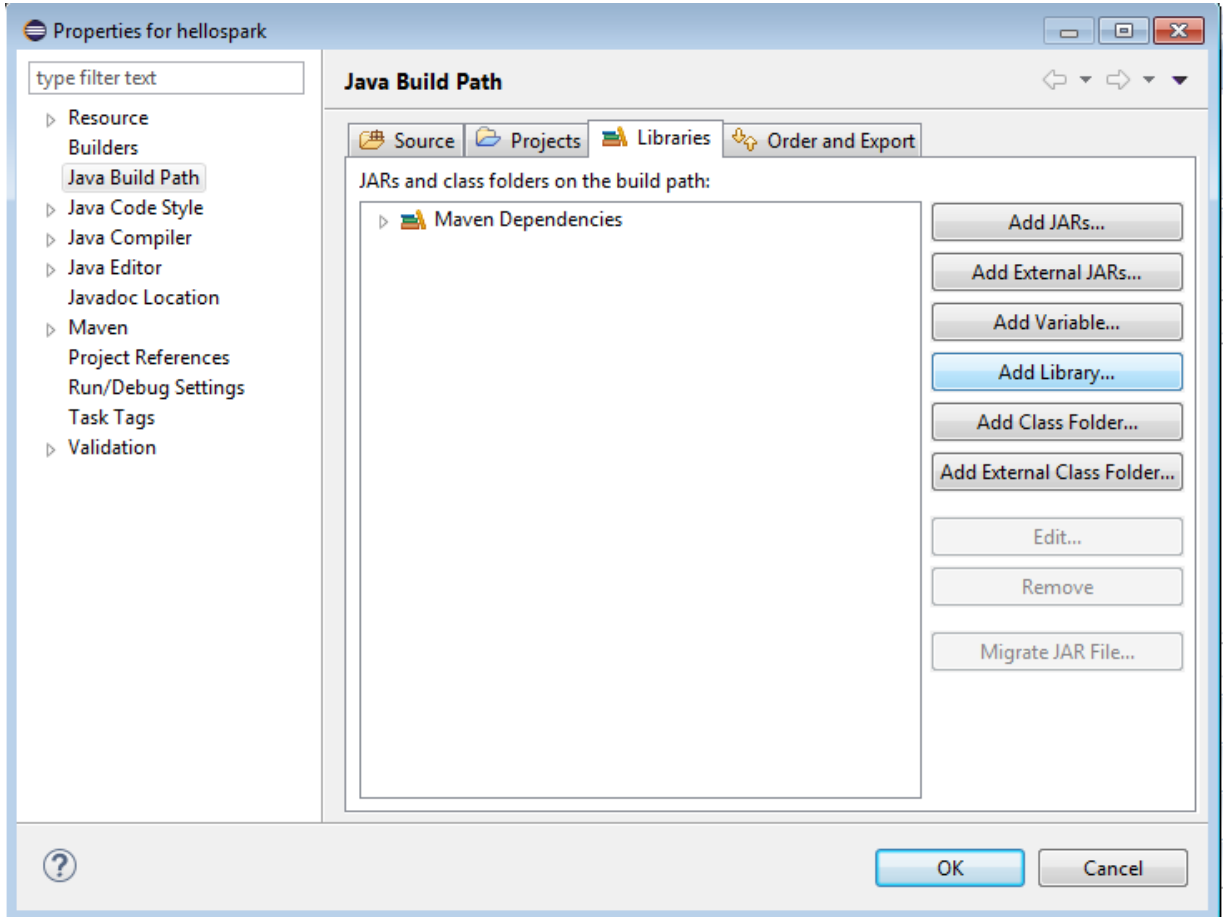
3A. Dodavanje podrške za Spark u maven projekat – Podešavanje Java 8 verzije



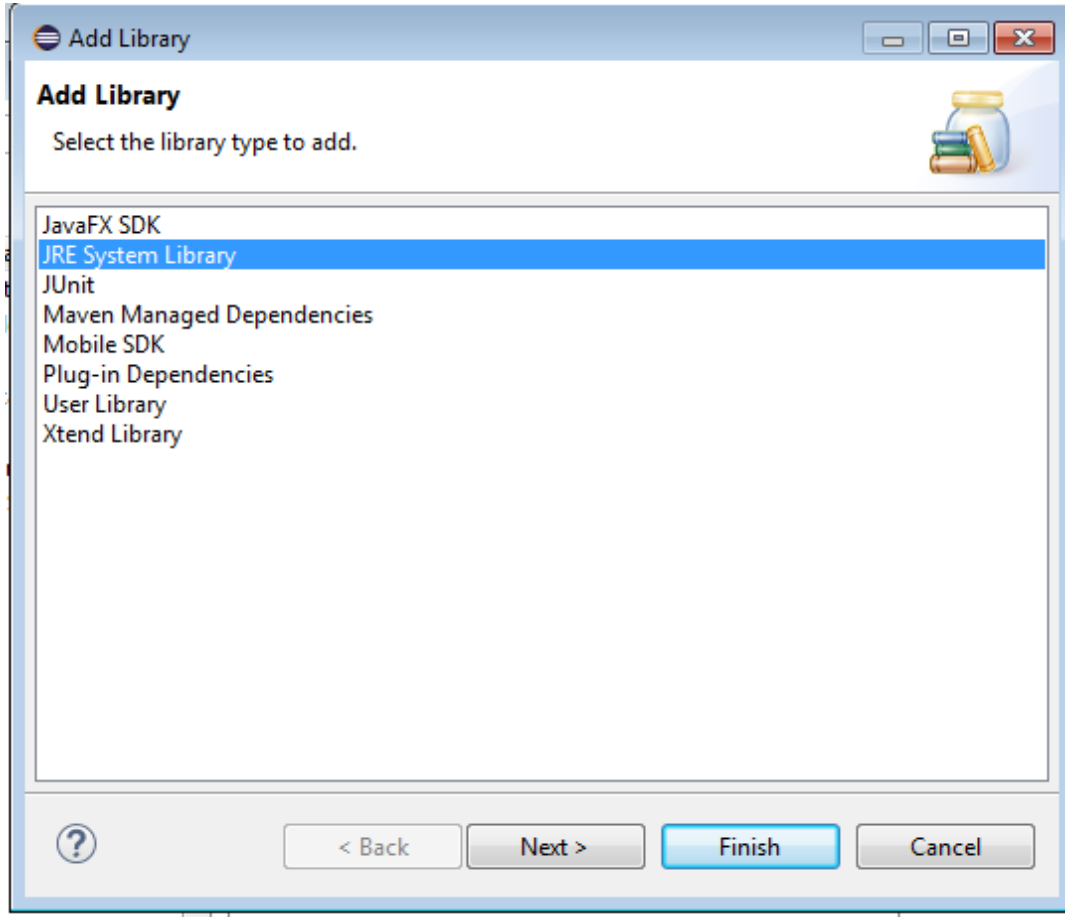
3A. Dodavanje podrške za Spark u maven projekat – Podešavanje Java 8 verzije



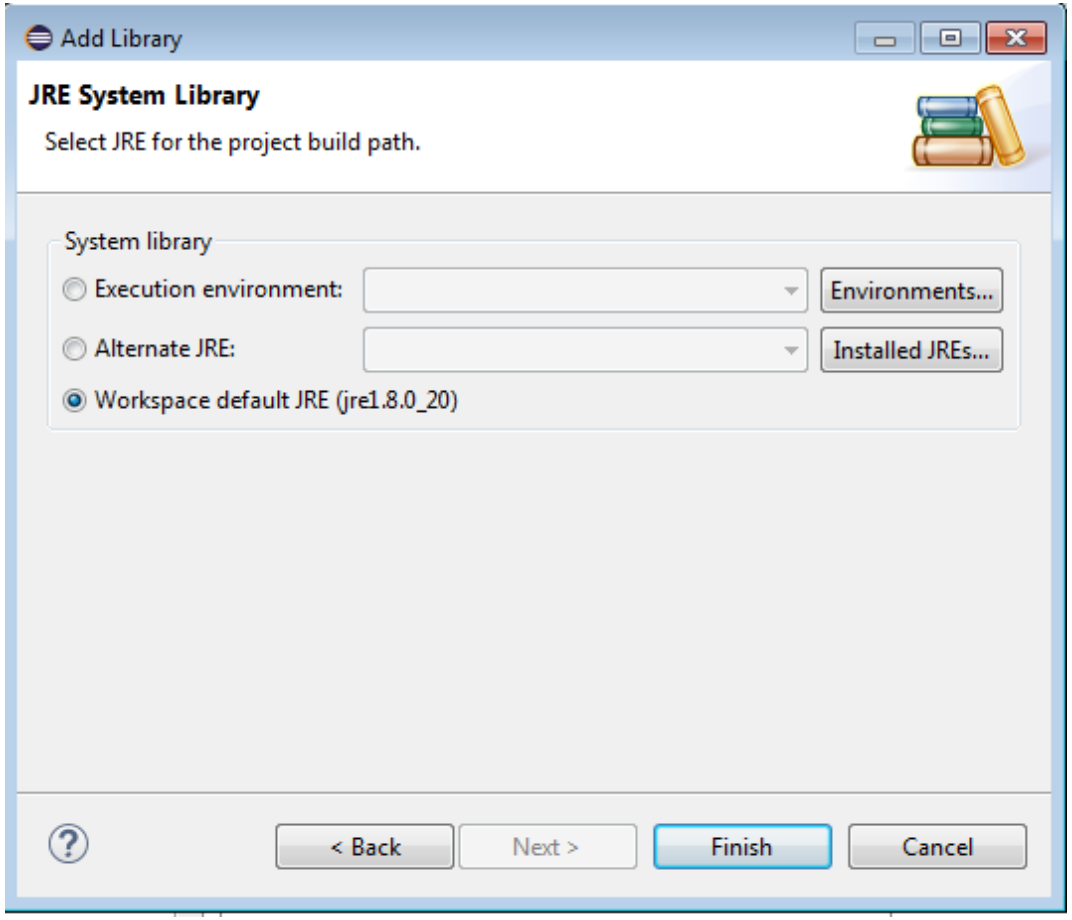
3A. Dodavanje podrške za Spark u maven projekat – Podešavanje Java 8 verzije



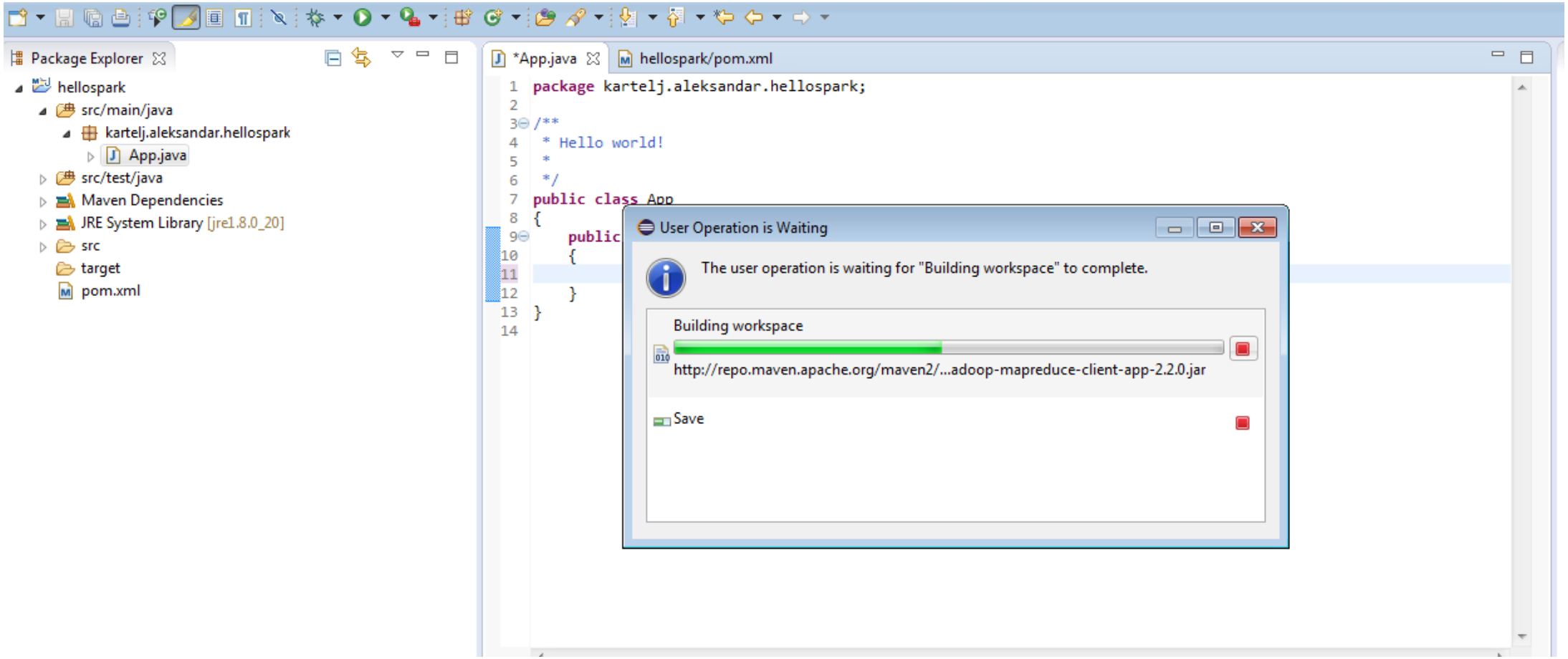
3A. Dodavanje podrške za Spark u maven projekat – Podešavanje Java 8 verzije



3A. Dodavanje podrške za Spark u maven projekat – Podešavanje Java 8 verzije



4. Izvršavanje jednostavnog programa u lokalnom okruženju



4. Izvršavanje jednostavnog programa u lokalnom okruženju

App.java | hellospark/pom.xml

```
1  package kartelj.aleksandar.hellospark;
2
3  import java.util.Arrays;
4  import java.util.List;
5
6  import org.apache.spark.SparkConf;
7  import org.apache.spark.api.java.JavaRDD;
8  import org.apache.spark.api.java.JavaSparkContext;
9
10 /**
11  * Hello world!
12  *
13  */
14 public class App
15 {
16     public static void main( String[] args )
17     {
18         //postavljamo konfiguraciju cvorova
19         SparkConf conf = new SparkConf().setAppName("aplikacija1").setMaster("local[*]");
20         JavaSparkContext sc = new JavaSparkContext(conf);
21         //rasporedjujemo podatke po raspolozivim cvorovima
22         List<Integer> data = Arrays.asList(1, 2, 3, 4, 5);
23         JavaRDD<Integer> distData = sc.parallelize(data);
24         Integer zbir = distData.reduce((x,y)->x+y);
25     }
26 }
27
```

Može da crveni jer ne prepoznaje lambda izraze.
Iako smo podesili verziju Jave 8 source kod se i dalje validira prema ranijoj verziji.
Kliknite na žuto crvenu ikonicu i pritisnite „Change project compliance...“

4. Izvršavanje jednostavnog programa u lokalnom okruženju

App.java | hellospark/pom.xml

```
1  package kartelj.aleksandar.hellospark;
2
3  import java.util.Arrays;
4  import java.util.List;
5
6  import org.apache.spark.SparkConf;
7  import org.apache.spark.api.java.JavaRDD;
8  import org.apache.spark.api.java.JavaSparkContext;
9
10 /**
11  * Hello world!
12  *
13  */
14 public class App
15 {
16     public static void main( String[] args )
17     {
18         //postavljamo konfiguraciju cvorova
19         SparkConf conf = new SparkConf().setAppName("aplikacija1").setMaster("local[*]");
20         JavaSparkContext sc = new JavaSparkContext(conf);
21         //rasporedjujemo podatke po raspolozivim cvorovima
22         List<Integer> data = Arrays.asList(1, 2, 3, 4, 5);
23         JavaRDD<Integer> distData = sc.parallelize(data);
24 Lambda expressions are allowed only at source level 1.8 or above);
25     }
26 }
27
```

Change project compliance and JRE to 1.8

Set proj
Set proj

4. Izvršavanje jednostavnog programa u lokalnom okruženju

App.java | hellospark/pom.xml

```
1  package kartelj.aleksandar.hellospark;
2
3  import java.util.Arrays;
4  import java.util.List;
5
6  import org.apache.spark.SparkConf;
7  import org.apache.spark.api.java.JavaRDD;
8  import org.apache.spark.api.java.JavaSparkContext;
9
10 /**
11  * Hello world!
12  *
13  */
14 public class App
15 {
16     public static void main( String[] args )
17     {
18         //postavljamo konfiguraciju cvorova
19         SparkConf conf = new SparkConf().setAppName("aplikacija1").setMaster("local[*]");
20         JavaSparkContext sc = new JavaSparkContext(conf);
21         //rasporedjujemo podatke po raspolozivim cvorovima
22         List<Integer> data = Arrays.asList(1, 2, 3, 4, 5);
23         JavaRDD<Integer> distData = sc.parallelize(data);
24         Integer zbir = distData.reduce((x,y)->x+y);
25     }
26 }
27
```

4. Izvršavanje jednostavnog programa u lokalnom okruženju

App.java | hellospark/pom.xml

```
1  package kartelj.aleksandar.hellospark;
2
3  import java.util.Arrays;
4  import java.util.List;
5
6  import org.apache.spark.SparkConf;
7  import org.apache.spark.api.java.JavaRDD;
8  import org.apache.spark.api.java.JavaSparkContext;
9
10 /**
11  * Hello world!
12  *
13  */
14 public class App
15 {
16     public static void main( String[] args )
17     {
18         //postavljamo konfiguraciju cvorova
19         SparkConf conf = new SparkConf().setAppName("aplikacija1").setMaster("local[*]");
20         JavaSparkContext sc = new JavaSparkContext(conf);
21         //rasporedjujemo podatke po raspolozivim cvorovima
22         List<Integer> data = Arrays.asList(1, 2, 3, 4, 5);
23         JavaRDD<Integer> distData = sc.parallelize(data);
24         //ovo je kao onaj fold kod haskell-a
25         Integer zbir = distData.reduce((x,y)->x+y);
26         //ispis rezultata na centralnom cvoru
27         System.out.println(zbir);
28     }
29 }
30
```

Toliko za sada...

Povezivanje na Amazon Hadoop sledeći put...