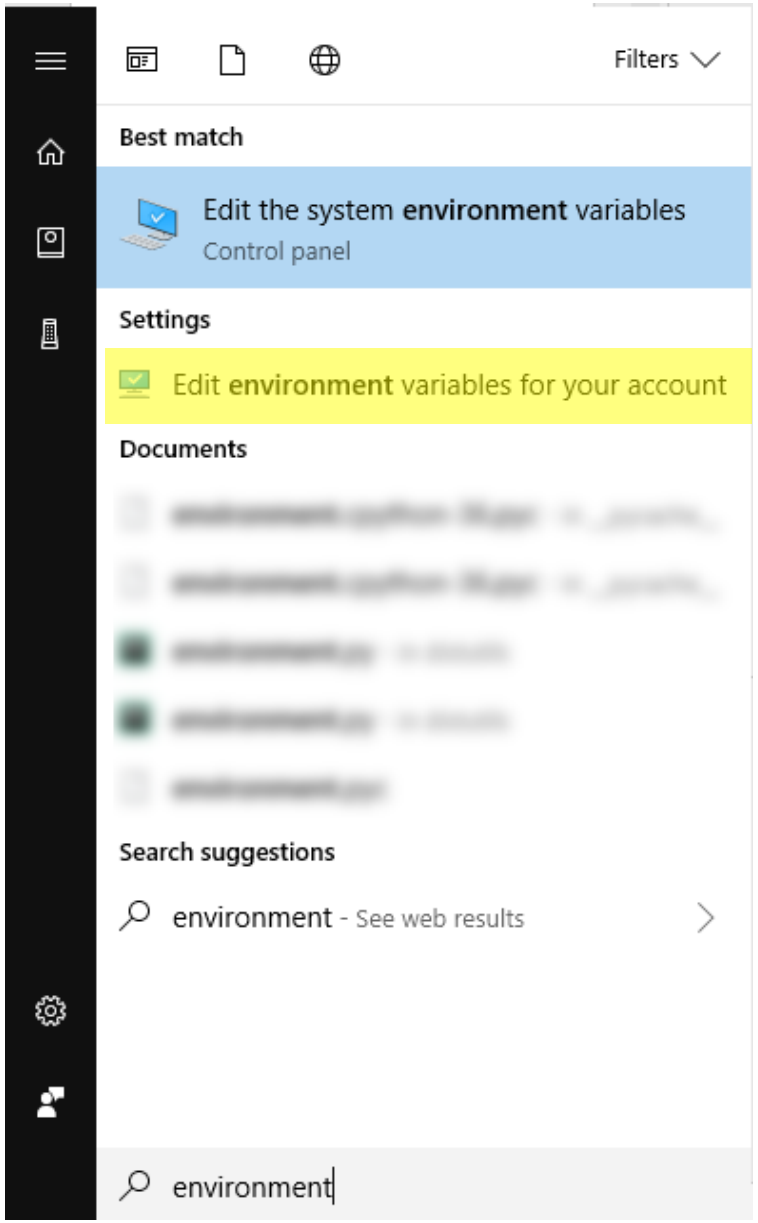
Spark Programming

Setting up a stand-alone Spark instance

Installing Spark on Windows 10

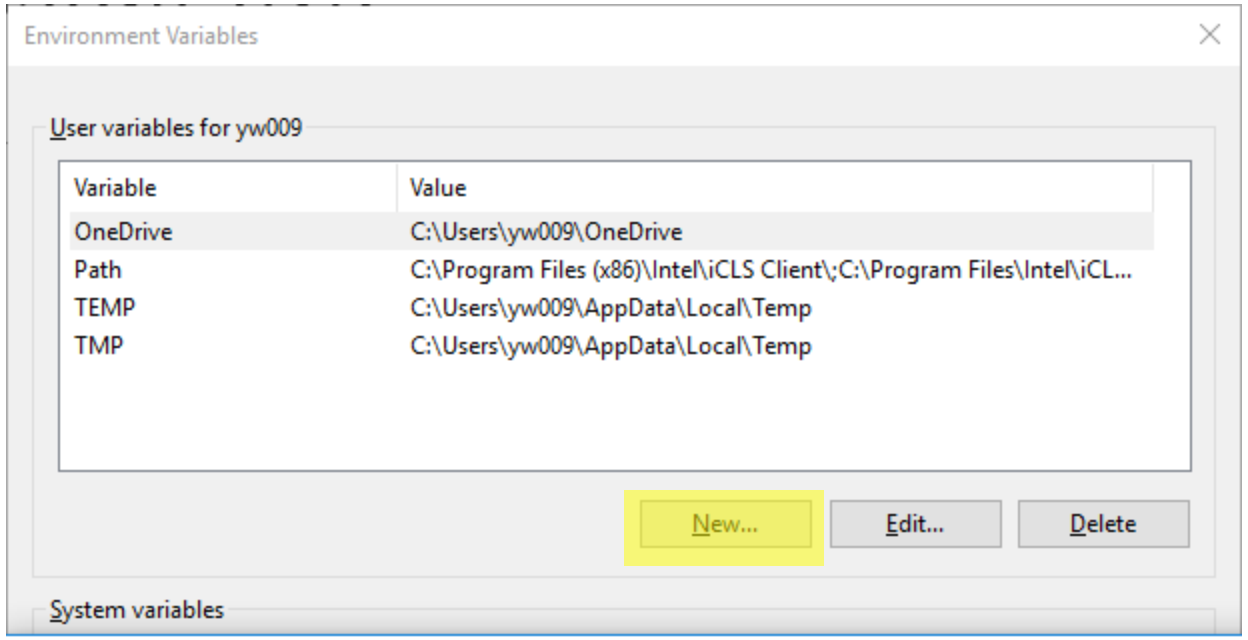
Steps of Set Environment Variables

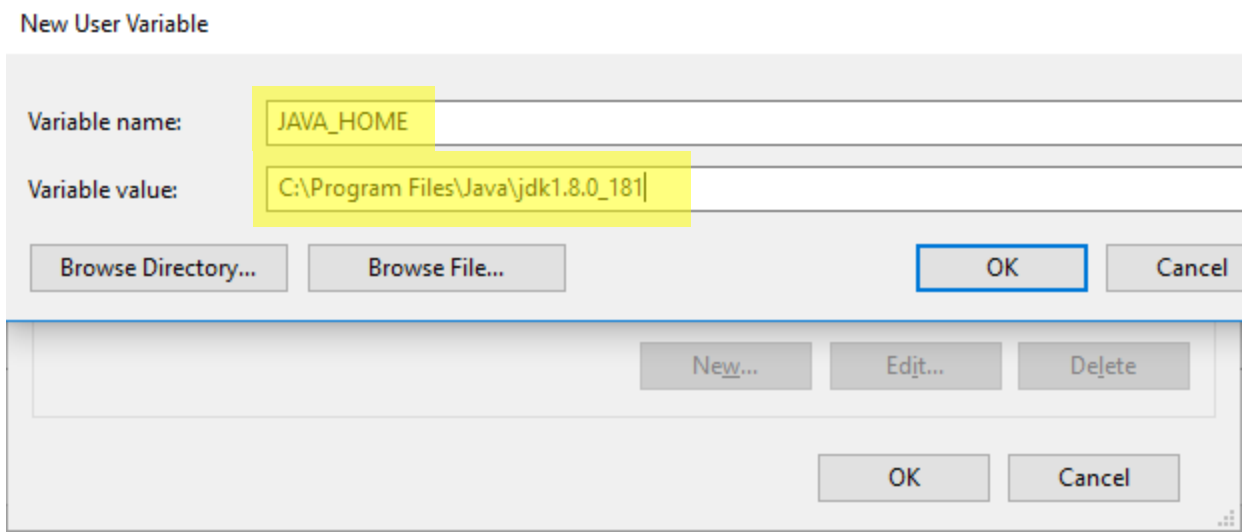
- Open Environment Variables
  - Open Start and type “environment”,
  - Select “Edit environment variables for your account”. (see highlight of the figure)



Steps of Set Environment Variables

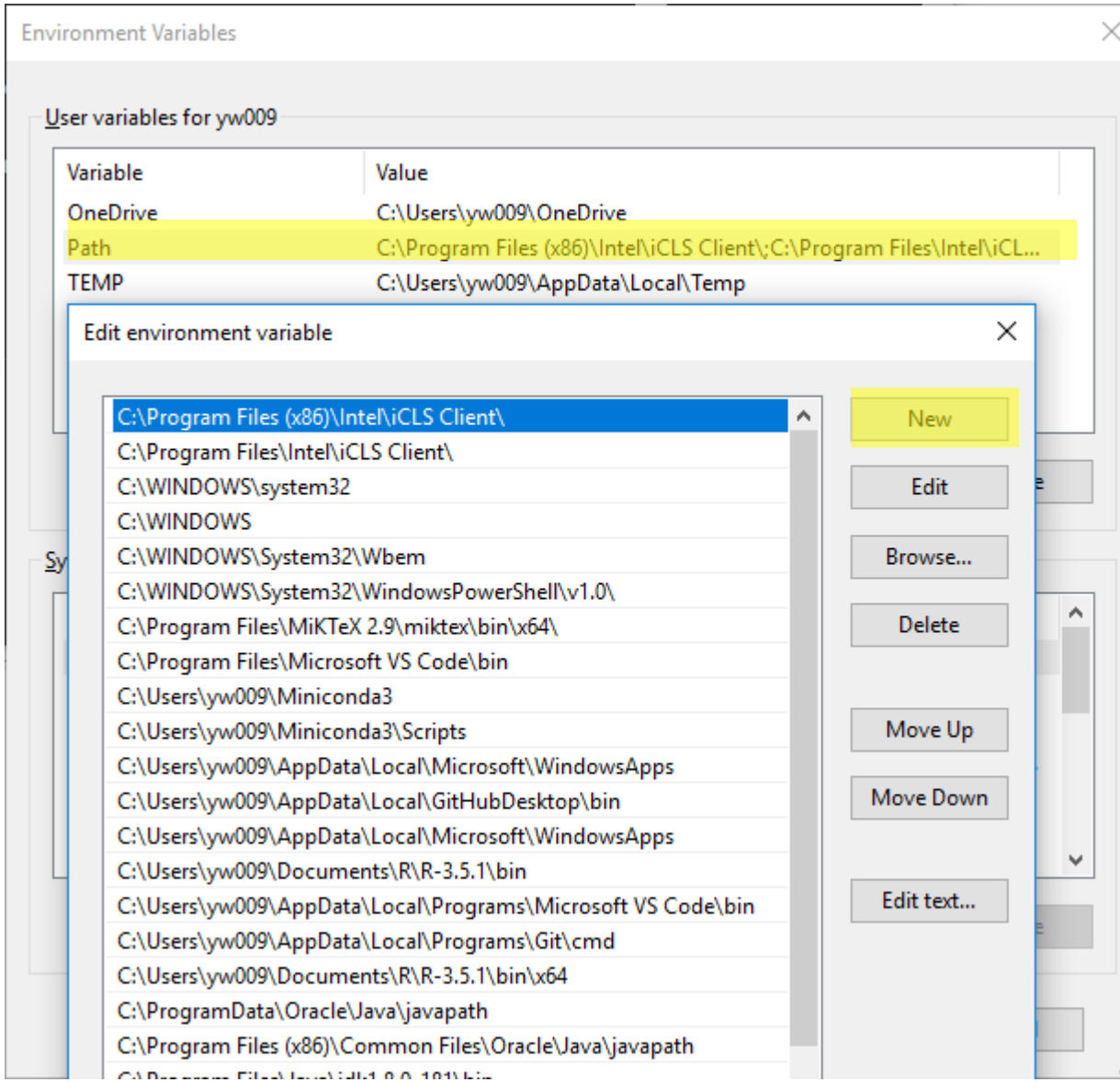
- Set Java_HOME
  1. Click on the “New…” button
  2. In the Pop-up window, Type “JAVA_HOME” and your jdk path





Steps of Set Environment Variables

- Add to path
- Steps
  1. Double-click on the Path
  2. In the Pop-up window, click on the “New” button, type “%JAVA_HOME%\bin”



Install Scala

- Download Scala from the link: http://downloads.lightbend.com/scala/2.11.8/scala-2.11.8.msi
- Set environmental variables:
  - User variable:
    - Variable: SCALA_HOME;
    - Value: C:\Program Files (x86)\scala
  - System variable:
    - Variable: PATH
    - Value: %SCALA_HOME%\bin
- Check it on cmd, see below:

```
D:\>scala
Welcome to Scala 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_91).
Type in expressions for evaluation. Or try :help.

scala> _
```

Install Java 8

- Download Java 8 from the link: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
- Set environmental variables:
  - User variable:
    - Variable: JAVA_HOME;
    - Value: C:\Program Files\Java\jdk1.8.0_91
  - System variable:
    - Variable: PATH
    - Value: %JAVA_HOME%\bin
- Check on cmd, see below:

```
D:\>java -version
java version "1.8.0_91"
Java(TM) SE Runtime Environment (build 1.8.0_91-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.91-b14, mixed mode)

D:\>_
```

Install Eclipse

- Download it from the link: https://eclipse.org/downloads/ and extract it into your hard drive

Install Spark 2.3.1

- Download it from the following link: http://spark.apache.org/downloads.html and extract it into D drive, such as D:\Spark
- Set environmental variables:
  - User variable:
    - Variable: SPARK_HOME;
    - Value: D:\spark\spark-2.3.1-bin-hadoop2.7
  - System variable:
    - Variable: PATH
    - Value: %SPARK_HOME%\bin

Download Windows Utilities

- Download it from the link: https://github.com/steveloughran/winutils/tree/master/hadoop-2.7.1 and paste it in D:\winutils
- Set environmental variables:
  - User variable:
    - Variable: HADOOP_HOME;
    - Value: D:\winutils\hadoop-2.7.1
  - System variable:
    - Variable: PATH
    - Value: %HADOOP_HOME%\bin

Install Maven 3.3

- Download Apache-Maven-3.3.9 from the link: http://apache.mivzakim.net/maven/maven-3/3.3.9/binaries/apache-maven-3.3.9-bin.zip and extract it into D drive, such as D:\apache-maven-3.3.9
- Set environmental variables:
  - User variable:
    - Variable: MAVEN_HOME;
    - Value: D:\apache-maven-3.3.9
  - System variable:
    - Variable: PATH
    - Value: %MAVEN_HOME%\bin

Validate Installation

- Check if the spark is correctly installed
  - Open CMD, type spark-shell

Command Prompt - spark-shell

```
Microsoft Windows [Version 10.0.17134.228]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\yw009>spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://wu-lab-03:4040
Spark context available as 'sc' (master = local[*], app id = local-1536079488512).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.3.1
      /_/

Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_181)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Installing Spark on Ubuntu

Install Spark on Ubuntu

- Ubuntu 16.04+
- Check Java version(=1.8),

  > java -version
- Otherwise

  > sudo apt-get default-jdk
- Download Spark 2.3.1 from http://spark.apache.org/downloads.html
- Unzip Spark to a destination, e.g. $HOME
- Set environment variables to spark

  > export SPARK_HOME="$HOME/spark-2.3.1-bin-hadoop2.7"

  > export HADOOP_HOME=$SPARK_HOME

  > export SPARK_LOCAL_IP=127.0.0.1

Installing Spark on Mac OS

Install Spark on Mac OS

- Java version = 1.8
- Set JAVA_HOME environment variable
  - > export JAVA_HOME="$(/usr/libexec/java_home)“
- Download Spark 2.3.1 from http://spark.apache.org/downloads.html
- Unzip Spark to a destination, e.g. $HOME
- Set environment variables to spark
  - > export SPARK_HOME="$HOME/spark-2.3.1-bin-hadoop2.7"
  - > export HADOOP_HOME=$SPARK_HOME
  - > export SPARK_LOCAL_IP=127.0.0.1

Running the Spark Shell

Spark Shell for Python and Scala

- Open a terminal window on Mac or Linux or a command window on Windows
- For Python Spark shell, run: pyspark
  - The Spark shell can be used to write and execute regular Python programs
  - Can also be used to write Spark applications in Python
- For Scala Spark shell, run: spark-shell
  - The Spark shell can be used to write and execute regular Scala programs
  - Can also be used to write Spark applications in Scala

Submitting Python Applications

Submitting Python Applications

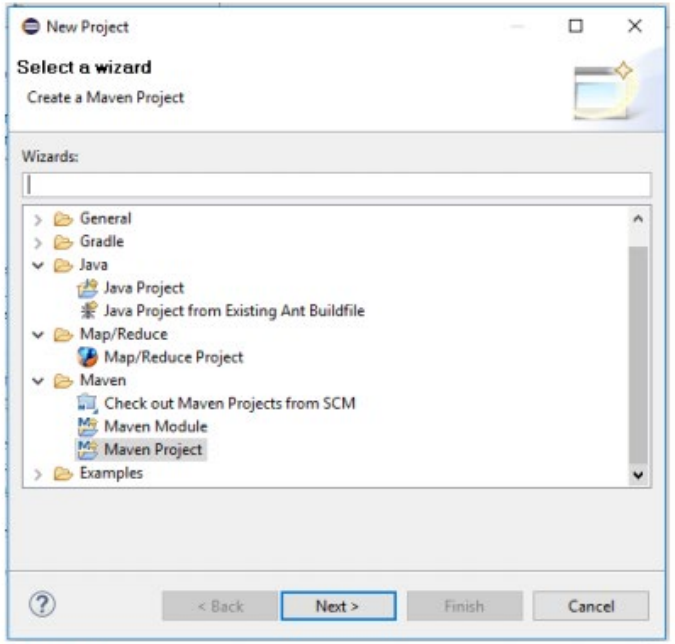
- Write your Python code and save as **.py
- Open CMD, type: spark-submit **.py [path]

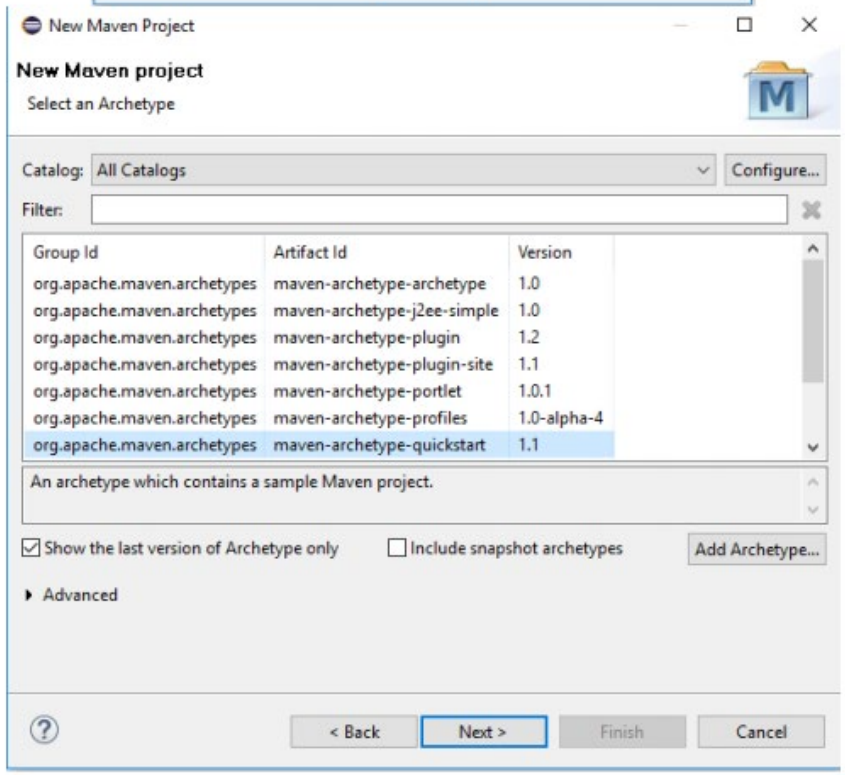
```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.17134.228]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\yw009\Desktop>spark-submit wordcount.py wordlist.txt
2018-09-04 12:18:23 INFO  SparkContext:54 - Running Spark version 2.3.1
2018-09-04 12:18:23 INFO  SparkContext:54 - Submitted application: PythonWordCount
2018-09-04 12:18:23 INFO  SecurityManager:54 - Changing view acls to: yw009
2018-09-04 12:18:23 INFO  SecurityManager:54 - Changing modify acls to: yw009
2018-09-04 12:18:23 INFO  SecurityManager:54 - Changing view acls groups to:
2018-09-04 12:18:23 INFO  SecurityManager:54 - Changing modify acls groups to:
2018-09-04 12:18:23 INFO  SecurityManager:54 - SecurityManager: authentication disabled; ui acls disabled; users  with v
iew permissions: Set(yw009); groups with view permissions: Set(); users  with modify permissions: Set(yw009); groups wit
h modify permissions: Set()
2018-09-04 12:18:23 INFO  Utils:54 - Successfully started service 'sparkDriver' on port 58022.
2018-09-04 12:18:23 INFO  SparkEnv:54 - Registering MapOutputTracker
2018-09-04 12:18:23 INFO  SparkEnv:54 - Registering BlockManagerMaster
2018-09-04 12:18:23 INFO  BlockManagerMasterEndpoint:54 - Using org.apache.spark.storage.DefaultTopologyMapper for getti
ng topology information
2018-09-04 12:18:23 INFO  BlockManagerMasterEndpoint:54 - BlockManagerMasterEndpoint up
2018-09-04 12:18:23 INFO  DiskBlockManager:54 - Created local directory at C:\Users\yw009\AppData\Local\Temp\blockmgr-b8
e9a90b-b634-4dcc-8aa3-010c70051ca5
2018-09-04 12:18:23 INFO  MemoryStore:54 - MemoryStore started with capacity 366.3 MB
2018-09-04 12:18:23 INFO  SparkEnv:54 - Registering OutputCommitCoordinator
2018-09-04 12:18:23 INFO  log:192 - Logging initialized @6806ms
2018-09-04 12:18:23 INFO  Server:346 - jetty-9.3.z-SNAPSHOT
2018-09-04 12:18:23 INFO  Server:414 - Started @6882ms
2018-09-04 12:18:23 INFO  AbstractConnector:278 - Started ServerConnector@44be25d9{HTTP/1.1,[http/1.1]}{0.0.0.0:4040}
2018-09-04 12:18:23 INFO  Utils:54 - Successfully started service 'SparkUI' on port 4040.
2018-09-04 12:18:23 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@413ac0a8{/jobs,null,AVAILABLE,@Spar
k}
```

Submitting Java Applications

Create First WordCount Project

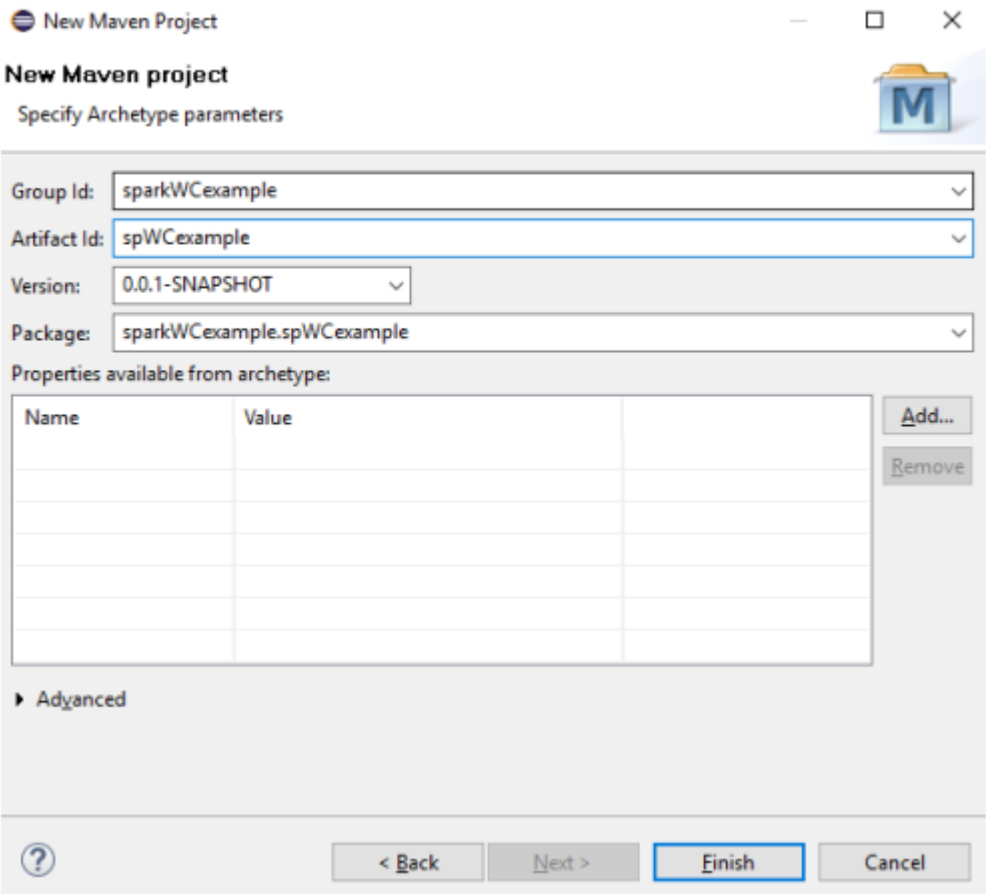
- Open Eclipse and do File->New->project->Maven Project; see right.





Create First WordCount Project

- Enter Group id, Artifact id, and click finish.



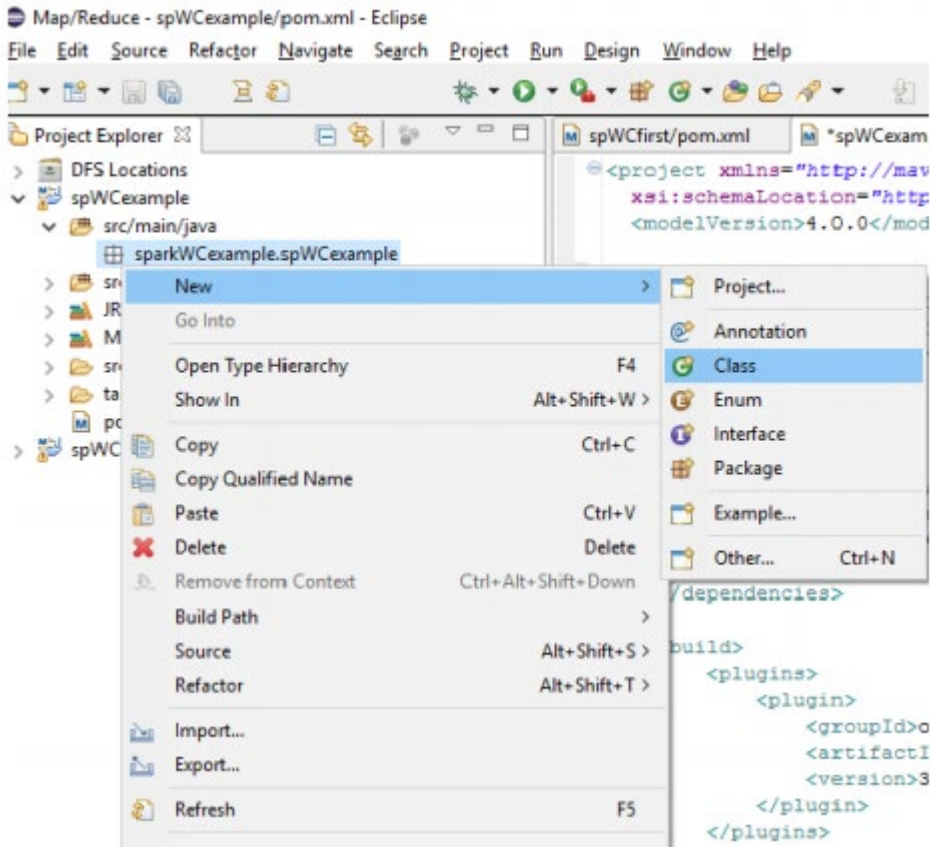
Create First WordCount Project

- Edit pom.xml. Paste the following code.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance"
 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>sparkWCexample</groupId>
   <artifactId>spWCexample</artifactId>
   <version>0.0.1-SNAPSHOT</version>
   <dependencies>
      <dependency>
         <groupId>org.apache.spark</groupId>
         <artifactId>spark-core_2.11</artifactId>
         <version>2.2.0</version>
      </dependency>
   </dependencies>
   <build>
      <plugins>
         <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.3</version>
         </plugin>
      </plugins>
   </build>
</project>
```

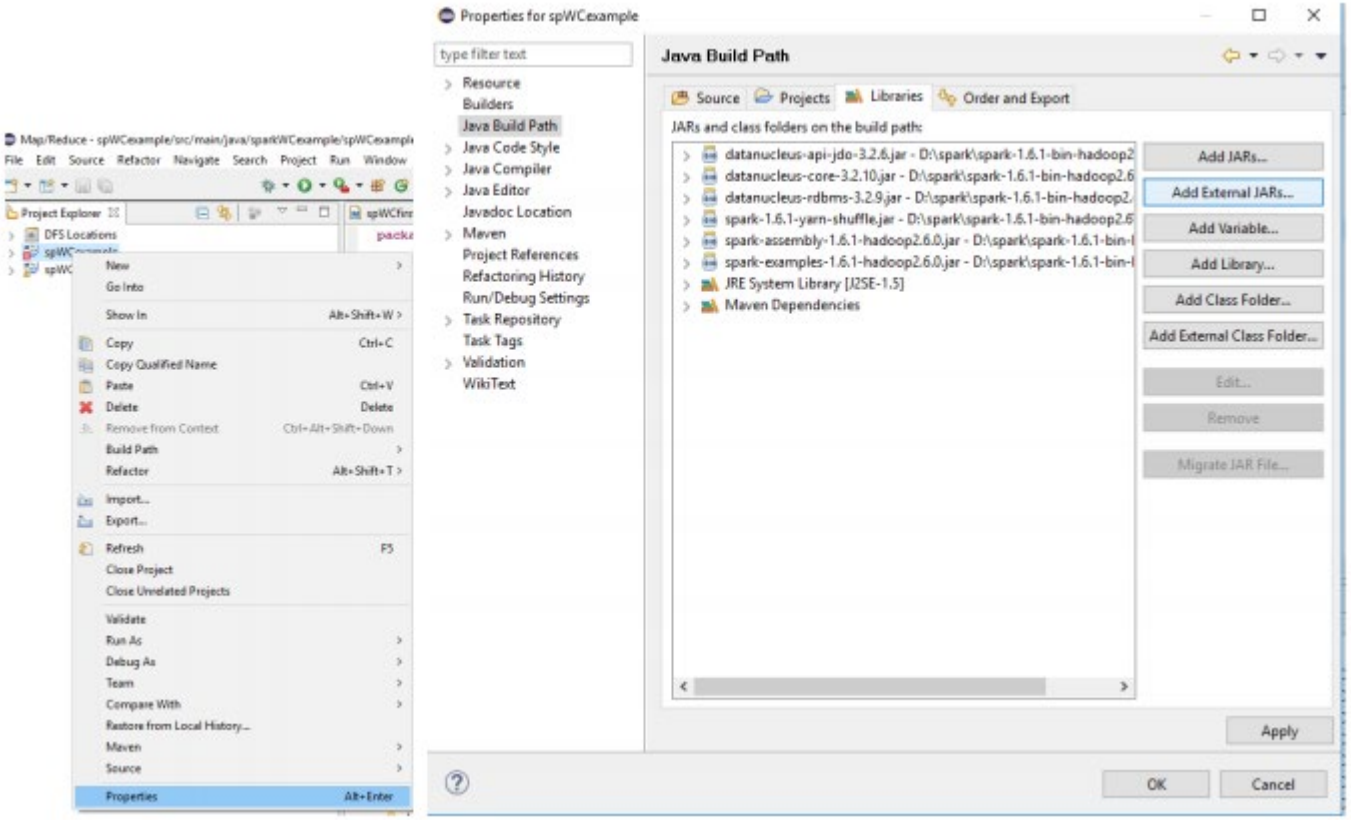
Create First WordCount Project

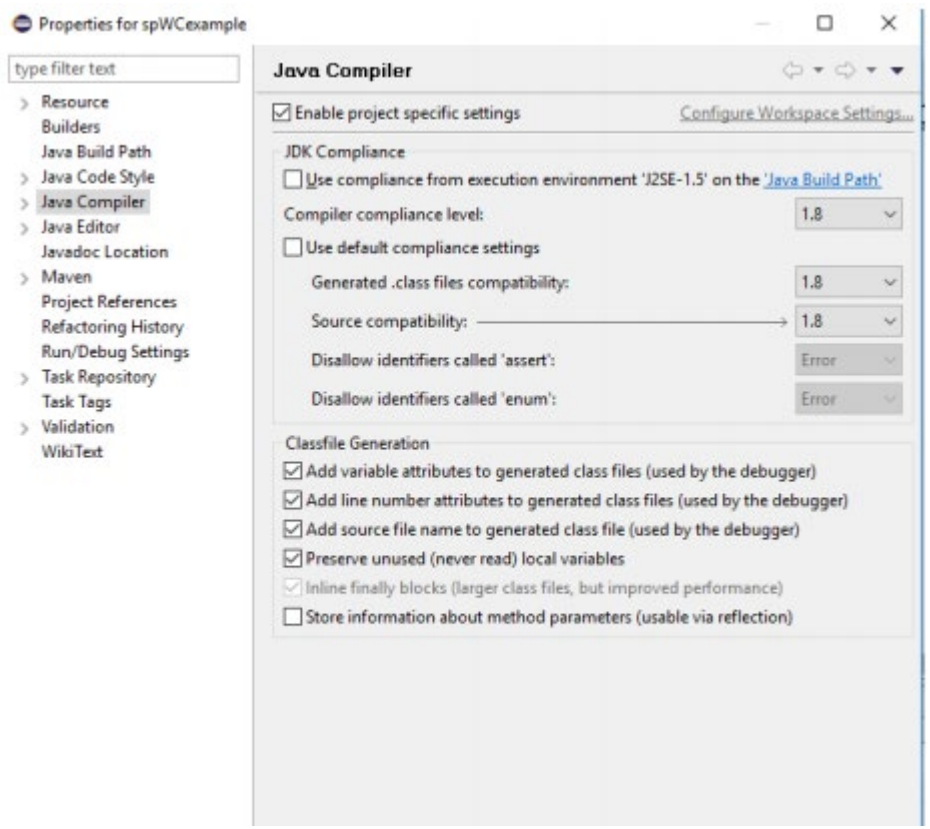
- Write your java code in class JavaWordCount



Create First WordCount Project

- Add external jar from the location D:\spark\spark-1.6.1-bin-hadoop2.6\lib and set Java 8 for compilation; see below.





Create First WordCount Project

- Build the project: Go to the following location (where we stored the project) on cmd: D:\hadoop\examples\spWCexample
  - Write **mvn package** on cmd
- Execute the project:
  - Write the following command
  - spark-submit --class sparkWCexample.spWCexample.JavaWordCount --master local[2] D:\spark\spark-2.3.1-bin-hadoop2.7\workspace\spWCexample-0.0.1-SNAPSHOT.jar D:\spark\spark-2.3.1-bin-hadoop2.7\workspace\input.txt D:\spark\spark-2.3.1-bin-hadoop2.7\workspace\output

Word Count

Word Count in Java

- The first step of every Java Spark application is to create a Spark context:

```
import java.util.Arrays;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import scala.Tuple2;
public class JavaWordCount {
  public static void main(String[] args) throws Exception {
    SparkConf conf = new SparkConf();
    JavaSparkContext sc = new JavaSparkContext(conf);
    ...
  }
}
```

Word Count in Java

- Next, you'll need to read the target file into an RDD:

```
JavaRDD<String> lines = sc.textFile(args[0]);
```

- You now have an RDD filled with strings, one per line of the file.
- Next you'll want to split the lines into individual words:

```
JavaRDD<String> words =
  lines.flatMap(l -> Arrays.asList(l.split(“ ”)).iterator());
```

- The flatMap() operation first converts each line into an array of words, and then makes each of the words an element in the new RDD. Note that the lambda argument to the method must return an iterator, not a list or array.

Word Count in Java

- Next, you'll want to replace each word with a tuple of that word and the number 1.

```
JavaPairRDD<String, Integer> pairs =
  words.mapToPair(w -> new Tuple2<>(w, 1));
```

- The mapToPair() operation replaces each word with a tuple of that word and the number 1. The pairs RDD is a pair RDD where the word is the key, and all of the values are the number 1. Note that the type of the RDD is now JavaPairRDD. Also note that the use of the Scala Tuple2 class is the normal and intended way to perform this operation.

Word Count in Java

- Now, to get a count of the number of instances of each word, you need only group the elements of the RDD by key (word) and add up their values:

```
JavaPairRDD<String, Integer> counts =
  pairs.reduceByKey((n1, n2) -> n1 + n2);
```

- The reduceByKey() operation keeps adding elements' values together until there are no more to add for each key (word).

Word Count in Java

- Finally, you can store the results in a file and stop the context:

```
counts.saveAsTextFile(args[1]);
sc.stop();
```

Word Count in Java

```
import java.util.Arrays;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import scala.Tuple2;
public class WordCount {
public static void main(String[] args) throws Exception {
SparkConf conf = new SparkConf();
JavaSparkContext sc = new JavaSparkContext(conf);
JavaRDD<String> lines = sc.textFile(args[0]);
JavaRDD<String> words =
lines.flatMap(l -> Arrays.asList(l.split(“ ")).iterator());
JavaPairRDD<String, Integer> pairs =
words.mapToPair(w -> new Tuple2<>(w, 1));
JavaPairRDD<String, Integer> counts =
pairs.reduceByKey((n1, n2) -> n1 + n2);
counts.saveAsTextFile(args[1]);
sc.stop();
}
}
```

Word Count in Python

- Create a Spark context:

```
import re
import sys
from pyspark import SparkConf, SparkContext
conf = SparkConf()
sc = SparkContext(conf=conf)
```

Word Count in Python

- Next, you'll need to read the target file into an RDD:

```
lines = sc.textFile(sys.argv[1])
```

- You now have an RDD filled with strings, one per line of the file.
- Next you'll want to split the lines into individual words:

```
words = lines.flatMap(lambda line: line.split(" "))
```

- Next, you'll want to replace each word with a tuple of that word and the number 1.

```
pairs = words.map(lambda w: (w, 1))
```

Word Count in Python

- Now, to get a count of the number of instances of each word, you need only group the elements of the RDD by key (word) and add up their values:

```
counts = pairs.reduceByKey(lambda n1, n2: n1 + n2)
```

- Finally, you can store the results in a file and stop the context:

```
counts.saveAsTextFile(sys.argv[2])
sc.stop()
```

Word Count in Python

```
import sys
from pyspark import SparkConf, SparkContext
sc = SparkContext("local", "pysaprk word counts")
lines = sc.textFile(sys.argv[1]).cache()
words = lines.flatMap(lambda line: line.split(" "))
pairs = words.map(lambda word: (word, 1))
counts = pairs.reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("counts");
sc.stop()
```