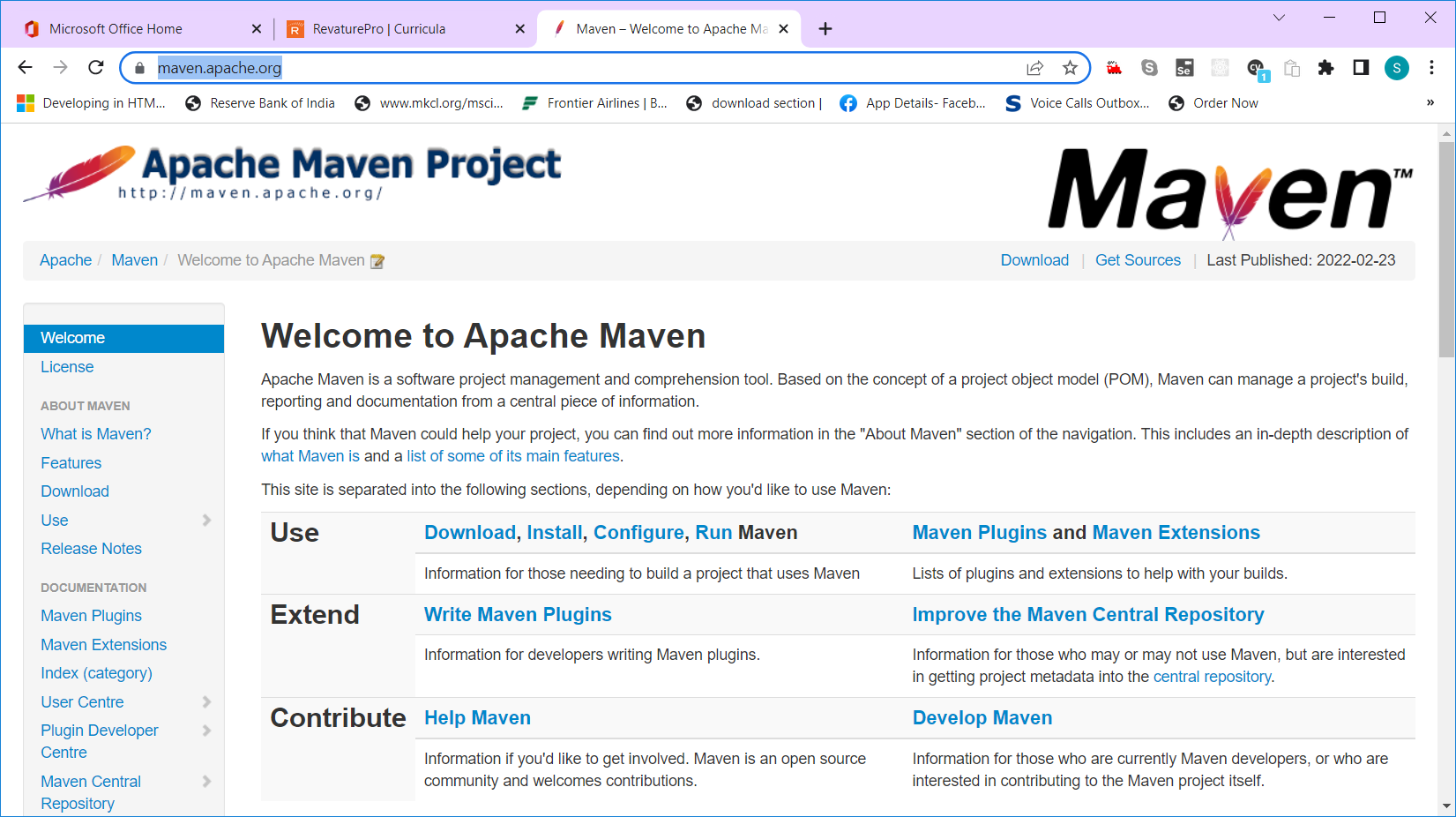
Revisit –

Maven – It’s a open source project management tool for Java. Maintained by Apache.

Official Site : <https://maven.apache.org/>



Download & Extract - [apache-maven-3.8.4-bin.zip](https://dlcdn.apache.org/maven/maven-3/3.8.4/binaries/apache-maven-3.8.4-bin.zip) from the site <https://maven.apache.org/download.cgi>

Added JAVA\_HOME & M2\_HOME to Environment Variable and updated them in path.

JAVA\_HOME = C:\Program Files\Java\jdk1.8.0\_281

M2\_HOME = C:\apache-maven-3.8.1

Path = %JAVA\_HOME%\bin;%M2\_HOME%\bin;

Maven will search for JAVA\_HOME environment variable.

Two ways of accessing 1) Using Maven CLI (Command Line Interface) 2) Using IDE plugins

Maven commands a) mvn install b) mvn -v c) mvn package d)mvn archetype:generate e) mvn clean

Maven will expect a file called pom.xml

Pom.xml – Project Object model, This is back bone of the maven project

Maven Local repo - .m2 (C:\Users\Siva\.m2\repository)

Maven Central/remote repo – mvnrepository.com , repo.maven.org , search.maven.org

Maven project management activities

* Build the project
* Test the project
* Generate source
* Package the project
* Deploy the project
* Manage all the dependencies of the project

For a pom.xml file, the important properties are a)groupId b) artifactId

* GroupId will be reverse of the company url (need to fill it)
* artifactId will be the name of the project (need to fill it)

Creating Maven Projects

Method 1) using maven cli – a) batch mode b) interactive mode [mvn archetype:generate]

Method 2) using IDE (Eclipse/STS) File🡪New🡪others 🡪Maven Project (maven)

Enter values for groupId and artifactId, build type, version , description

Default folder structure of a maven structure –

Src/main/java - source code folder

Src/main/test – Unit test code folder

Target – output folder

Threads – In Java, Multi thread concepts helps to perform parallel processing

Parallel Processing

1. Printing “Hello World” message in the console (sysout) – 5 ms
2. Open a file, and read the content of the files. Print them in console. (fileReader) – 400 ms
3. Get a user input for a primitive integer variable. (Scanner) – 50ms – 500ms
4. Read the contents of a database table (Employee - MySQL) and display it in the console (Conn, statement, rs) – 1000ms

Total duration to perform all operation in sequence = 5+400+500+1000 = 1905 ms

Total duration to perform all operations in parallel = Max(5,400,500,1000) = 1000ms

Multiple threads – Thread can able to perform multiple operation in parallel.

Multi- Thread Example --- MS-Word file. MSWORD is word processing application. Using MSWord software, we create word documents. MS-Word (Process- with PID – Process ID) – there are multiple threads. (Auto-Save Thread, spell-check thread, grammar check thread)

Thread – Light weight process

Process – Heavy weight thread

Multi Threading – Parallel processing.

Two ways of creating Thread in JAVA

1. Using Thread Class
2. Using Runnable Interface

Thread Life Cycle methods: start(), run(), sleep(), yield(), join(), suspend(), stop()

States of Thread : t1.getState()

* NEW (Newly created)
* RUNNABLE (Ready to execute)
* RUNNING (Currently performing some operations)
* BLOCKED (Other thread pre-empted)
* WAITING (Waiting for user input/operation)
* TERMINATE (Thread completed it’s execution)

Thread class will have 3 final values called MIN\_PRIORITY (1), NORM\_PRIORITY (5), MAX\_PRIORITY (10)

Thread Priority is a property of a thread which decides which thread will get CPU time quickly.

When there are multiple threads are waiting for the CPU time, then Scheduling algorithm will give more preference to the threads with higher priority.

Higher priority threads will pre-empt lower priority threads

When we create a thread, it will automatically takes it’s parent thread priority.

In Java, main method is also a Thread. It’s priority is 5. (NORM\_PRIORITY)

States Of A Thread

1. NEW
2. RUNNABLE (Ready to RUN)
3. RUNNING (Executing the RUN Method – when it gets CPU time)
4. NON-RUNNING (BLOCKED, WAITING, TIMED\_WAITING)
5. TERMINATED (Completed Execution/Opeartion)



Inside the JVM , the application memory is divided in to stack & heap

Stack – To maintain program reference variables, counters, method invocations (LIFO)

Heap – In this memory, all the objects will get created by the JVM

OutOFMemoryError – When heap area is full – no space to create new objects

StackOverflowError – When the stack area is full – no space to assign new reference variable or method reference

Java Automatically manage the memory.

Garbage Collection is one of the automatic memory management technique.

Garbage collection – Is a process of reclaiming the space of un-used or un-referenced objects

In other languages we have malloc(), alloc()

Agenda

1. Functional Interface
2. Lambda

Functional Interface – Newly introduced in JAVA 8.

What is Functional Interface – An Interface with only one abstract method is called functional Interface.

Runnable Interface is a Functional Interface (run() – abstract method)

Types of Interfaces

1. Marker Interface – An Interface with no methods (Serializable, cloneable)
2. Functional Interface – An interface with only one abstract method (Runnable,

In java 8, they are introduced a new annotation called @FunctionalInterface

In java 8, they also created a new package called java.util.function

All the interfaces available inside the package (java.util.function) is called functional interfaces.

Before Java 8, other than abstract method, we could not add any other methods

But In java 8, they introduced concepts of adding static & default methods to interface

100% abstraction could be achieved using interface.

But that is not true in Java8 and above version.

In Interface, by default all the methods are public and abstract. So abstract keyword is not necessary.

Inside class, we could not use default keyword for methods. – This will create compilation error.

Inside interface, we can use default keyword for methods – This will add default method in interface.

Functional Interface, can have n number of default methods and n number of static methods.

Lambda – Anonymous method (Nameless Method)

Public void add(int a, int b) //method signature line

{ //method body starts here

System.out.println(a+b);

} //method body ends here

Public void (int a, int b) {

…. }

Void (int a, int b) \_

---- }

(int a, int b) {

----}

(a,b) {

-----}

Lambda’s are oneline function.

(a,b)🡪{ }; // Lambda

Lambda with one argument/parameter === (n)🡪{System.out.println(n)};

n🡪System.out.println(n);

Lambda with two arguments/parameters ---- (a,b) 🡪{ System.out.println(a+b)};

(a,b) 🡪System.out.println(a+b);

Note :

1. We can’t create object of interface. We can create object reference of interface
2. We can’t create object of abstract Class.

FucntionalInterface reference can hold lambda reference.

FunctionalInterface reference can hold lambdas

Functional Interface is also called as one method interface

With the help of Lambda, we can achieve functional programming in java.

Performing multiple operations at the same time. – is called functional programming

This will have some intermediate operations, (Filtering, sorting & Searching)

Force Updating project – Select the project and press Alt+F5. (Force Update) --- Right click your project, select maven 🡪 Update Project (Click on Force Update check box)

Lambda’s can be stored in a functional interface reference.

()->{}; [basic lambda syntax]

In Lambda, () is optional for a single parameter Eg: n->System.out.println(n);

In Lambda, {} is optional for a single statement in body Eg: ()->System.out.println(“lambda”);

In Lambda, () is compulsory when we pass more than one argument Eg: (m,n)-> System.out.println(“m=”+m+” n=” +n);

In Lambda, {} is compulsory when we have more than one statement in the body

Eg: () -> {

int i=10;

while(i<=20){

System.out.println(i);

i++;

}

};