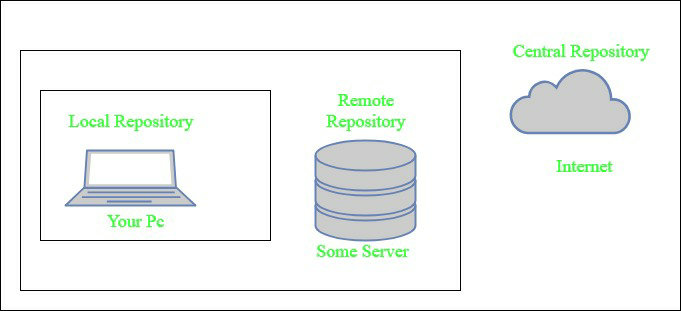
***Maven***

What is Maven and why it is used?

**Maven** is a powerful project management tool that is based on POM (project object model). It is **used** for projects build, dependency and documentation. It simplifies the build process.In short terms we can tell **maven** is a tool that can be **used**for building and managing any Java-based project.

Maven has three types of repository :

1. **Local repository**
2. **Central repository**
3. **Remote repository**



1. **Local repository-** A local repository is a directory on the machine of developer. This repository contains all the dependencies Maven downloads. Maven only needs to download the dependencies once, even if multiple projects depends on them (e.g. ODBC).  
   By default, maven local repository is user\_home/m2 directory.  
   example – **"C:\Users\Lenovo\.m2"**
2. **Central repository-** The central Maven repository is created Maven community. Maven looks in this central repository for any dependencies needed but not found in your local repository. Maven then downloads these dependencies into your local repository.
3. **Remote repository-** remote repository is a repository on a web server from which Maven can download dependencies.it often used for hosting projects internal to organization. Maven then downloads these dependencies into your local repository.

Maven does a lot of helpful task like

1. We can easily build a project using maven.
2. We can add jars and other dependencies of the project easily using the help of maven.
3. Maven can add all the dependencies required for the project automatically by reading pom file.
4. One can easily build their project to jar, war etc. as per their requirements using Maven.
5. Maven makes easy to start project in different environments and one doesn’t needs to handle the dependencies injection, builds, processing, etc.
6. Adding a new dependency is very easy. One has to just write the dependency code in pom file.

**Core Concepts of Maven:**

1. **POM Files:**Project Object Model(POM) Files are XML file that contains information related to the project and configuration information such as dependencies, source directory, plugin, goals etc. used by Maven to build the project. When you should execute a maven command you give maven a POM file to execute the commands. Maven reads pom.xml file to accomplish its configuration and operations.
2. **Dependencies and Repositories:**Dependencies are external Java libraries required for Project and repositories are directories of packaged JAR files. The local repository is just a directory on your machine hard drive. If the dependencies are not found in the local Maven repository, Maven downloads them from a central Maven repository and puts them in your local repository

**Maven pom.xml file**

POM means Project Object Model is key to operate Maven. Maven reads pom.xml file to accomplish its configuration and operations. It is an XML file that contains information related to the project and configuration information such as **dependencies**, **source directory**, **plugin**, **goals etc**. used by Maven to build the project.

What is POM XML in Maven?

A Project Object Model or **POM** is the fundamental unit of work in **Maven**. It is an **XML** file that contains information about the project and configuration details used by **Maven** to build the project. It contains default values for most projects.

The sample of pom.xml:-

<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>Obsqura</groupId>

<artifactId>FirstMaven</artifactId>

<version>0.0.1-SNAPSHOT</version>

<!-- Add typical dependencies for a web application -->

<dependencies>

<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>3.141.59</version>

</dependency>

</dependencies>

</project>

1. **dependencies-**used to defines a list of dependency of project.
2. **dependency-** used inside dependencies tag. Each dependency is described by its groupId, artifactId and version.

Dependencies can be downloaded from <https://mvnrepository.com/>

# **Maven Eclipse Example**

How to create maven project in eclipse.

In eclipse, click on File menu → New → Project → Maven → Maven Project → Tick the first check box (Create a simple project) → Next → Next → Next. Now write the group Id, artifact Id → finish.

Understand the following key concepts −

|  |  |
| --- | --- |
| **Sr.No.** | **Folder Structure & Description** |
| 1 | **GroceryStore Ecommerce**  contains src folder and pom.xml |
| 2 | **src/main/java**  contains java code files under the package structure (com/companyName/ecommerce). |
| 3 | **src/main/test**  contains test java code files under the package structure (com/companyName/ ecommerce). |
| 4 | **src/main/resources**  it contains images/properties files (In above example, we need to create this structure manually). |

**Maven Clean**

1. **Clean** Lifecycle
2. Its **clean**:cleangoal deletes the output of a **build** by deleting the **build** directory. Thus, when **maven clean** command

executes, **Maven** deletes the **build** directory.

**Apache Maven Install Plugin**

1. install: install is used to automatically install the project's main artifact (the JAR, WAR or EAR), its POM and any attached artifacts (sources, javadoc, etc) produced by a particular project.
2. install: install-file is mostly used to install an externally created artifact into the local repository, along with its POM.

**Excel read using Maven**

**Apache POI** (**POI** stands For **“Poor Obfuscation Implementation”)** is a popular API that allows programmers to create, modify, and display MS Office files using**Java** programs. It is an open source library developed and distributed by **Apache** Software Foundation to design or modify Microsoft Office files using **Java** program.

## Classes

**Workbook**  
It’s the super-interface of all classes that create or maintain Excel workbooks. Following are the two classes that implement this interface

1. **HSSFWorkbook**  
   It implements the Workbook interface and is used for Excel files in .xls format.
2. **XSSFWorkbook**  
   It is a class that is used to represent both high and low level Excel file formats(.xlsx also). It belongs to the org.apache.xssf.usemodel package and implements the Workbook interface.
3. **XSSFSheet**

It is a class It belongs to the org.apache.xssf.usemodel package and implements the Sheet interface.

1. **XSSFRow**

It is a class It belongs to the org.apache.xssf.usemodel package and implements the  
 Row interface.

5. **XSSFCell**

It is a class It belongs to the org.apache.xssf.usemodel package and implements the Cell interface.

Create Maven Project:

Click on File menu → New → Project → Maven → Maven Project → Tick the first check box (Create a simple project) → Next → Next → Next. Now write the group Id, artifact Id → finish.

Add following two dependencies of **Apache POI** in pom.xml file:

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi</artifactId>

<version>3.7</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>3.7</version>

</dependency>

If you still see an error, you can add these too:-

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-scratchpad -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-scratchpad</artifactId>

<version>3.7</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml-schemas -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml-schemas</artifactId>

<version>3.7</version>

</dependency>

Create a package and java class:-

package excelcode;

import java.io.FileInputStream;

import java.io.IOException;

import org.apache.poi.ss.usermodel.Cell;

import org.apache.poi.ss.usermodel.Row;

import org.apache.poi.xssf.usermodel.XSSFSheet;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class Excel {

XSSFSheet sh;

public Excel() throws IOException {

FileInputStream f= new FileInputStream("C:\\Users\\Lenovo\\Desktop\\ExcelFileToRead.xlsx");

XSSFWorkbook w= new XSSFWorkbook(f);

sh= w.getSheet("Sheet1");

}

public double readData(int i, int j) {

Row r= sh.getRow(i);

Cell c= r.getCell(j); //Import Cell of apache poi not Table

**return c.getNumericCellValue();** //Read integer data and return it as double

}

}

Create another main class :-

package excelcode;

import java.io.IOException;

public class ExcelMain {

public static void main(String[] args) throws IOException {

Excel ob= new Excel();

**double a=ob.readData(0,0);**

System.***out***.println("Value of a is "+a);

}

}

O/p:-

Value of a is 1234.0

**Read String Data**

Class 1:

package excelcode;

public class Excel {

XSSFSheet sh;

public Excel() throws IOException {

FileInputStream f= new FileInputStream("C:\\Users\\Lenovo\\Desktop\\ExcelFileToRead.xlsx");

XSSFWorkbook w= new XSSFWorkbook(f);

sh= w.getSheet("Sheet1");

}

public String readData(int i, int j) {

Row r= sh.getRow(i);

Cell c= r.getCell(j); //Import Cell of apache poi not Table

**int celltype=c.getCellType();**

**switch(celltype)**

**{**

**case Cell.*CELL\_TYPE\_NUMERIC*:**

**{**

**double a=c.getNumericCellValue();**

**return String.*valueOf*(a);**

**}**

**case Cell.*CELL\_TYPE\_STRING*:**

**{**

**return c.getStringCellValue();**

**}  
 }**

**return null;**

}}

Class 2:

package excelcode;

import java.io.IOException;

public class ExcelMain {

public static void main(String[] args) throws IOException {

Excel ob= new Excel();

**String a=ob.readData(0,0);**

**System.*out*.println("Value of a is "+a);**

**String b=ob.readData(0,1);**

**System.*out*.println("Value of b is "+b);**

}}

O/p:

Value of a is Obsqura

Value of b is 1234.0

Print all values by Using For loop

Class 1:

package excelcode;

public class Excel {

XSSFSheet sh;

public Excel() throws IOException {

FileInputStream f= new FileInputStream("C:\\Users\\Lenovo\\Desktop\\ExcelFileToRead.xlsx");

XSSFWorkbook w= new XSSFWorkbook(f);

sh= w.getSheet("Sheet1");

}

**public int rowsize() {**

**int noOfRows=sh.getLastRowNum()+1;**

**return noOfRows;**

**}**

**public String readData(int i, int j) {**

**Row r= sh.getRow(i);**

**Cell c= r.getCell(j); //Import Cell of apache poi not Table**

**int celltype=c.getCellType();**

**switch(celltype)**

**{**

**case Cell.*CELL\_TYPE\_NUMERIC*:**

**{**

**double a=c.getNumericCellValue();**

**return String.*valueOf*(a);**

**}**

**case Cell.*CELL\_TYPE\_STRING*:**

**{**

**return c.getStringCellValue();**

**}**

**}**

**return null;**

}

}

Class 2:

package excelcode;

public class ExcelMain {

public static void main(String[] args) throws IOException {

Excel ob= new Excel();

**for(int i=0;i<ob.rowsize();i++) {**

**for(int j=0;j<2;j++) {**

**String a=ob.readData(i,j);**

**System.*out*.println("Value of cell("+i+","+j+") is "+a);**

}

}

}

}

O/p:

Value of cell(0,0) is Obsqura

Value of cell(0,1) is 1234.0