INTRODUCTION TO MAVEN FUNDAMENTALS

COURSE OVERVIEW :

* In this course , we are going to cover :  
  Brief Introduction to Maven  
  We will see the difference ie the Pros and Cons of Maven vs Ant  
  Key Concepts of Maven , Convention over Configuration.  
  Day-to-Day programming concepts and how to intergrate Maven  
  Basic Intergration with IDE like Eclipse or STS  
  Maven can be a standalone tool also  
  Maven can be very complex –multimodule builds or tough structure concepts.
* Topics that we are going to cover  
  Introduction to Maven  
  Folder Structure  
  Dependencies-handling transitive dependencies  
  Repositories-local and corporate  
  Plugins-Every action we do within Maven is a plugin.  
  IDE Integration with Eclipse or STS

INTRODUCTION :

* This module is a basic introduction to Maven and what Maven is.
* We are going to do a High-Level overview of maven.
* Why you would want to use Maven.
* Dive a little deeper and check with Ant vs Maven vs IDE.
* Installation Best practices , how it should work , where to download it from.
* Little HelloWorld application.
* We will explore a pom file a little bit.
* A brief summary.

HIGH LEVEL OVERVIEW :

* Maven is at its simplest , A build tool.
* Tool to build source code and produce an artifact.
* Always produces one artifact(component,JAR,WAR,ZIP)
* Manages dependencies.
* If we are going to use Hibernate , one JAR is enough but it has other 13-14 transitive dependency JAR that also needs to be imported. Like commons-io , commons-lang.
* With Maven , it will download all of my transitive dependencies.
* At a macro level , it can be used a project management tool , it handles the versioning of your code.
* It has a version number associated with each component.
* Before we produce only 1 version of an artifact.
* We can use Maven to tell its 1.0,1.1…
* Documentation stored , who the developer are like meta information.
* I can easily produce all that extra information , I can produce JavaDocs , SourcesJAR file and other information my IDE will use.
* When I produce a library , Maven can generate all of that for me.
* Maven is managed by Apache Software Foundation.
* Maven sites are built with Maven
* Its open source. – maven.apache.org

WHY DO I WANT TO USE MAVEN :

* Repeatable builds – We can create our build for any environment.
* Dev-ops , Web-Ops. We need not change our settings for our external environment.
* We can develop on windows and test on linux or so.
* Transitive Dependencies – It will also pull other items that are related to it.
* It contains everything we need for the environment – To build or reproduce our code.
* It works with a local repo – we would have downloaded our JARs and kept them with our project , If suppose I had 20 projects , I would have those JARs 20 times , It works with a local repo , like a local folder structure and enables me to download it once and allows me to refer it as many times as I want.
* It works with our IDE , but it can also be standalone , I can build from the command line , is called from the IDE too. Its all derived from Maven files.
* It’s the preferred choice to work with Build tools like Jenkins , Cruise Control and so on. These are various automated build tools that does Continuous Intergration. There are also additional plugins we can use.

ANT vs MAVEN :

* Many people think that Ant and Maven are meant to compete against each other , they can be used in conjunction with each other , but they are solving two entirely different problems.
* Ant was developed as a replacement for the build tool called Make.
* Make was a cross platform tool.
* Ant was developed with Java and XML.
* Make was built around a UNIX environment and had problems with white spaces and such.
* Ant is very procedural , we have to go out of our way to use different pieces and have it stretch out to be able to use Composition inside your Ant Scripts.
* Its not much of a build tool , it’s a scripting tool.
* We have to do explicitly everything inside Ant.
* We have to call out what our targets are , am I going to chain goals.
* What if I like the word clean , someone else likes the word clear , someone else says cleanup.
* Its not standard. We have to remember what we have used.
* There is a lot of tribal knowledge built in Ant , everybody does it differently , there is no standard.
* Every organization ends up having a large repository of the scripts are unique to them.
* Not a lot of reuse , no inheritance.

MAVEN :

* Maven is full fledged build tool.
* It has a lot of implicit functionality , Maven clean is Maven clean.
* We get a lot of consistency across project , Inheritence like parent POM and such.
* We get transitive dependencies , if I pull a JAR , it pulls all dependant JAR.
* It can be achieved in ANT through Ivy. Ivy was not built to work with ANT but when people had invested much with ANT and didn’t wanna change , they started using Ivy to pull all those transitive dependencies.
* It is built around versioning , Its calling something as SNAPSHOT has some meaning behind it , its one of the key goals.
* Look in the folder for screenshot describing the pros and cons of Maven.
* Ant is quicker to learn , but is more copy paste oriented.
* Refer the screenshot for a snippet of a Ant Build File.xml
* We have the following  
  clean – deletes the build directory  
  compile – checks our directory whether it exists and then compiles our code into the destination directory  
  jar- Take the build directory and JAR that up.  
  The problem would be that I could execute any command up first and that would be really bad , because if suppose I have a JAR first without setting any goals , then it would contain information that’s not even relevant to my application.  
  It’s a very brittle file.
* I know personally that I have to call   
  Ant-clean  
  Ant-compile  
  Ant-jar  
  Ant-run for my application to work.
* It’s a false sense of security.

MAVEN POM FILE :

* In a Maven Pom file , I can do all that I ran in a ANT file in the application.
* Its completely like a black box , because I don’t need to define anything if I follow Mavens conventions and structure.
* It’s a little built non constructive at first.

SUMMARY FOR THE ABOVE TOPICS :

* Ant is very declarative , I have to lay everything out , shorter learning curve.
* Maven follows a conventions over configuration model.
* Ant is easier to learn but its like a scripting tool.
* Maven is all about managing your entire project life cycle , like versions ,structure our code and such.

BEST PRACTICES INSTALLATION :

* We are going to see some best practices of Maven Installation and how to do it , also set it up with a project.
* Go to maven.apache.org
* Download the bin.zip extension.
* Click on the mirror to download.
* Extract Maven Folder to a directory.
* apache-maven-3.0.4
* DC on that folder.
* Start -> Control Panel -> Environment variables -> New
* Variable Name : JAVA\_HOME
* Variable value : Copy the path of jdk and paste it.
* Ensure that we are doing it in System Variable and not User Variable.
* New
* Variable Name : MAVEN\_HOME
* Variable Value : Copy the path of maven extraction and paste it.
* Scroll down in System Variables to Path and click on EDIT.
* Variable Value : %JAVA\_HOME%\bin;%MAVEN\_HOME%\bin;
* Click on OK.
* Click on Start.
* Command Prompt. -> mvn –version
* Shows where my Maven home is , Java home is , Java version is etc.
* Now everything is set up with Maven and Java , Now we can go on to do our HelloWorld program.

HELLO WORLD DEMO :

* In this demo we will be developing a HelloWorld application and we will be using Maven to build that application.
* Open up Spring STS
* RC on Package Explorer
* New -> Project -> General project
* Project Name : HelloWorld -> Finish
* RC on Project and say New->File
* File Name : pom.xml -> Maven looks only for this file while building the application
* Since its an XML document , it has to be well formed and well structured.
* Add the following  
    
    
    
    
    
    
  <project>  
  <groupId>com.pluralsight</groupId> -- It would describe which company Eg: org.hibernate

<artifactId>HelloWorld</artifactId> -- Project name  
<version>1.0-SNAPSHOT</version>  
<modelVersion>4.0.0</modelVersion> -- Represents the schema version we are using XML   
<packaging>jar</packaging>  
</project>

* Now we can write a Java Application to test this POM here.
* Maven expects a standard folder structure.
* RC on Project -> New -> Folder -> src as the name of the folder -> Ok
* RC on src -> New -> Folder -> main as the name of the folder -> Ok
* RC on main -> New -> Folder -> java as the name of the folder -> Ok
* RC on java -> New -> File -> HelloWorld.java
* Add the following to HelloWorld.java  
  public class HelloWorld   
  {  
  public static void main(String[] args)  
  {  
  System.out.println(“HelloWorld”);  
  }  
  }
* Open the command prompt
* Type the following  
  cd \  
  cd workspace  
  inside workspace  
  cd HelloWorld  
  dir  
  mvn clean  
  mvn compile  
  Once compile is done , I will have a target directory  
  java HelloWorld  
  cd ..  
  mvn package  
  Its created a jar in the target directory.
* It produced a jar which says , HelloWorld was our artifactId , 1.0-SNAPSHOT was our version and JAR was our packaging type.

SUMMARY :

* Lets do a brief summary of what we covered in this module.
* We covered what is maven
* How it is different from other scripting tools like ANT and MAKE.
* Its not just a scripting tool.
* Where to get it and how to install it.
* We finally did a HelloWorld application and built it using Maven.