# Trabajo Práctico 2 - H. de C. de SW

## **1- Objetivos de Aprendizaje**

* Utilizar herramientas de construcción de software y manejo de paquetes y dependencias
* Familiarizarse con las herramientas más utilizadas en el lenguaje Java.

## **2- Unidad temática que incluye este trabajo práctico**

Este trabajo práctico corresponde a la unidad Nº: 4 (Libro Continuous Delivery: Cap 6 y 13)

## **3- Consignas a desarrollar en el trabajo práctico:**

* Las aplicaciones utilizadas son del tipo "Hello World", dado que el foco del trabajo práctico es cómo construirlas y no el funcionamiento de la aplicación en sí.
* La mayoría de los ejercicios se realizarán en clase con asistencia del Jefe de trabajos prácticos.
* En los puntos en los que se pida alguna descripción, realizarlo de la manera más clara posible.

## **4- Desarrollo:**

#### **1- Instalar Java JDK si no dispone del mismo.**

* Java 8 es suficiente
* Utilizar el instalador que corresponda a su sistema operativo
* <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>
* Agregar la variable de entorno JAVA\_HOME
  + En Windows temporalmente se puede configurar
* SET JAVA\_HOME=C:\Program Files\Java\jdk1.8.0\_181
  + O permanentemente entrando a Variables de Entorno (Winkey + Pausa -> Opciones Avanzadas de Sistema -> Variables de Entorno)
* Otros sistemas operativos:
  + <https://www3.ntu.edu.sg/home/ehchua/programming/howto/JDK_Howto.html>
  + <https://www.digitalocean.com/community/tutorials/how-to-install-java-with-apt-on-ubuntu-18-04>

#### **2- Instalar Ant**

* Descargar Ant (archivo apache-ant-1.10.5-bin.zip) desde <https://ant.apache.org/bindownload.cgi>
* Descomprimir en una carpeta, por ejemplo C:\tools
* Agregar el siguiente directorio a la variable de entorno PATH, asumiendo que los binarios de Ant están en C:\tools\apache-ant-1.10.5\bin

SET PATH=%PATH%;C:\tools\apache-ant-1.10.5\bin

* Se puede modificar permanentemente la variable de sistema PATH entrando a (Winkey + Pausa -> Opciones Avanzadas de Sistema -> Variables de Entorno)
* En Linux/Mac se puede agregar la siguiente entrada a ~/.bash\_profile

export PATH=/opt/apache-ant-1.10.5/bin:$PATH

#### **3- Introducción a Ant**

* Explicar brevemente qué es Ant:

**Apache Ant** es una herramienta usada en programación para la realización de tareas mecánicas y repetitivas, normalmente durante la fase de compilación y construcción (build). Es, por tanto, un software para procesos de automatización de compilación, similar a Make pero desarrollado en lenguaje Java y requiere la plataforma Java, así que es más apropiado para la construcción de proyectos Java. Esta herramienta, hecha en el lenguaje de programación Java, tiene la ventaja de no depender de las órdenes del shell de cada sistema operativo, sino que se basa en archivos de configuración XML y clases Java para la realización de las distintas tareas, siendo idónea como solución multi-plataforma. La diferencia más notable entre Ant y Make es que Ant utiliza XML para describir el proceso de generación y sus dependencias, mientras que Make utiliza formato makefile. Por defecto, el archivo XML se denomina *build.xml*. Ant es un proyecto de la Apache Software Foundation. Es software open source, y se lanza bajo la licencia Apache Software.

* Analizar el contenido del ./trabajo-practico-02/ant/build.xml
* Ejecutar los siguientes comandos desde la consola:

ant all  
ant javadoc

* Comentar que se obtiene con los mismos.
* Agregar al build.xml la posibilidad de generar archivo zip con la salida del proceso de build. En el target dist y usando en AntTask Zip.
* Sacar conclusiones.

#### **3- Instalar Maven**

* Instalar maven desde <https://maven.apache.org/download.cgi> (última versión disponible 3.4.5)
* Descomprimir en una carpeta, por ejemplo C:\tools
* Agregar el siguiente directorio a la variable de entorno PATH, asumiendo que los binarios de ant estan en C:\tools\apache-maven-3.5.4\bin
* SET PATH=%PATH%;C:\tools\apache-maven-3.5.4\bin
* Se puede modificar permanentemente la variable PATH entrando a (Winkey + Pausa -> Opciones Avanzadas de Sistema -> Variables de Entorno)
* En Linux/Mac se puede agregar la siguiente entrada al archivo ~/.bash\_profile
* export PATH=/opt/apache-maven-3.5.4/bin:$PATH

#### **4- Introducción a Maven**

* Qué es Maven?

**Maven** es una herramienta de software para la gestión y construcción de proyectos Java creada por Jason van Zyl, de Sonatype, en 2002. Es similar en funcionalidad a Apache Ant (y en menor medida a PEAR de PHP y CPAN de Perl), pero tiene un modelo de configuración de construcción más simple, basado en un formato XML. Estuvo integrado inicialmente dentro del proyecto Jakarta pero ahora ya es un proyecto de nivel superior de la Apache Software Foundation.  
  
Maven utiliza un Project Object Model (POM) para describir el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Viene con objetivos predefinidos para realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado.  
  
Una característica clave de Maven es que está listo para usar en red. El motor incluido en su núcleo puede dinámicamente descargar plugins de un repositorio, el mismo repositorio que provee acceso a muchas versiones de diferentes proyectos Open Source en Java, de Apache y otras organizaciones y desarrolladores. Este repositorio y su sucesor reorganizado, el repositorio Maven 2, pugnan por ser el mecanismo de facto de distribución de aplicaciones en Java, pero su adopción ha sido muy lenta. Maven provee soporte no solo para obtener archivos de su repositorio, sino también para subir artefactos al repositorio al final de la construcción de la aplicación, dejándola al acceso de todos los usuarios. Una caché local de artefactos actúa como la primera fuente para sincronizar la salida de los proyectos a un sistema local.  
  
Maven está construido usando una arquitectura basada en plugins que permite que utilice cualquier aplicación controlable a través de la entrada estándar. En teoría, esto podría permitir a cualquiera escribir plugins para su interfaz con herramientas como compiladores, herramientas de pruebas unitarias, etcétera, para cualquier otro lenguaje. En realidad, el soporte y uso de lenguajes distintos de Java es mínimo. Actualmente existe un plugin para .Net Framework y es mantenido,​ y un plugin nativo para C/C++ fue alguna vez mantenido por Maven 1.

* Qué es el archivo POM? (*En la definición de Maven*)
  + modelVersion

Declares to which version of project descriptor this POM conforms.

* + groupId

A universally unique identifier for a project. It is normal to use a fully-qualified package name to distinguish it from other projects with a similar name (eg. org.apache.maven).

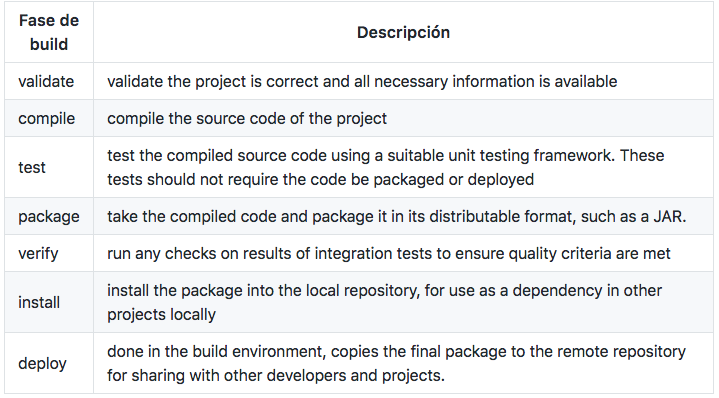
* + artifactId

The identifier for this artifact that is unique within the group given by the group ID. An artifact is something that is either produced or used by a project. Examples of artifacts produced by Maven for a project include: JARs, source and binary distributions, and WARs.

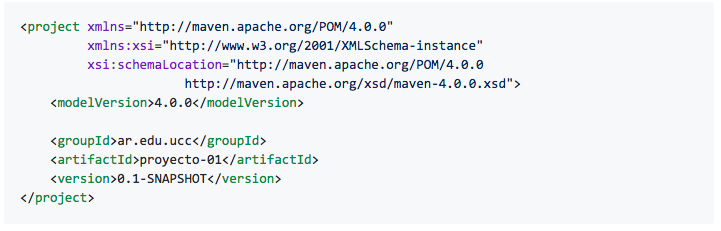
* + versionId

The current version of the artifact produced by this project.

* Repositorios Local, Central y Remotos <http://maven.apache.org/guides/introduction/introduction-to-repositories.html>
* Entender Ciclos de vida de build
  + default
  + clean
  + site
  + Referencia: <http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html#Build_Lifecycle_Basics>
* Comprender las fases de un ciclo de vida, por ejemplo, default:



* Copiar el siguiente contenido a un archivo, por ejemplo ./trabajo-practico-02/maven/vacio/pom.xml



* Ejecutar el siguiente comando en el directorio donde se encuentra el archivo pom.xml

mvn clean install

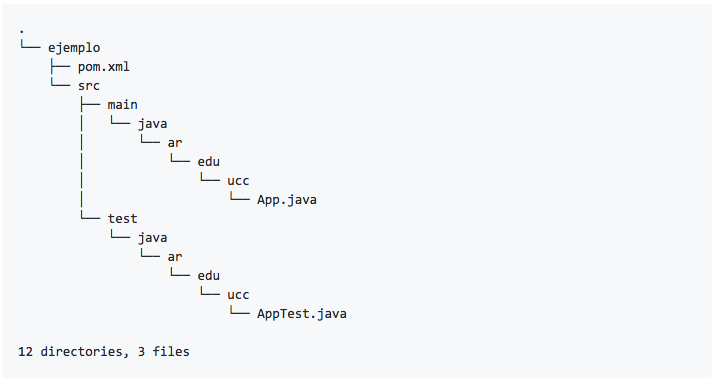
* Sacar conclusiones del resultado

#### **5- Maven Continuación**

* Generar un proyecto con una estructura inicial:

mvn archetype:generate -DgroupId=ar.edu.ucc -DartifactId=ejemplo -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

* Analizar la estructura de directorios generada:



* Compilar el proyecto

mvn clean package

* Analizar la salida del comando anterior y luego ejecutar el programa

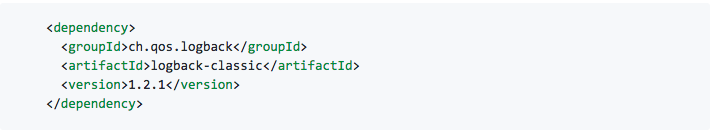
java -cp target/ejemplo-1.0-SNAPSHOT.jar ar.edu.ucc.App

#### **6- Manejo de dependencias**

* Crear un nuevo proyecto con artifactId ejemplo-uber-jar
* Modificar el código de App.java para agregar utilizar una librería de logging:



* Compilar el código e identificar el problema.
* Agregar la dependencia necesaria al pom.xml



* Verificar si se genera el archivo jar y ejecutarlo

java -cp target\ejemplo-uber-jar-1.0-SNAPSHOT.jar ar.edu.ucc.App

* Sacar conclusiones y analizar posibles soluciones
* Implementar la opción de uber-jar: <https://maven.apache.org/plugins/maven-shade-plugin/>



* Volver a generar la salida y probar ejecutando

java -jar target\ejemplo-uber-jar.jar

#### **7- Ejemplo con nodejs (Opcional, pero recomendando)**

* Instalar Nodejs: <https://nodejs.org/en/>
* Instalar el componente para generar aplicaciones Express

npm install express-generator -g

* Crear una nueva aplicación

express --view=ejs hola-mundo

* Ejecutar la aplicación

cd hola-mundo  
npm install  
npm start

* La aplicación web estará disponible en [http://localhost:3000](http://localhost:3000/)
* Analizar el manejo de paquetes y dependencias realizado por npm.

#### **8- Presentación**

* Subir todo el código, ejemplos y respuestas a una carpeta trabajo-practico-02.

Tip: Agregar un archivo .gitignore al repositorio para evitar que se agreguen archivos que son resultado de la compilación u otros binarios, que no son necesarios, al mismo.