Herramientas para el desarrollo de aplicaciones

Que es Java:

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "write once, run anywhere"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados.1 2.

Plataformas de Java:,

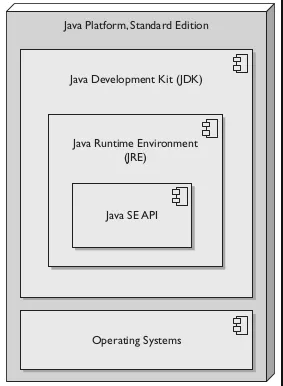
Java Platform, Standard Edition (J2SE)

Java Platform, Micro Edition (J2ME)

Java Platform, Enterprise Edition (J2EE)

Java Platform, Standard Edition:

Está compuesta por la API mas rica en contenido, la Java SE API, Java Runtume Environment, Java Development Kit. Cada uno de estos elementos sirve para un propósito especial. Colectivamente, estos componentes proveen el medio de desarrollar, desplegar y ejecutar sistemas basados en cliente, cliente-servidor.



Java SE API

Es una colección de paquetes de software. Cada paquete guarda un set de clases e interfaces. Todos los paquetes que deberemos conocer los veremos en la tabla que viene a continuación. La mayoría de estos paquetes tienen subpaquetes que no están listados:



Java Runtime Environment (JRE)

Es el set de software que permite a las aplicaciones en Java ser ejecutadas. El JRE incluye los siguientes items:

Java Virtual Machine(JVM)

Java Hotspot Client VIrtual Machine

Java Hotspot Server Virtual Machine

Deployment technologies

Java Plug-in

Java Web Start Technology

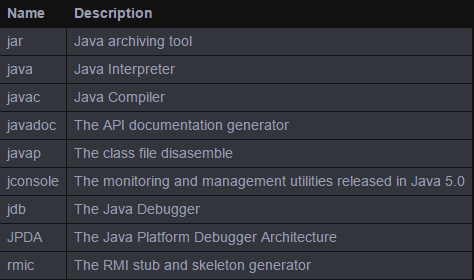
Java Control Panel

Java Update Mechanism

Java es compilado en byte code, y cada SO tiene su propia JVM que ejecutará el Byte code. Esto es lo que el da la capacidad WORA(Write one, run anywhere)

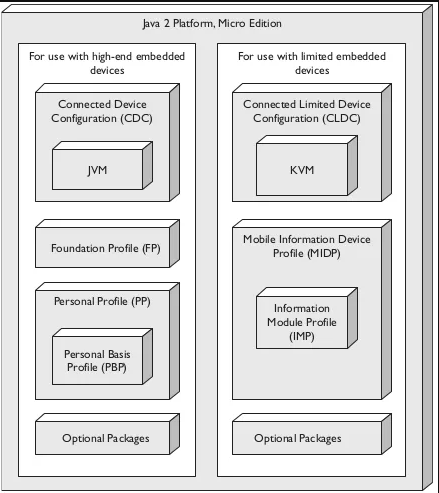
Java Development Kit (JDK)

Es la caja de herramientas del desarrollador por así decirlo. Contiene todas las utilidades que necesitaremos para compilar, probar, depurar y ejecutar nuestras aplicaciones. En la siguiente tabla evmos las herramientas mas comunes:



Java Platform, Micro Edition

Está diseñado para pequeños dispositivos integrados como PDA y teléfonos móviles. la arquitectura de Java ME está basada en configuraciones, perfiles y paquetes opcionales:



Java Platform, Enterprise Edition

Provee el medio de crear verdaderos sistemas para empresas que son flexibles, escalables y seguros. El mayor beneficio de estos sistemas es la separación de los componentes del software. Java EE sigue el Modelo-Vista-Controlador (MVC) donde los servlets trabajan como controlador, JavaServer maneja la vista opresentación lógica, y la lógica de negocio es representado como modelo tipicamente Enterprise JavaBeans(EJBs). Java Enterprise Edition requiere una colección de paquetes opcionales que soportan cada una de estas tareas, tambien como tecnologías complementarias. Los paquetes son actualmente implementaciones de especificaciones. Las APIs que describiremos en este capítulo y en los 2 siguientes serán:

Enterprise JavaBeans 2.1

Servlet 2.4

Java Server Pages 2.0

Java Message Service 1.1

JavaMail 1.3

Web Services 1.1

JAX RPC 1.1

Cuando se desarrollan sistemas con Java EE, necesitaremos siempre el JDK. Aunque el JDK es una peiza principal de la plataforma Java SE, podremos decir que Java SE es parte de Java EE. A menudo tendremos la opción de usar las versiones mas nuevas del JDK con la implementación Java EE de nuestra elección. Tenemos que ser conscientes que no hay correlación directa entre los números de Java SE y Java EE. Tendremos que ver la documentación para ver que versiones del JDK trabajarán con nuestra implementación de Java EE.

Java Development Kit

Java Development Kit o (JDK), es un software que provee herramientas de desarrollo para la creación de programas en Java. Puede instalarse en una computadora local o en una unidad de red. En la unidad de red se pueden tener las herramientas distribuidas en varias computadoras y trabajar como una sola aplicación.

JDK en Windows[editar]

En los sistemas operativos Microsoft Windows sus variables de entorno son:

JAVAPATH: es una ruta completa del directorio donde está instalado JDK.

CLASSPATH: son las bibliotecas o clases de usuario.

PATH: variable donde se agrega la ubicación de JDK.

Los programas más importantes que se incluyen son:

appletviewer.exe: es un visor de applets para generar sus vistas previas, ya que un applet carece de método main y no se puede ejecutar con el programa java.

javac.exe: es el compilador de Java.

java.exe: es el masterescuela (intérprete) de Java.

javadoc.exe: genera la documentación de las clases Java de un programa.

Java Archive:

Un archivo JAR (por sus siglas en inglés, Java ARchive) es un tipo de archivo que permite ejecutar aplicaciones escritas en el lenguaje Java. Las siglas están deliberadamente escogidas para que coincidan con la palabra inglesa "jar" (tarro). Los archivos JAR están comprimidos con el formato [ZIP](https://es.wikipedia.org/wiki/Formato_de_compresi%C3%B3n_ZIP) y cambiada su extensión a .jar. Existen tres operaciones básicas con este tipo de archivos: ver contenido, comprimir y descomprimir.

Instrucciones:

Ver contenido: $ jar -tvf archivo.jar

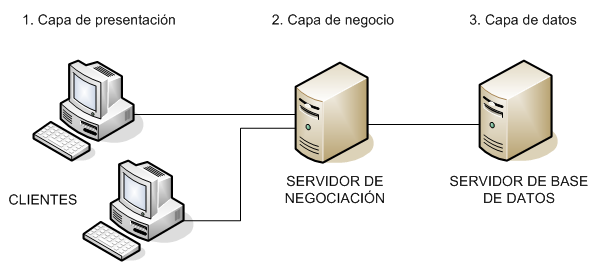
Comprimir: $ jar -cvf archivo.jar archivo\_1 archivo\_2 archivo\_3 ... archivo\_n

Descomprimir: $ jar -xvf archivos.jar

Un vínculo de descarga a .jar es .jad, y pesa muy poco. Estos archivos se utilizan normalmente para aplicaciones en teléfonos móviles.

ARQUITECTURA 3 CAPAS PROGRAMACIÓN POR CAPAS

Es un estilo de programación, su objetivo primordial es la separación de la capa de presentación, capa de negocio y la capa de datos.



La arquitectura de una aplicación es la vista conceptual de la estructura de esta. Toda aplicación contiene código de presentación, código de procesamiento de datos y código de almacenamiento de datos. La arquitectura de las aplicaciones difiere según como está distribuido este código.

Arquitectura de Capas

En toda arquitectura de capa los elementos agrupados en una misma capa pueden comunicarse entre si; pero existen variantes en cuanto a las comunicaciones permitidas entre elementos de capas diferentes:

Arquitectura top-down de capas:

Los elementos de una capa i+1 pueden enviar solicitudes de servicio a elementos de la capa inferior i. Típicamente se produce una cascada de solicitudes, es decir para satisfacer una solicitud a una capa i+2, ésta requiere enviar varias solicitudes a la capa i+1; cada una de estas solicitudes a la capa i+1 genera a su vez un conjunto de solicitudes a la capa i y así sucesivamente. Una arquitectura top-down es laxa (o no estricta) si los elementos de una capa i+1 pueden enviar solicitudes de servicio directamente a un elemento de cualquiera de las i capas inferiores.

Arquitectura bottom-up de capas:

Cada elemento de una capa i puede notificar a elementos de la capa superior i+1 de que ha ocurrido algún evento de interés (ej. manejadores de dispositivos). La capa i+1 puede juntar varios eventos antes de notificar a su vez an elemento de la capa i+2. Una arquitectura bottom-up tambien puede ser no estricta si el elemento de la capa i puede notificar a cualquier elemento de cualquier capa superior a la capa i.

Arquitectura bidireccional de capas

En su forma más común involucra dos pilas de N capas que se comunican entre si. El ejemplo más conocido es el de los protocolos en Redes de Computadores.

La programación por capas se refiere a un estilo de programación que tiene como objetivo separar la lógica de diseño de la lógica de negocios. Una de las ventajas que podemos destacar sobre este estilo es que el desarrollo del software se puede llevar a cabo en varios tipos de niveles, así, cuando suceda algún cambio solo nos iremos sobre el nivel requerido.

La programación por capas es una técnica de la ingeniería del software propia de la programación a objetos, que se divide en 3 capas: la capa de presentación o frontera, la capa de lógica de negocio y por ultimo la capa de datos.



Capa de presentación

Se refiere a la presentación del programa frente al usuario, esta presentación debe cumplir su propósito con el usuario final, una presentación fácil de usar y amigable. También las interfaces deben ser consistentes con la información dentro del software (Por ejemplo; en los formularios no debe haber más que lo necesario), tomar en cuenta los requerimientos del usuario, la capa de presentación va de la mano con capa de la lógica de negocio.

Capa de lógica de negocio

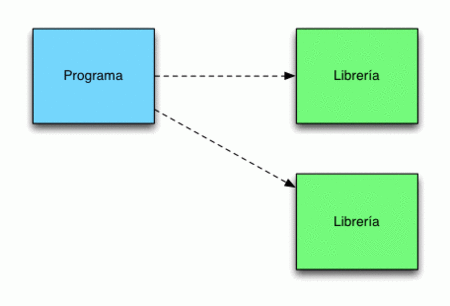
En esta capa es donde se encuentran los programas que son ejecutados, recibe las peticiones del usuario y posteriormente envía las respuestas tras el proceso. Esta capa es muy importantes pues es donde se establecen todas aquellas reglas que se tendrán que cumplir, decía anteriormente que la capa de presentación tiene comunicación con la capa de lógica de negocio ya que se tienen que comunicar para recibir las solicitudes y presentar los resultados.

Capa de datos

Esta capa es la que se encarga de hacer las transacciones con la base de datos y con otros sistemas para descargar o insertar información al sistema. La consistencia en los datos es sumamente importante, es decir, los datos que se ingresan o insertan deben ser precisos y consientes. Aquí definimos las consultas que vamos a realizar en la base de datos, o consultas para reporteo. La comunicación de esta capa con la capa de lógica de negocio se refiere a que la capa de datos es la que le enviara información a la capa de negocio para que sea procesada e ingresada en objetos según sea necesario (encapsulamiento).

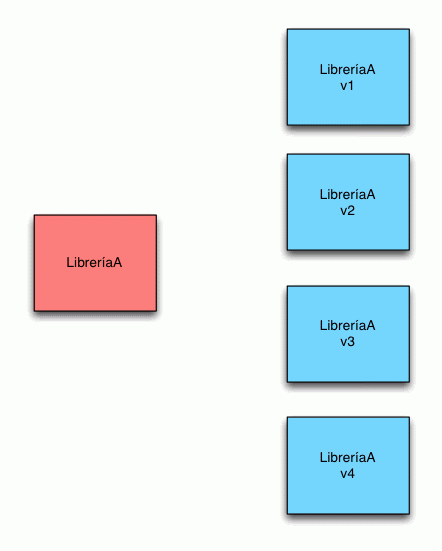
MAVEN

Normalmente cuando nosotros trabajamos con Java/JavaEE el uso de librerías es algo común como en cualquier otro lenguaje de programación

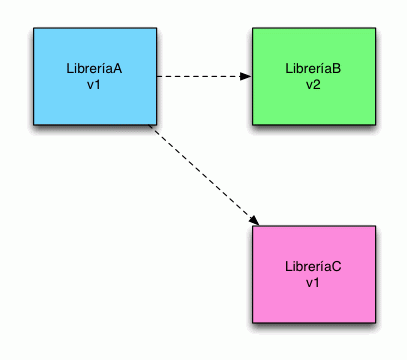


Librerías y limitaciones

El concepto de librería es un concepto que a veces es limitado. Por ejemplo nosotros podemos querer utilizar la librería A en nuestro proyecto. Sin embargo no nos valdrá con simplemente querer utilizar la librería sino que además necesitaremos saber que versión exacta de ella necesitamos.

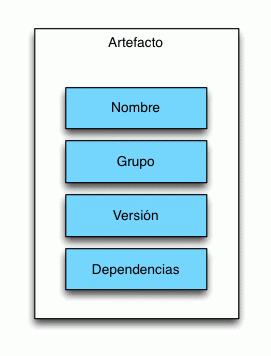


¿Es esto suficiente?. Lamentablemente no lo es, una librería puede depender de otras librerías para funcionar de forma correcta. Así pues necesitamos más información para gestionarlo todo de forma correcta.



Maven y Artefactos

Maven solventa este problema a través del concepto de Artefacto. Un Artefacto puede verse como una librería con esteroides (aunque agrupa más conceptos). Contiene las clases propias de la librería pero además incluye toda la información necesaria para su correcta gestión (grupo, versión, dependencias etc).



Artefactos y POM

Para definir un Artefacto necesitamos crear un fichero POM.xml (Proyect Object Model) que es el encargado de almacenar toda la información que hemos comentado anteriormente:

<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>com.genbetadev.proyecto1</groupId>

<artifactId>proyecto1</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<dependencies>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.17</version>

</dependency>

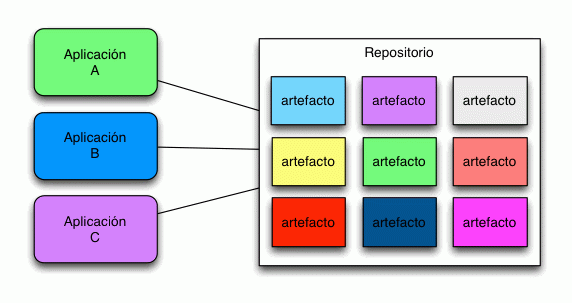
</dependencies>

</project>

La estructura del fichero puede llegar a ser muy compleja y puede llegar a depender de otros POM. En este ejemplo estamos viendo el fichero más sencillo posible. En él se define el nombre del Artefacto (artifactID) el tipo de empaquetado (jar) y también las dependencias que tiene (log4j). De esta manera nuestra librería queda definida de una forma mucho más clara.

Maven Repositorio y Artefactos

Una vez definidos correctamente todos los Artefactos que necesitamos, Maven nos provee de un Repositorio donde alojar, mantener y distribuir estos. Permitiéndonos una gestión correcta de nuestra librerías, proyectos y dependencias.



Preguntas:

* Que es lo primero que se debe instalar para ejecutar una aplicación en java
* Se debe tener un entorno de programación como (eclipse o netbeans) para ejecutar una aplicación en java
* Cuál es la diferencia de usar librerías (archivos .zip o .jar) descargándolas manualmente y usar maven
* Instalar eclipse y mave, crear el primero proyecto Java general
* Investigar los conceptos básicos de hsqldb