




Documentación Java






Contenido

-  Introducción a Java
-  Ediciones
-  Empaquetados

Introducción a Java, ¿qué es Java?

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. Hay muchas aplicaciones y sitios web que no funcionarán a menos que tenga Java instalado y cada día se crean más. Java es rápido, seguro y fiable. Desde portátiles hasta centros de datos, desde consolas para juegos hasta súper computadoras, desde teléfonos móviles hasta Internet, Java está en todas partes. Es uno de los lenguajes de programación más potentes, conocidos y usados en todo el mundo.

Ediciones

-  **Java Micro Edition (Java ME)**
-  **Java Standard Edition (Java SE)**
-  **Java Enterprise Edition (Java EE)**

➤ **Java Micro Edition (Java ME)**

Es una versión reducida de la edición Java Standard Edition de la que hablaremos más adelante. Esta edición está enfocada para la creación de aplicaciones tanto en dispositivos móviles como dispositivos integrados.

Con Java ME podemos desarrollar aplicaciones para diferentes dispositivos, no limitándonos únicamente a teléfonos inteligentes. Si así lo deseamos, podemos crear aplicaciones para Televisores inteligentes, consolas de videojuegos, etc.

Su popularidad se vio afectada con la llegada de Android aunque hoy en día está retomando terreno principalmente por el tema del Internet de las cosas.

➤ **Java Standard Edition (Java SE)**

Es la edición estándar de Java, la versión original de Sun Microsystems. Con esta versión podemos crear tanto aplicaciones web como aplicaciones de escritorio.

Esta edición cuenta con una amplia biblioteca de clases pensadas para agilizar el desarrollo. Disponemos de clases enfocadas en seguridad, red, acceso a base de datos, interfaces gráficas, conexión entre dispositivos, XML, etc.

A la hora de empezar a programar en este lenguaje es recomendable comenzar desde esta edición ya que será la que proporcione una base sólida de aprendizaje tocando temas como **Java Virtual Machine**, **Java Runtime Environment**, **Java Development Kit** y **API** de Java. A continuación, vamos a explicar brevemente cada uno de estos elementos:

❖ **Java Virtual Machine:** Java es un lenguaje compilado. Cuando compilamos nuestras aplicaciones, obtenemos Bytecode que es un conjunto de instrucciones altamente optimizado diseñadas para ser ejecutadas por el sistema de tiempo de ejecución Java, también conocido como Java Virtual Machine o JVM. Es un componente esencial ya que se encarga de que la aplicación sea escrita una sola vez y ejecutada las veces que el usuario desee en distintos dispositivos. **(Write Once Run Anywhere).**

Cada sistema operativo necesita su propia implementación de esta herramienta ya que de lo contrario no sería posible ejecutar aplicaciones Java.

❖ **Java Runtime Environment:** es un conjunto de herramientas que proporcionan un entorno en donde las aplicaciones Java pueden ser ejecutadas. Cuando queremos ejecutar nuestras aplicaciones, Java debe elegir el entorno que mejor se adecue a las necesidades de dicha aplicación debiendo elegir arquitectura y sistema operativo.

❖ **Java Development Kit:** es una extensión de JRE. Junto a los archivos y herramientas que proporcionados por JRE, el JDK proporciona compiladores y herramientas para crear programas Java. Por esta razón, cuando uno quiere desarrollar una aplicación Java, necesita instalar un JDK.

❖ **API de Java:** Java SE provee una amplia biblioteca de clases las cuales están pensadas para agilizar nuestro proceso de desarrollo, son clases las cuales ya vienen con el propio lenguaje.

A esta biblioteca de clases se le denomina API de Java y puede ser consultada de forma gratuita en la documentación oficial de Java.

➤ Java Enterprise Edition (Java EE)

Es la edición más grande de Java. Contiene toda la Standard Edition y mucho más. Por lo general se utiliza para crear aplicaciones con la arquitectura cliente servidor.

Fue pensado para el mundo empresarial. Posee una amplia biblioteca de clases con las cuales podemos trabajar con JSON, Email, base de datos, transacciones, persistencia, envío de mensajes, etc.

Cuando ya hemos establecido una buena base de Java se aborda el mundo de las Bases de Datos, el modelado de datos, el modelo relacional y como acceder e interrogar al sistema desde Java.

Empaquetados

➤ JAR (Java Archives)

Es un formato desarrollado por "Sun" que permite agrupar las clases diseñadas en el lenguaje Java, este formato es ampliamente utilizado en ambientes Java de todo tipo, esto se debe a que otorga un nivel de compresión y reduce la carga administrativa al distribuir clases en el lenguaje.

Debido a la compresión de estos archivos JAR estos no pueden ser observados ni manipulados directamente, sin embargo, existe la utilidad jar proporcionada con todo JDK que permite observar el contenido.

➤ WAR (Web-Archives)

Cuando uno recién registra su sitio en Internet y empieza a desarrollar aplicaciones de servidor, estas son relativamente sencillas. Si se utilizan JSP's seguramente el sitio será bastante complicada de entender, es por esto que los WARS **modularizan** el desarrollo de aplicaciones.

Al utilizar WARS basta con colocar los archivos bajo un directorio y el Servlet Engines terminará el trabajo. El uso de WARS también facilita la distribución de esta aplicación a otros "Servlet Engines" utilizando un solo archivo.

Estructura de un archivo WAR (Web-Archive)

La estructura de un (Web-Archive) WAR es la siguiente:

/*.html *.jsp *.css : Este directorio base contiene los elementos que comúnmente son utilizados en un sitio, Documentos en HTML, JSP's , CSS("Cascading Style Sheets"), etc.

/WEB-INF/web.xml : Contiene elementos de seguridad de la aplicación así como detalles sobre los Servlets que serán utilizados dentro de la misma.

/WEB-INF/classes/ : Contiene las clases Java adicionales a las del JDK que serán empleadas en los JSP's y Servlets

/WEB-INF/lib/ : Contiene los JAR's que serán utilizados por su aplicación.

La creación de WARS puede realizarse en IDE's o en algún otro programa como Ant.

➤ **EJB-JAR**

Es la manera en la que son distribuidos desarrollos EJB. De la misma forma en que los WARS modularizan el desarrollo de aplicaciones JSP/Servlets, un EJB-JAR lo hace para desarrollos EJB.

La estructura de un EJB-JAR es la siguiente:

- **/*.class** : Bajo este directorio base se encuentran las diversas *clases* que conforman a un EJB.
- **/META-INF/ejb-jar.xml** : Este archivo contiene el denominado *Deployment Descriptor* utilizado en EJB's.
- **/META-INF/*** : Este directorio además del *Deployment Descriptor* puede contener otros archivos de configuración utilizados por el "Application Server" ("EJB Container" para ser más exacto).

Al igual que los WAR's ("Web-Archives") se *supone* que esta estructura no debe variar, pero obviamente existen algunas discrepancias entre los diversos "Application Servers", por lo que un *EJB-JAR* creado para JBoss probablemente no funcione para Web-Logic.

➤ **EAR (Enterprise Archives)**

Un EAR es simplemente un WAR y EJB-JAR agrupados. La razón de su existencia se debe básicamente a la estructura de un "Java Application Server". Por lo general, ningún vendedor de "Application Servers" hace una clara distinción entre el "Servlet Engine" (ambiente utilizado para WAR's) y el "EJB Container" (utilizado para EJB-JAR's) ,debido a esto se generó el termino EAR, que desde luego también sigue estando en el formato JAR.

Por último, al igual que en las estructuras anteriores, aunque estandarizado el formato, no existe garantía de que un EAR utilizado en WebSphere sea ejecutable en Bluestone.