TEMA 067. ENTORNO DE DESARROLLO JAVA

Actualizado a 15/05/2023

1. INTRODUCCIÓN

Java es un lenguaje de programación y una **plataforma informática** que fue creada por Sun Microsystems. Proporciona un lenguaje de programación y una serie de herramientas y estándares para crear programas informáticos de una forma unificada e interoperable.

Tras su creación ha pasado por muchas modificaciones y versiones hasta que ha llegado a ser lo que es ahora. El lenguaje de programación y las diferentes interfaces que estandariza dan lugar múltiples implementaciones que consiguen que el código generado sea **portable** entre diferentes máquinas y sistemas operativos. Además, estas especificaciones son de **código libre**. Estos son factores fundamentales que le han dado la relevancia que tiene en las infraestructuras de las organizaciones.

2. PLATAFORMAS DE JAVA

Como se ha introducido Java no es sólo un lenguaje de programación sino una plataforma informática. Para definir todas estas funciones esta plataforma se divide en partes, las cuales se utilizan y a veces extienden o reducen.

La primera, y fundamental, es **Java SE (Java Platform Standar Edition)** es el propio lenguaje de programación y las interfaces estándar del lenguaje de programación Java, forma la funcionalidad básica de las plataformas Java para el desarrollo de aplicaciones.

La segunda, y muy relevante para el ámbito empresarial, es **JackartaEE** (anteriormente **Java EE (Java Platform Enterprise Edition)**). Añade nuevas interfaces basándose en Java SE para el despliegue y construcción de aplicaciones empresariales a gran escala.

La tercera es **Java ME (Java Platform Micro Edition):** un subconjunto de Java SE para el desarrollo de programas que van a ejecutarse en dispositivos de pequeño tamaño como teléfonos móviles o electrodomésticos. Usando JavaCard se puede usar incluso en tarjetas inteligentes.

2. JAVA SE

Como se ha introducido Java SE es el lenguaje de programación, compilación y ejecución y las interfaces de funcionamiento.

2.1. EL LENGUAJE JAVA

El lenguaje es influenciado por C y se presenta, en la época como una alternativa mas conveniente.

El lenguaje Java se define por las siguientes características:

- Es **orientado a objetos**, aunque no se considera puramente orientado a objetos debido a:
  - No todos los tipos son objetos. Se dispone de tipos primitivos como: Números enteros,Números reales ect.
- Lenguaje **fuertemente tipado**.
- Independiente de la plataforma, por tanto es **portable**.
- Compilado a bytecode. Despues el bytecode es interpretado según la máquina
- Usa **recolección de basura** como método de gestión de memoria

2.2. EJECUCIÓN DE JAVA

Existen dos tipos de paquetes de que se pueden descargar para trabajar con Java:

- **JRE** (Java Runtime Enviroment)
  - Dispone del software necesario para ejecutar programas Java

- **JDK**
    - Además de lo contenido en el JRE añade programas de desarrollo para crear los programas Java

Existen varios vendedores que proporcionarán el JDK y JRE por ejemplo:

- Oracle JDK
- IBM Semeru Runtime Certified Edition
- OpenJDK JDK

Entrando en la creación y ejecución de Java de una forma más técnica. La pieza fundamental en la ejecución es la **Máquina virtual Java**. Esta se encarga de traducir el bytecode generado en la compilación a código máquina que puede ejecutar el sistema anfitrión. Por tanto en los sistemas en los cuales se cuente con una implementación de esta máquina podrá ser ejecutado el programa.

Existen varias implementaciones de la máquina virtual Java entre las que destacan:

- Java HotSpot VM (OpenJDK)
- GraalVM(Oracle)
- Eclipse OpenJ9 (Eclipse foundation)

2.3. VERSIONES DE JAVA SE

Referido a las versiones de Java SE existen dos tipos versiones diferentes. Las primeras son las versiones convencionales, las segundas son las versiones **LTS (Soporte extendido)** las cuales tienen un período de soporte más longevo que las versiones normales. El período de soporte dependerá en parte del fabricante de la plataforma en la que ejecutes el código java, por tanto no es trivial ponerles fechas de fin de soporte. Las versiones se van sucediendo incluyendo funcionalidades al lenguaje.

A continuación, una tabla con las versiones más modernas:

Versión	Fecha de publicación	Novedades más relevantes
Java SE 11 (LTS)	25 de Septiembre de 2018	Acceso basado en anidación, cliente HTTP
Java SE 12	19 de Marzo de 2019	Microbenchmark
Java SE 13	17 de septiembre de 2019	
Java SE 14	17 de marzo de 2020	Mejores excepciones de puntero nulo, mejoras en switch
Java SE 15	16 de septiembre de 2020	Clases ocultas, ZGC
Java SE 16	16 de marzo de 2021	Implementación Windows/AArch64, jpackager
Java SE 17 (LTS)	14 de septiembre 2021	Implementación macOS/AArch64, Clases selladas

Java SE 18	22 de marzo de 2022	UTF-8 por defecto, Snippets para Javadoc
Java SE 19	20 de septiembre de 2022	Implementación RISC-V
Java SE 20	21 de marzo de 2023	

2. JACKARTAEE/ JAVAEE

Esta versión de java pretende dar respuesta a los retos que supone ejecutar JavaSE en un entorno corporativo de una forma portable entre sistemas. Definiendo **interfaces**, que los fabricantes han de implementar, se consigue sistemas que pueden ser portados entre fabricantes y permiten tener una gran cantidad de funcionalidad que podría ser necesaria en los entornos. Los fabricantes crean software basado en una Máquina virtual Java añadiendo las **implementaciones** para todas las interfaces especificadas en JackartaEE.

Cuando se lea documentación es fácil encontrarse con el termino **JavaEE**. Anteriormente **JackartaEE** era llamado JavaEE. esto es debido a que la especificación perteneció a Oracle durante mucho tiempo, para continuar con la tecnología de forma libre se donó la especificación a la Eclipse Foundation la cual continua el desarrollo de la especificación sin el control exclusivo de una empresa. La versión JavaEE 8 es la última y es plenamente coincidente con JackartaEE 8.

A continuación, una relación de las versiones más recientes de Jackarta EE/ JavaEE:

Versión	Fecha de publicación	Versión de Java SE soportada
Jakarta EE 10	13 de septiembre del 2022	Java SE 17 Java SE 11
Jakarta EE 9.1	25 de mayo del 2021	Java SE 11 Java SE 8
Jakarta EE 9	8 de diciembre del 2020	Java SE 8
Jakarta EE 8	10 de septiembre del 2019	Java SE 8
Java EE 8	31 de agosto del 2017	Java SE 8

Como se ha comentado en la introducción la especificación está compuesta por interfaces, cada una con el propósito de facilitar una funcionalidad comúnmente necesitada en entornos corporativos. A continuación, las interfaces más comúnmente usadas en Jakarta EE, con el nombre que tenían en JavaEE.

API (Jackarta EE)	API (Java EE)	Funcionalidad
Jakarta Servlet	Java Servlet	Interfaz para el manejo del protocolo HTTP

Jakarta Server Pages	Java Server Pages	Motor de plantillas para aplicaciones web
Jakarta Faces	JavaServer Faces	Framework MVC para aplicaciones web
Jakarta Persistence	Java persistence	Interfaz para gestionar la persistencia en bases de datos relacionales
Jakarta XML Binding	Java Architecture for XML binding	Permite la transformación automática entre Xml y objetos Java

Por completar el apartado, en el siguiente gráfico se ven todas las funcionalidades de JackartaEE agrupadas por perfiles:

Finalmente, una lista con productos usados comúnmente como servidores de aplicaciones:

Producto	Última versión	Estandar soportado
Oracle WebLogic	14c (30 de mayo de 2020)	Java EE 8 Jakarta EE 8
IBM WebSphere Application Server	9	Java EE 8
Eclipse GlassFish	7	Jakarta EE 10

Red Hat JBoss Enterprise Application Platform	7	Java EE 8 Jakarta EE 8
Apache tomcat	10	Jakarta EE 10

3. ECOSISTEMA JAVA

Comúnmente se usan ciertas herramientas en los ecosistemas Java. En este apartado se las va a nombrar para que queden referenciadas.

Herramientas de compilación:

- Apache Maven
- Graddle

Herramientas de test e integración continua:

- Junit
- Jmetter
- Jprofiler

Entornos de desarrollo integrados

- Eclipse
- Intelij Idea

Integración contínua:

- SonarQube
- Jenkins

Frameworks:

- Spring MVC
- Spring boot
- Apache Structs

Librerías:

- Hibernate

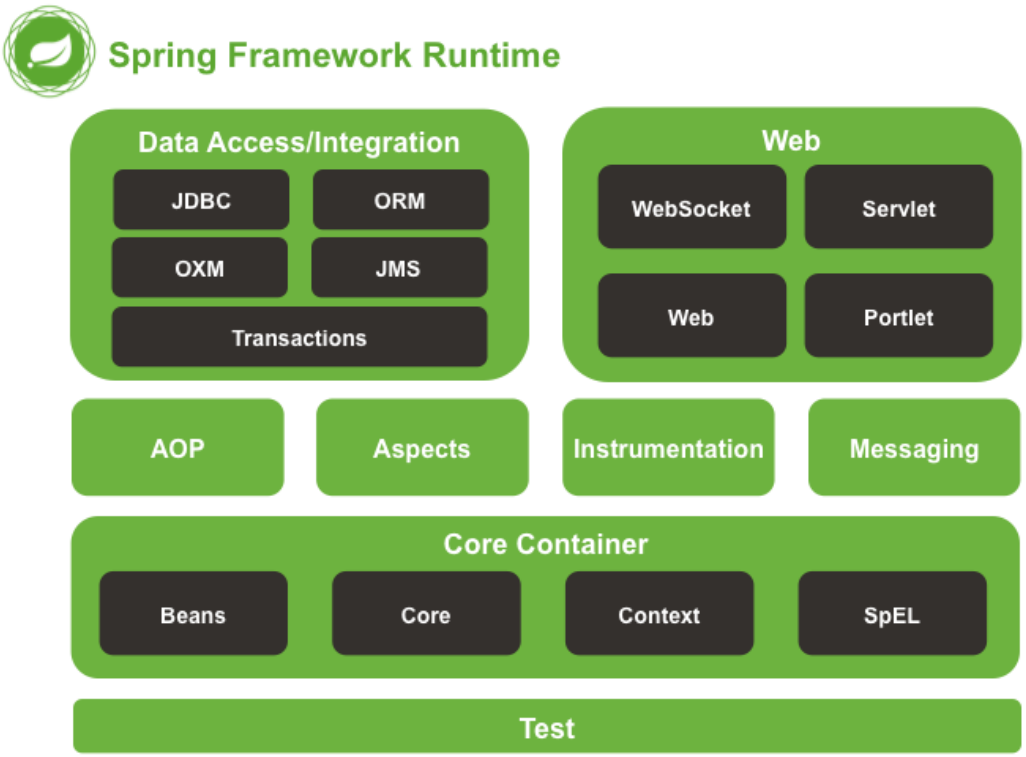
Lenguajes secundarios:

- Kotlin
- Clojure
- Scala.

3.1. ESPECIAL MENCIÓN A SPRING

Spring es uno de los frameworks más usados de Java. Su popularidad radica en que implementó funcionalidades y lo sigue haciendo mucho más rápido que las especificaciones y por tanto es el más usado de todo el ecosistema Java. Añade funcionalidades útiles para el programador y se construye

sobre las bases sentadas de las especificaciones de Java. A continución el diagrama básico de los componentes de este Framwork.



Recientemente ha ganado popularidad una configuración de este framework llamada Spring Boot, esta pretende simplificar la configuración y habilitar el framework para un caso de uso de Microservicios, simplificando el proceso de craci ón y el código necesario para la creación de una aplicación spring.

3.1. Especial mención a los Lenguajes secundarios

A raiz del éxito de Java han ido naciendo lenguajes que se ejecutan en la máquina virtual de Java pero no son Java. Nacen con un caso de uso en mente normalmente y tienen usos específicos. Se hará un repaso de los más importantes.

Kotlin: Es el lenguaje de programación oficial de Android y se presenta como un lenguaje más conciso que Java y adaptado al gusto de los programadores. Presenta total interoperatividad con Java pudiendo mezclar clases directamente en ambos lenjguajes.

Scala: Scala es un lenguaje para Big Data pensado para ejecutarse en la arquitectura de Spark, está construido con el paralelismo masivo en mente.

Clojure: Implementa un paradigma de programación funcional, es una forma diferente de funcionar de los lenguajes de programación que le es de utilidad a ciertos programadores.

4. Glosario

Bytecode:

Bytecode se la llama en java al compilado que se produce con el código Java. Este compilado esta formado de instrucciones muy simples, las cuales son fácilmente traducibles a intrucciones nativas. Por tanto este código es muy muy rápido, casi nativo en la ejecución y mantiene la propiedad de que se puede usar en muchas máquinas diferentes.

Recolección de basura:

Es un método de gestión de memoria en programación. Se basa en que se sigue la ejecución del programa registando cada vez que se adquiere memoria ram. Cuando lógicamente es imposible que esa misma memoria se vuelva a usar, será detectado y liberada. Esto libera al programador de la tarea de gestión de memoria aumentando la productividad a costa de un ejecutable más lento.

Orientación a objetos:

Es un paradigma de programación basado en el concepto de objeto. Esto es que la información y las instrucciones para procesarla han de ir de la mano dando lugar a fenómenos, que incrementan la productividad de los programadores y la mantenibilidad del código, como la modularidad, el encapsulamiento, abstracción, reusabilidad.

5. REFERENCIAS

Especificaciones Jackarta EE	https://jakarta.ee/specifications/
Documentación Java SE 17	https://docs.oracle.com/en/java/javase/17/
Documentación de Spring framework	https://docs.spring.io/spring-framework/docs/current/reference/html/