Tabla de Contenidos

¿Qué es Jakarta EE? Guía actualizada 2026

Jakarta EE es, hoy, el estándar abierto para el desarrollo de aplicaciones empresariales en Java. Nació en 2019 como la evolución de Java EE tras su donación por parte de Oracle a la Eclipse Foundation, y en estos años ha dejado de ser "una promesa de futuro open source" para convertirse en la base real de servidores de aplicaciones como WildFly, Open Liberty, Payara o GlassFish.

Este artículo era originalmente de 2019, cuando Jakarta EE acababa de nacer y todo eran incertidumbres: cómo se renombrarían los paquetes, si Oracle cedería la marca, qué pasaría con `javax.*`. Todo eso ya está resuelto. Vamos a ver dónde estamos realmente en 2026.

De Java EE a Jakarta EE: qué cambió de verdad

El cambio más disruptivo fue el que en 2019 generaba más dudas: el paso de `javax.*` a `jakarta.*`. Oracle mantuvo la propiedad de la marca "Java" y del namespace `javax`, así que a partir de Jakarta EE 9 (2020) todas las especificaciones renombraron sus paquetes.

En la práctica, esto significa que cualquier proyecto Java EE 8 que quiera migrar a Jakarta EE moderno necesita:

- Cambiar todos los imports de `javax.persistence`, `javax.servlet`, `javax.ejb`, etc. a `jakarta.persistence`, `jakarta.servlet`, `jakarta.ejb`.
- Actualizar el servidor de aplicaciones a una versión compatible con Jakarta EE 9+.

- Revisar librerías de terceros que todavía dependan del namespace antiguo. Herramientas como OpenRewrite automatizan buena parte de esta migración, así que hoy es un problema resuelto, no un obstáculo.

Dónde estamos ahora: Jakarta EE 11

Jakarta EE 11 se publicó en 2025 y es la versión de referencia actual. Sus novedades más relevantes:

Jakarta Data 1.0, una especificación nueva que introduce el patrón *repository* al estilo Spring Data, pero como estándar Jakarta EE, reduciendo mucho el código repetitivo de acceso a datos.

Soporte de Virtual Threads (Java 21) en Jakarta Concurrency, lo que permite manejar miles de tareas concurrentes con un modelo de programación mucho más simple que el tradicional basado en pools de hilos.

Alineación con Java SE 17 y 21 LTS, dejando atrás definitivamente el mundo Java 8 en el que vivía la especificación original.

Actualizaciones profundas en CDI 4.1, Jakarta Persistence 3.2, Jakarta Security 4.0 y Jakarta RESTful Web Services 4.0.

Lo que viene: Jakarta EE 12

El trabajo en Jakarta EE 12 ya está en marcha, con lanzamiento previsto para este mismo año. Entre lo que se está discutiendo en la comunidad:

- Soporte de nivel de API sobre Java SE 21, con soporte en tiempo de ejecución para Java SE 25.
- Nuevas especificaciones candidatas como Jakarta Query y mejoras en Jakarta NoSQL.
- Una posible convergencia con MicroProfile, que aportaría una API de configuración común y

reuniría especificaciones que hoy viven de forma separada (por ejemplo, MicroProfile REST Client y Jakarta RESTful Web Services).
Jakarta EE mantiene un ritmo de release aproximadamente bianual, lo que da bastante previsibilidad para planificar migraciones en proyectos empresariales.

¿Por qué sigue importando Jakarta EE en la era de Spring y la IA?

Es una pregunta razonable en 2026, con Spring Boot dominando gran parte del ecosistema y la IA generativa entrando en cada capa del desarrollo. Pero Jakarta EE conserva un papel relevante por varios motivos:

Spring Boot se apoya en Jakarta EE. Muchas de las anotaciones y APIs que usas a diario en Spring (`jakarta.persistence.Entity`, `jakarta.validation.constraints`, `jakarta.servlet`) son, literalmente, especificaciones Jakarta EE. Entender el estándar de base te da criterio para entender por qué Spring toma las decisiones de diseño que toma.

Sigue siendo el estándar en sistemas legacy y sector público. Buena parte de la administración pública española y de grandes corporaciones mantiene sistemas Java EE clásicos sobre WildFly, WebLogic o WebSphere, y su modernización hacia Jakarta EE actual (o su migración a Spring) es un trabajo real y bien pagado para arquitectos.

Portabilidad frente a vendor lock-in. Al ser un estándar con múltiples implementaciones, Jakarta EE sigue siendo la opción de referencia cuando la portabilidad entre servidores de aplicaciones es un requisito de arquitectura.

Otros artículos relacionados

- ¿Es el fin de los servidores Java EE?
- Servlets y ciclo de vida
- Java EE MicroServicios