Unión Europea

Fondo Social Europeo

El FSE invierte en tu futuro

Fecha	Versión	Descripción
01/09/2022	1.0.0	Versión inicial.
01/09/2024	2.0.0	Revisión y actualiación

Unidad 1 - Introducción a las aplicaciones empresariales

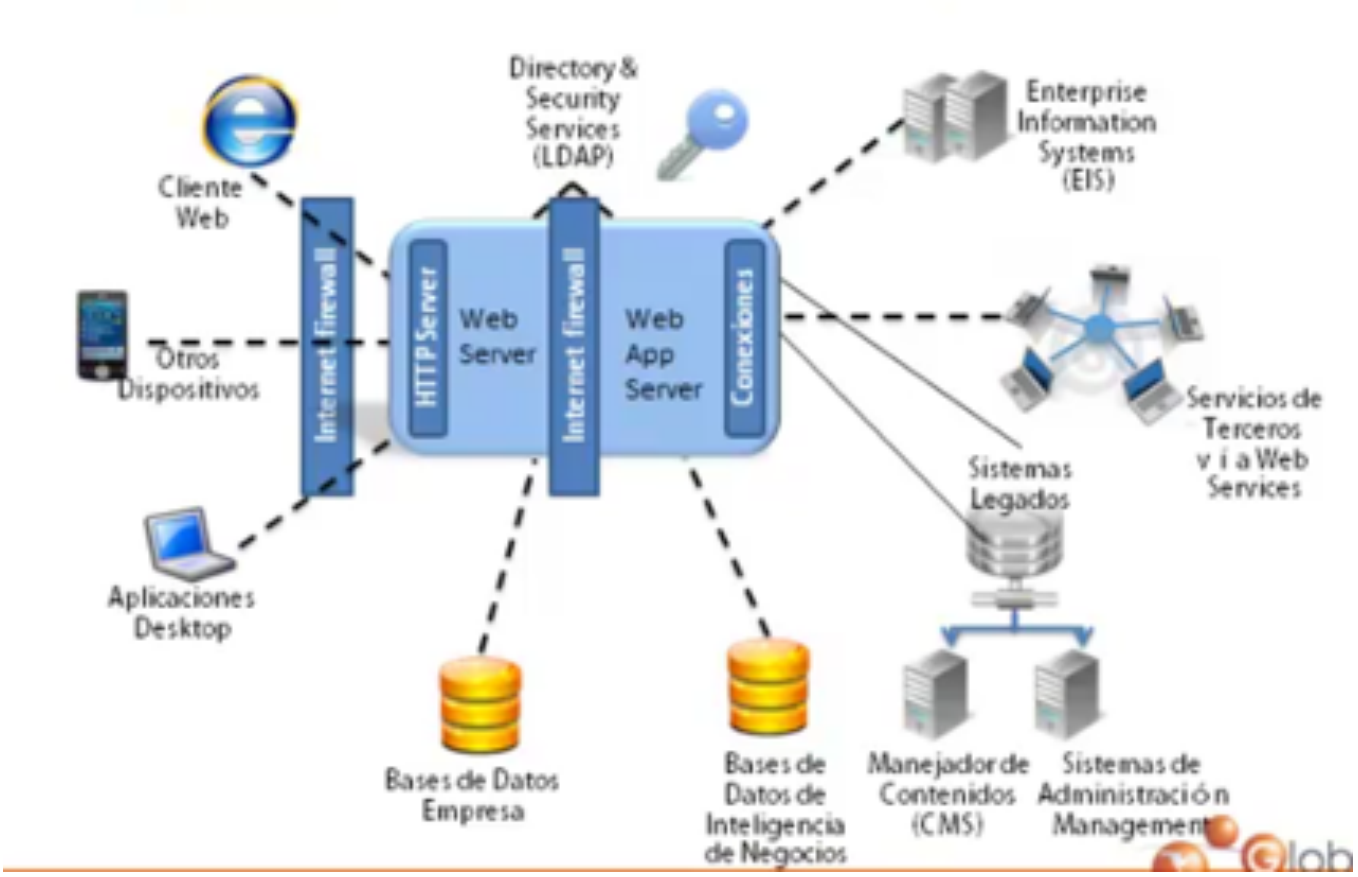
1. ¿Qué es una aplicación empresarial?

Una aplicación empresarial, básicamente, se basa en una aplicación funcionando sobre un servidor de aplicaciones, que proporcionará soporte a diferentes tipos de clientes, ya sea clientes web, móvil o de escritorio, todos queriendo acceder a la misma aplicación, ya sea por medio de Web Services o por medio de un Application Service de Java.

Además, estos servidores también cuentan con:

- Seguridad.
- Bases de datos propias de la empresa
- Bases de datos de tipo BI (Business Intelligence)
- CMS
- Sistemas de administración de la empresa
- Comunicación con otras aplicaciones mediante Web Services

- Cualquier otra información almacenada en un sistema de ficheros, el cual será conocido como Enterprise Information System (EIS)



Las aplicaciones empresariales Java tienen a su cargo establecer las reglas de negocio de la empresa y/o sistema y ofrecer conectividad a los distintos tipos de clientes, con ello se logra ofrecer una solución integral a sus necesidades de sistemas de información a la medida.

La versión empresarial de Java (Java Enterprise Edition) cuenta con una enorme comunidad de programadores alrededor del mundo. A su vez, una de las mayores ventajas de Java es su independencia de plataforma, de esta manera, el programador puede seleccionar entre utilizar herramientas de paga o de software libre, y esto comienza desde el sistema operativo, hasta las herramientas de desarrollo, el servidor de aplicaciones, la base de datos, etc.

La versión 8 de Java tiene como objetivo la simplificación en la programación para requerimientos empresariales, y con ello busca incrementar la productividad del desarrollador Java.

Desde el 24 de abril de 2018 Java EE es gestionado por la Fundación Eclipse. La Fundación Eclipse fue forzada a eliminar la palabra Java del nombre debido a que Oracle posee el registro de la marca "Java". El 26 de febrero de 2018 se anunció que el nuevo nombre de Java EE sería Jakarta EE, aunque nosotros seguiremos llamándole Java EE.

La última versión que existe es Jakarta EE 10, donde podemos ver la documentación en la siguiente URL: https://jakarta.ee/

2. ¿Que es Java EE (o Jakarta EE)?

Para entender qué es Java EE, comencemos por responder a la pregunta de ¿Qué es un API?. Un API (Application Programming Interface) es un conjunto de clases que resuelven una necesidad muy particular. Por ejemplo el API de JDBC permite crear código Java para establecer la comunicación con una base de datos.

Java EE es un conjunto de API's enfocadas en brindar una serie de servicios que toda aplicación empresarial necesita, tales como: transaccionalidad, seguridad, interoperabilidad, persistencia, objetos distribuidos, entre muchos servicios más. Estas APIs se basan en un conjunto de especificaciones, las cuales pueden ser implementadas por empresas orientadas a software libre (Tomcat, Jboss, etc) o software comercial (Oracle, IBM, etc).

Una de las grandes ventajas de seleccionar estas tecnologías es que son el estándar propuesto por el grupo JCP (Java Community Process), el cual se encarga de revisar y liberar las especificaciones Java y las APIs empresariales respectivas.

En resumen, la versión empresarial de Java se puede entender como una extensión de la versión estándar (JSE), pero con la intención de facilitar el desarrollo de aplicaciones empresariales, permitiendo agregar de manera muy simple los servicios descritos anteriormente, y así crear aplicaciones Java robustas, poderosas, y de alta disponibilidad.

3. El Modelo de Aplicación Java EE.

La Tecnología Empresarial Java EE (o Jakarta EE) incluye muchas mejoras en cada una de las tecnologías que la componen, en particular se enfoca en simplificar la integración de varios componentes a través del concepto de CDI (Contexts and Dependency Injection), el uso de anotaciones y el uso de POJOs (Plain Old Java Objects).

El modelo de aplicación Java EE define una arquitectura para la implementación de servicios como aplicaciones de varios niveles que ofrecen la escalabilidad, accesibilidad y capacidad de gestión, necesaria para aplicaciones de nivel empresarial. Este modelo divide el trabajo, necesario para implementar un servicio de varios niveles, en las siguientes partes:

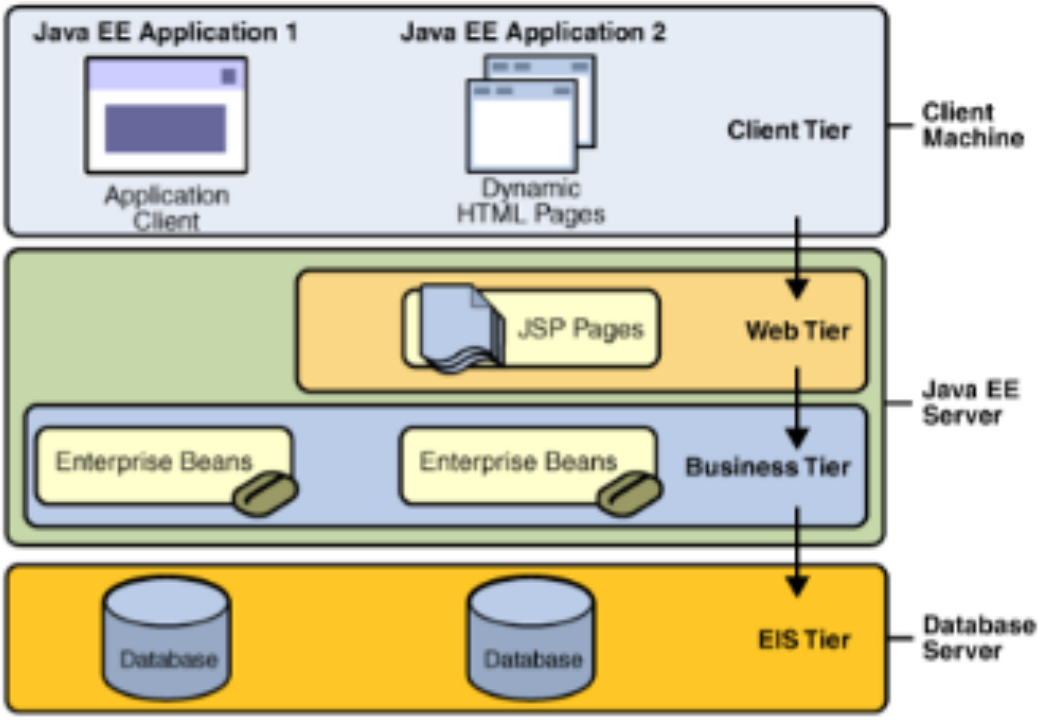
- La lógica de negocio y presentación para ser implementado por el desarrollador.
- Los servicios del sistema estándar proporcionados por la plataforma Java EE.

Tal vez llegados a este punto todavía no habéis comprendido que lo que se está planteando con Java Enterprise Application es una organización a la hora de desarrollar software, de forma que su diseño e implementación de este se debe contextualizar en toda la organización.

Retomando el tema, el modelo Java EE define una arquitectura para la implementación de servicios como aplicaciones de múltiples capas. Este modelo divide el trabajo de implementación en dos partes (ya lo hemos comentado, pero lo repetimos):

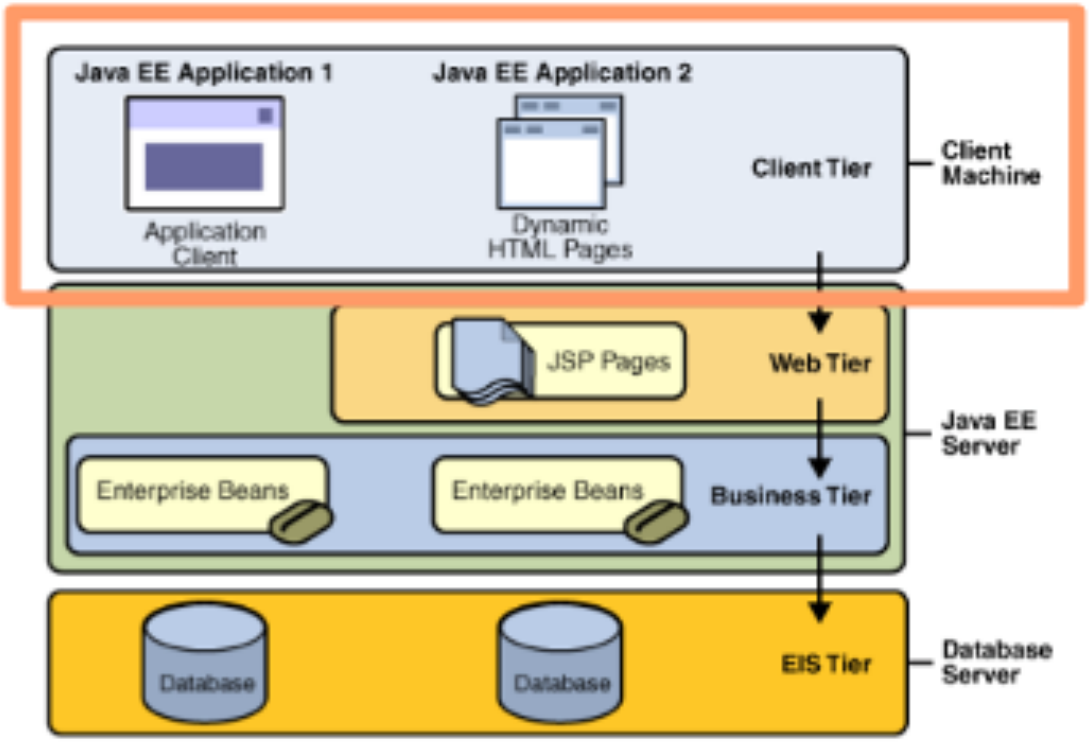
- La parte del desarrollador:
    - La lógica de la organización ó lógica de negocio (business logic)
- La lógica de la presentación (presentation logic)
- Los servicios proporcionados por la plataforma Java EE.

En la siguiente imagen podemos visualizar el modelo Java EE con los distintas capas:

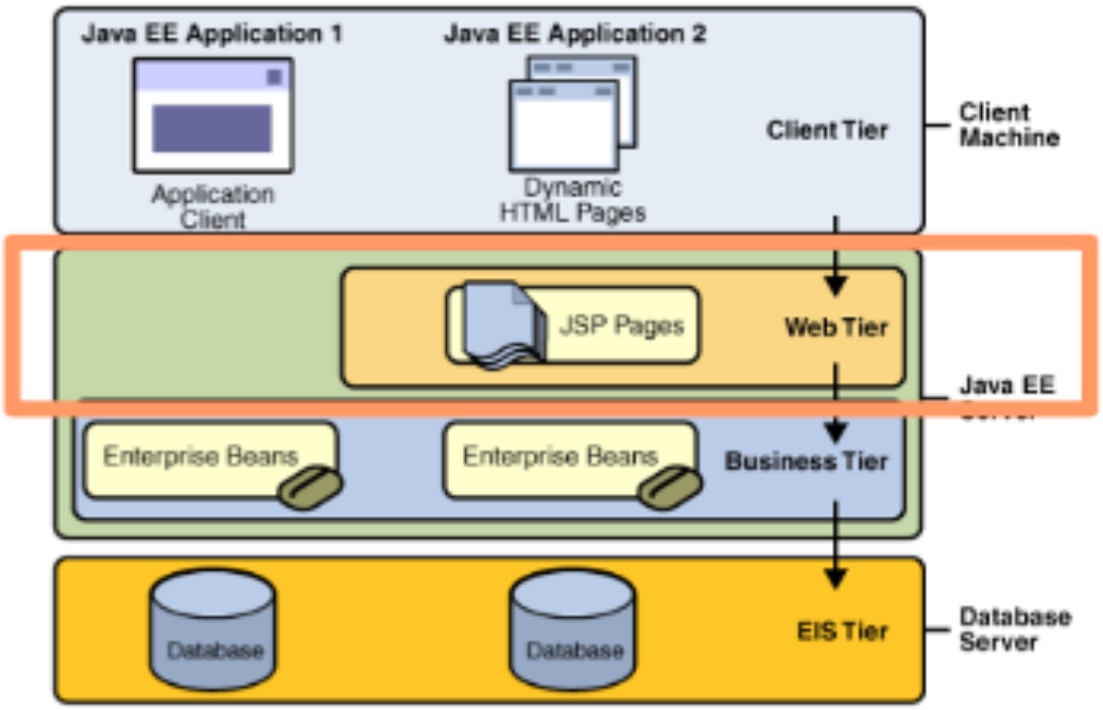


En esta arquitectura el diseñador decide que capas implementa dependiendo del tipo de aplicación o de la estructura de la organización se requiere.

Vamos a hablar un poco sobre cada una de ellas, ya que sin duda es a lo que nos vamos a dedicar todo el curso:

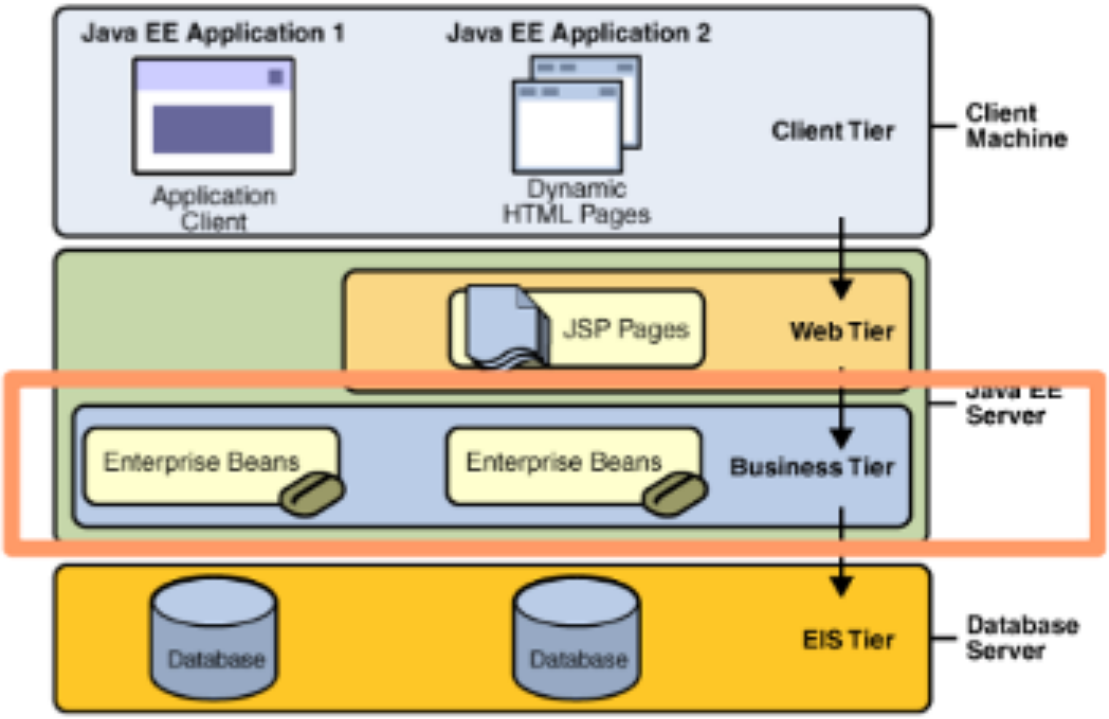


En la parte de las máquinas cliente (o capa cliente) podemos encontrar aplicaciones cliente que son ejecutadas en un navegador, teléfono inteligente, aplicación de escritorio, etc.



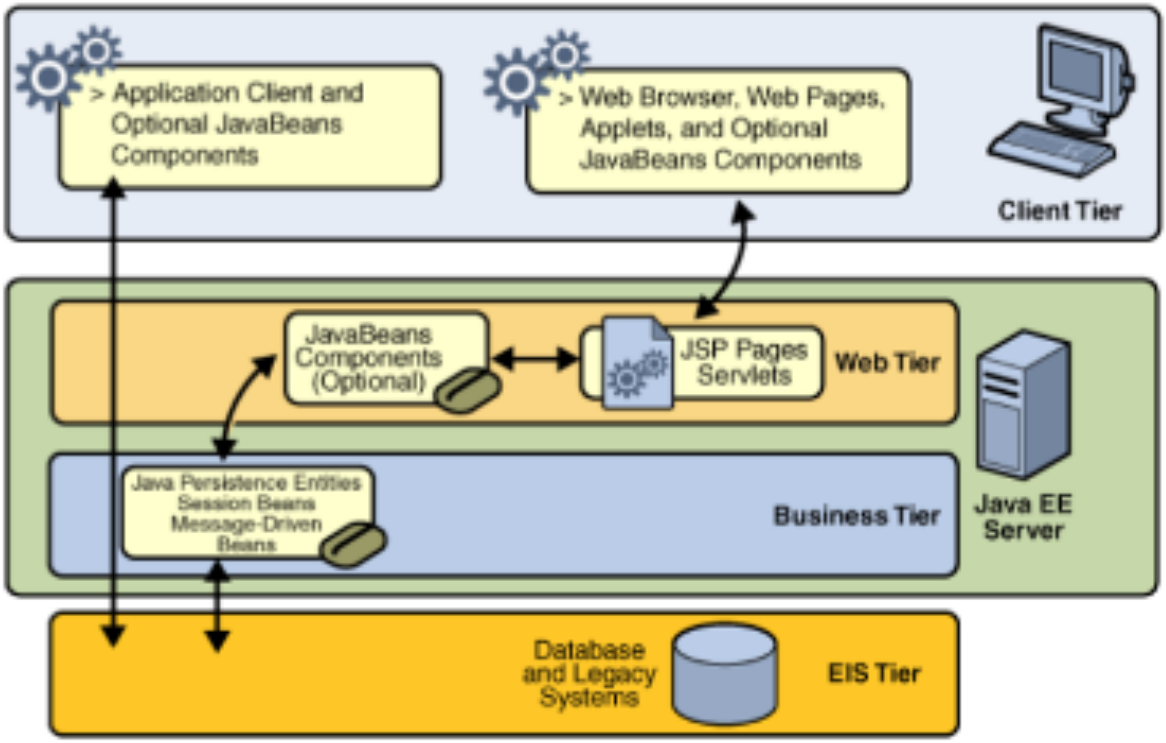
En el servidor tenemos las dos partes bien diferenciadas que hemos hablado con anterioridad: capa web (Web Tier) y capa de lógica de negocio (Business Tier).

La capa web corre en el servidor de aplicaciones y se comunica con las máquinas cliente (o capa cliente) y con la capa de lógica de negocio, de forma que obtiene información de esta y la transforma al formato adecuado (por ejemplo HTML, JSON, XML,...). En ell encontrarnos con las siguientes tecnologías: Java Servlets, Java Server Pages y JavaServer Faces, que implementan componentes web que se ejecutan del lado del servidor.



Mientras que en la lógica de negocio podemos encontrar la tecnología Enterprise JavaBeans (EJB), que es la que permite implementar componentes Enterprise Beans que se ejecutan al lado del servidor.

¿Toda esta organización que implica? Implica que las aplicaciones cliente han de comunicarse con la lógica de negocio para poder funcionar correctamente, es decir, todos han de pasar por esa capa, y esto no ocurre cuando en nuestras aplicaciones trabajamos con las antiguas entidades que representaban objetos (llamados también POJOs).



Esto hace que las aplicaciones empresariales sean de difícil desarrollo, ya que involucran muchas líneas de código para el manejo de transacciones, multitarea, administración de recursos (resource pooling), seguridad, entre otros.

Por ello la división y organización en la arquitectura nos permite una implementación más fácil, ya que la lógica de la organización es dividida en componentes utilizables, y un nuevo concepto software que no hemos nombrado hasta ahora, que son los contenedores, serán los responsables del manejo de transacciones, multitarea, administración de recursos (resource pooling), seguridad, etc.

Por lo que, antes de ser ejecutados los componentes web, Enterprise Bean o aplicaciones cliente, estos son comprobados y ensamblados en módulos de Java EE y distribuidos en sus respectivos contenedores.

Por lo que podemos decir que Java EE se basa en tres conceptos claves: Servicios, Contenedores y Componentes.

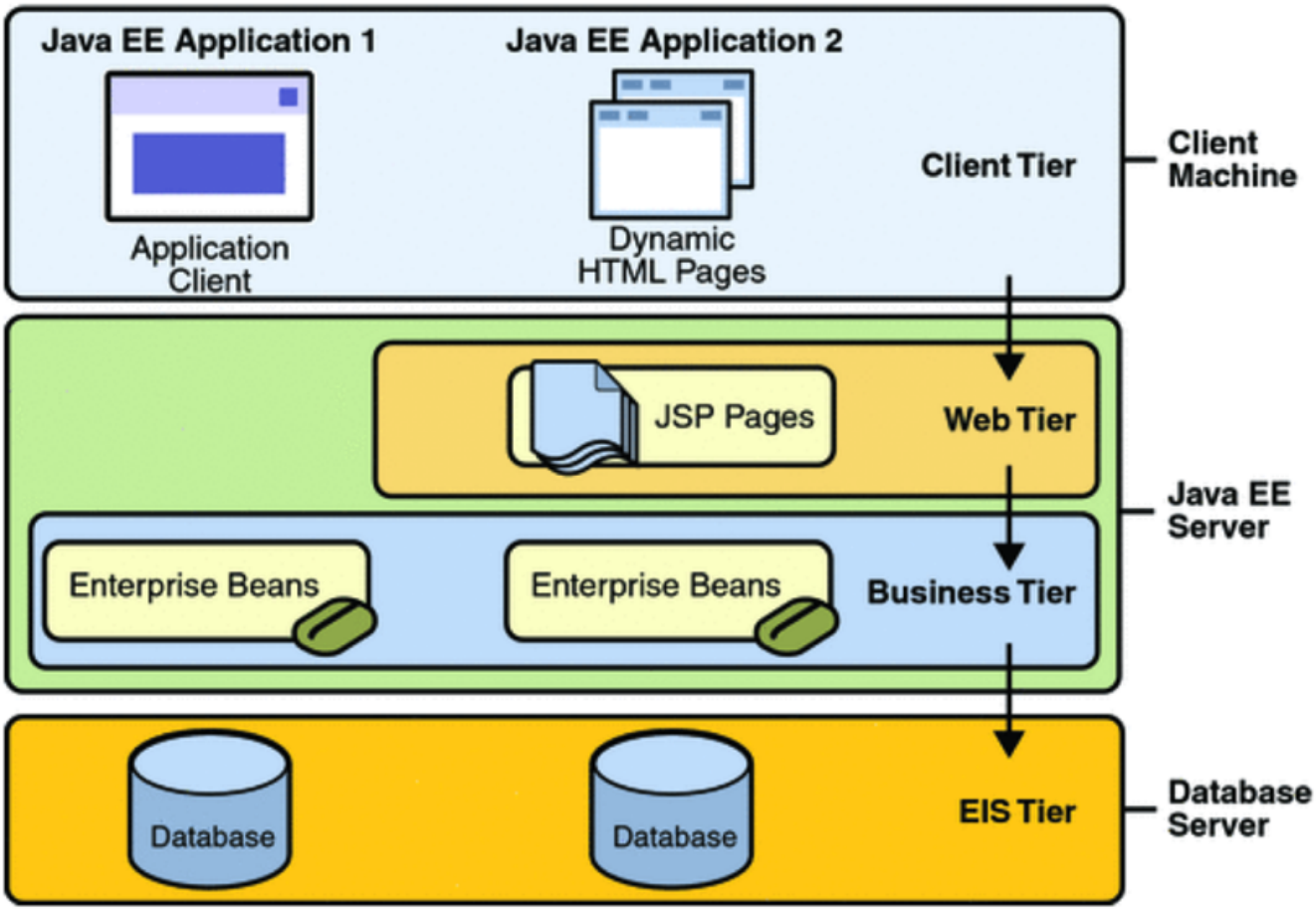
- Los Componentes: Unidades de software que forman o componen la aplicación. Son esos objetos POJO que pueden ser reusados. Estos contienen la lógica de negocio de la aplicación y usan los servicios proporcionados por el contenedor. Hay varios tipos de componentes y según ese tipo son instalados(desplegados) en un contenedor u otro.
- Los Contenedores: Entorno de ejecución donde se ejecutan los componentes. Hay diversos tipos según la funcionalidad que tengan.
- Los Servicios: Son funcionalidades estándar que todo contenedor debe proveer a los componentes Estos servicios son proporcionados por un contenedor. Así el programador se concentra en su lógica de negocio y usa estos servicios para su aplicación.

Los 3 conceptos anteriores permiten a Java EE definir una arquitectura de capas y distribuida. Aunque las hemos definido antes, las capas en el estándar Java EE son las siguientes:

- Capa cliente (Client Tier): responsable de la interacción con el usuario. Corren en la máquina del cliente.

- Capa web (Web Tier): responsable del control de la aplicación y en ocasiones también de la interacción con el usuario. Son componentes que corren en un contenedor web.
- Capa de negocio (Business Tier): responsable de la lógica de la aplicación propiamente dicha. Componentes que corren en un contenedor de negocio.
- Capa de datos (EIS Tier): responsable de la persistencia de datos y/o lógica especializada (conocida con el nombre de EIS: Enterprise Information System, o Sistema de Información Empresarial). Por ejemplo ERPs, BBDD, Motores Transaccionales (CICS, IMS, Tuxedo...).

Las capas 2 y 3 son las principales hacia donde se enfoca Java EE, es decir la capa web (o capa de presentación) y la capa de negocio. Ese es el dominio de acción de los servidores de aplicaciones. Recordemos que un servidor de aplicaciones esta formado por varios contenedores de componentes.



4. Contenedores Java EE.

Normalmente el desarrollo de una aplicación empresarial es muy complicado dado que el desarrollador debe tener en cuenta temas muy importantes como la gestión multiusuario, la gestión de la transaccionalidad, la gestión de la seguridad, la compartición de recursos, etc...

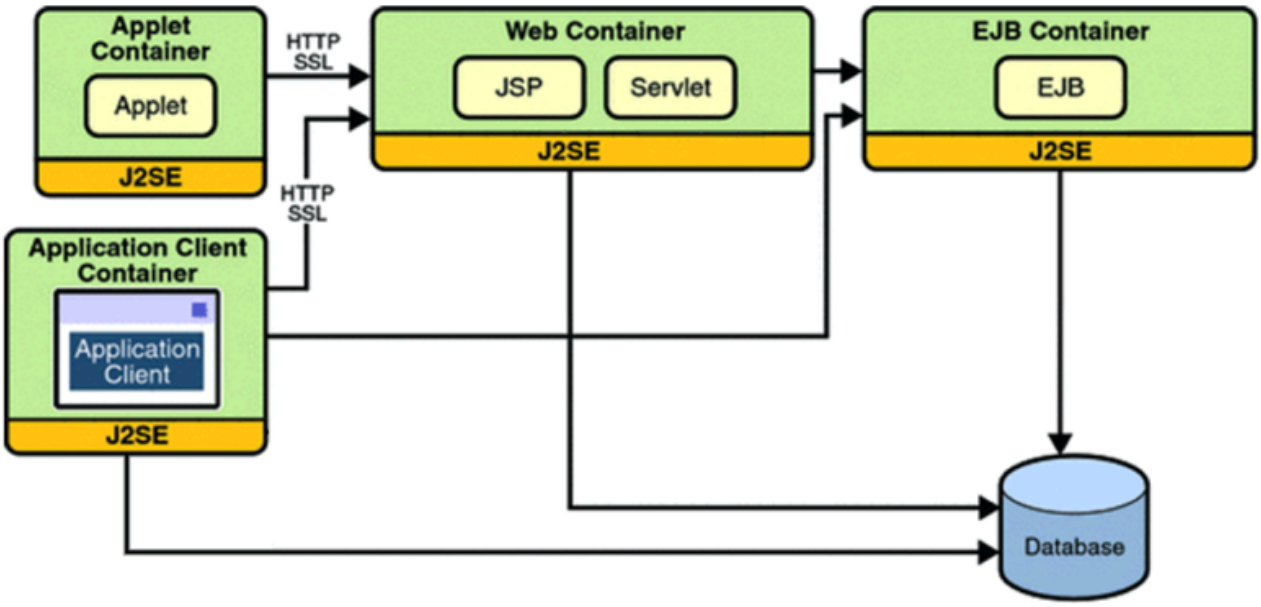
El modelo de programación de la plataforma Java EE facilita enormemente esta tarea con la definición de los contenedores Java EE. Estos contenedores ofrecen al desarrollador una serie de servicios sobre los que se puede apoyar permitiéndole centrarse en el desarrollo de la lógica de negocio de la aplicación propiamente dicha.

Dependiendo del tipo de contenedor, ofrecerá unos servicios u otros, y permitirá desplegar en él un tipo de componente u otro. Los tipos de contenedores Java EE son:

- Contenedor cliente (Application Client Container o Applet Container).
- Contenedor web (Web Container).
- Contenedor de negocio o de EJBs (EJB Container).

Como podemos ver, cada tipo de contenedor corresponde con una de las capas definidas, a excepción de la capa de datos que está implementada por otro tipo de productos (ya mencionados anteriormente) ajenos a la plataforma Java EE.

En el siguiente diagrama, podemos observar la relación entre los distintos tipos de contenedores Java EE:



5. Servicios Jakarta EE.

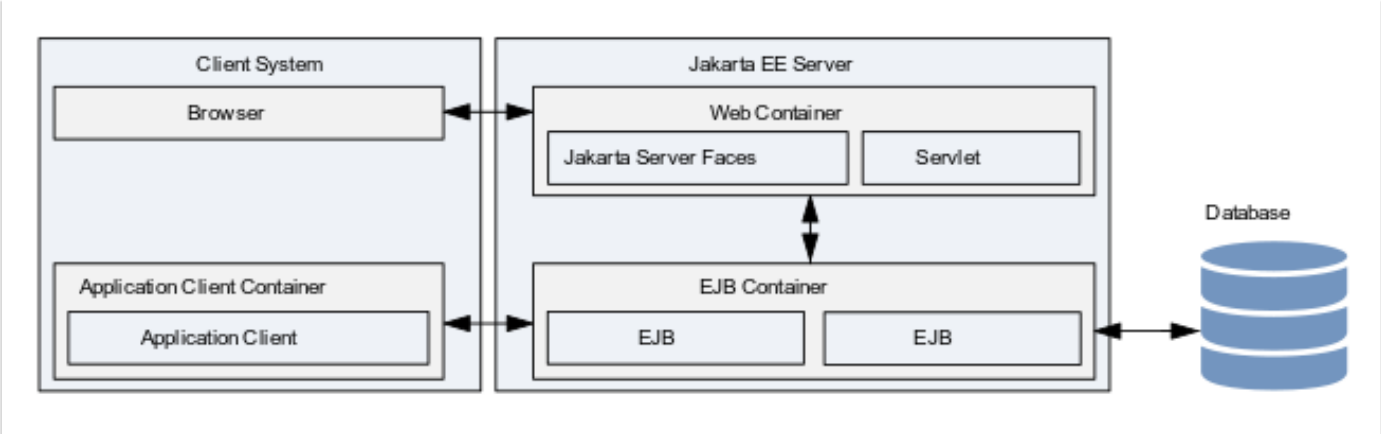
Las especificaciones Java EE, definen una serie de funcionalidades que los distintos tipos de contenedores deberán implementar y ofrecer a los desarrolladores de aplicaciones Java EE.

Existen multitud de servicios, pero simplemente destacaremos algunos:

- De directorio: para la indexación y búsqueda de componentes y recursos.
- De despliegue: para facilitar la descripción y personalización de componentes a la hora de su instalación.
- De transaccionalidad: para poder ejecutar distintas acciones en una misma unidad transaccional.
- De seguridad: para poder autenticar y autorizar a los usuarios de una aplicación.
- De acceso a datos: para facilitar el acceso a las Bases de Datos.
- De conectividad: para facilitar el acceso a los distintos Sistemas de Información Empresarial (EIS).
- De mensajería: para poder comunicarse con otros componentes, aplicaciones o EISs.

6. Jakarta EE APIs

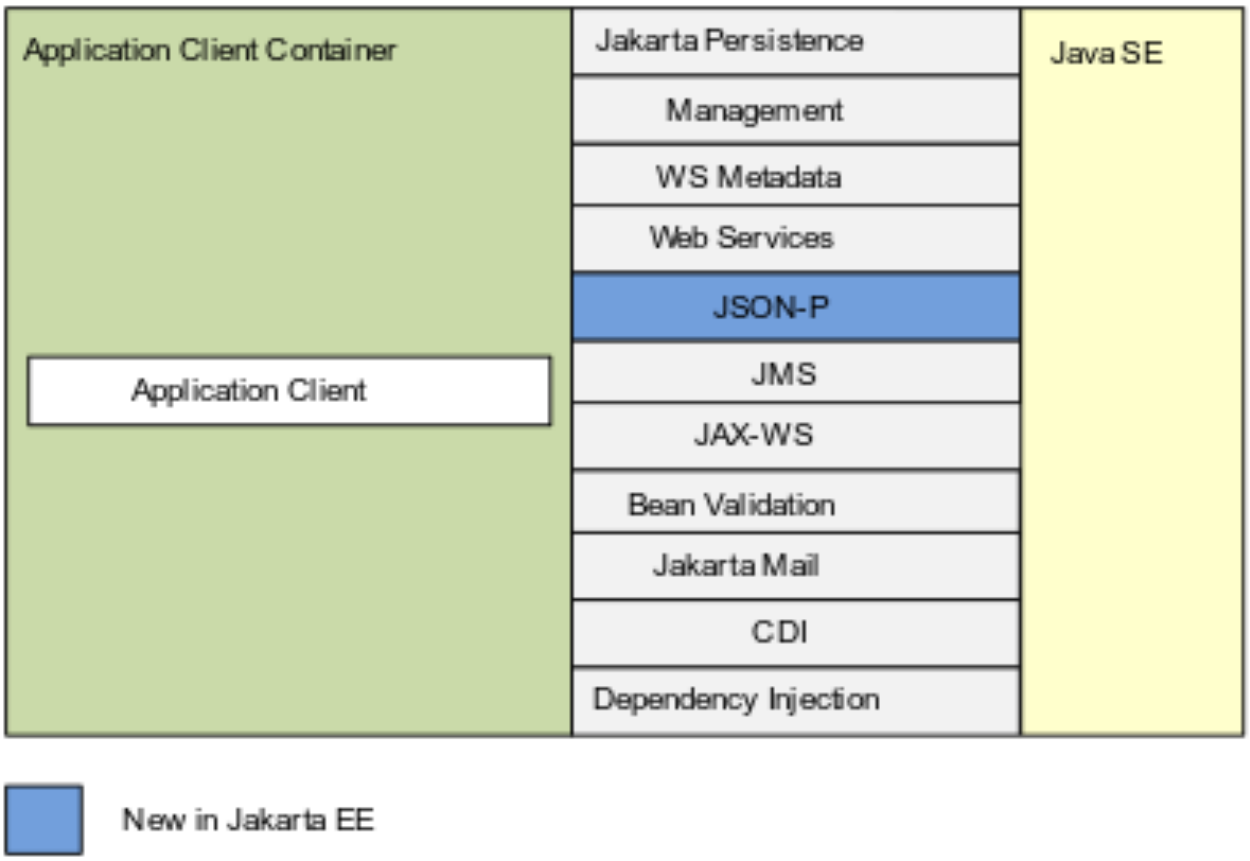
Una vez que tenemos los contenedores Jakarta EE tal como podemos ver en la siguiente figura:



Las APIs disponibles que tenemos en el contenedor web son las siguientes:

Mientras que las APIs disponibles en el contenedor EJB son las siguientes:

Y las APIs disponibles en el contenedor de cliente de aplicación son las siguientes:



En una aplicación Empresarial podemos utilizar tecnologías como EJB, JPA, Web Services, entre muchas más, a esto se le conoce como el Stack de tecnologías de la versión empresarial y podemos verlo simplemente como una extensión de la versión estándar de Java.

En una aplicación JEE podemos utilizar tecnologías como EJB, JPA, Web Services, entre muchas más. Jakarta EE ofrece características tales como:

- Empaquetar EJB locales en un archivo .war
- Singleton Session Beans
- EJB lite, que son EJB simplificados para utilizar en aplicaciones Web
- Integración entre JSF y EJB a través de CDI

Otras mejoras en la versión Java EE son:

- Contenedor Ligero: Para hacer más ligero el contenedor se introdujo el concepto de perfiles, donde podemos seleccionar el conjunto mínimo de tecnologías a utilizar. Por ejemplo, si una aplicación Web necesita de EJB, pero no de Servicios Web se puede utilizar EJB Lite, el cual se enfoca en utilizar únicamente las características básicas de los EJB.
- Remover APIs antiguas (Pruning): Se planteó que para versiones futuras de Java EE, se eliminen algunas API's, ya que son tecnologías mucho más complejas que las nuevas. Esto ya se hizo, por ejemplo, cuando EJB 2.x Entity Beans fueron sustituidos por JPA. El API de JAX-RCP se sustituyó por JAX-WS, y así varias tecnologías han sido reemplazadas. Por ello puede ser que en futuras versiones ya no sean soportadas.
- Facilidad de Uso: El uso de anotaciones simplificó en gran medida el uso de configuración vía archivos xml, por lo que archivos como faces-config.xml, ejb-jar.xml y persistance.xml se redujeron a unas cuantas líneas, e incluso en algunos casos es opcional el utilizarlos Además las clases ahora están orientadas a clases puras de Java (POJOs) e Interfaces, y en algunos casos, como los EJB's, el uso de interfaces es opcional. Sin embargo, al igual que en Spring Framework, el uso de interfaces es una buena práctica que se sigue aplicando al día de hoy.
- Etc...

7. Empaquetado

Para poder desplegar una aplicación Java EE, después de desarrollar los diferentes componentes, se empaqueta en archivos especiales que contienen los ficheros de las clases y los descriptores de despliegue XML. Estos descriptores de despliegue contienen información específica de capa componente empaquetado y son un mecanismo para configurar el comportamiento de la aplicación. Estos archivos especiales varían según el tipo de componentes:

Los componentes Web se empaquetan en un archivo Web (.war) que contiene los Servlets, las páginas JSP y los elementos estáticos como las páginas HTML y las imágenes. El fichero .war contiene clases y ficheros utilizados en el contenedor Web junto con un descriptor de despliegue (XML) de componentes Web.

Los Applets y sus descriptores de despliegue se empaquetan en un fichero Java (.jar) que se situará en la máquina cliente.

Los componentes de negocio se empaquetan en un archivo Java (.jar) que contiene los descriptores de despliegue de componentes EJB, además de los ficheros del interfaz remoto y del objeto junto con ficheros de ayuda requeridos por el componente EJB.

Una aplicación JEE se empaqueta en un archivo enterprise (.ear) que contiene toda la aplicación junto con el descriptor de despliegue que proporciona información sobre la aplicación y sus componentes.

En la siguiente figura vemos resumido todo lo explicado anteriormente.

8. Maven

Maven es una herramienta de gestión de información de proyectos. Maven está basado en el concepto de un modelo de objetos del proyecto POM (Project Object Model) en el que todos los productos (artifacts) generados por Maven son el resultado de consultar un modelo de proyecto bien definido. Compilaciones, documentación, métricas sobre el código fuente y un innumerable número de informes son todos controlados por el POM.

Además de esto, Maven consta de repositorios donde se encuentran las dependencias (bibliotecas .jar) necesarias para ejecutar nuestros proyectos. La ventaja que ofrece es que no es necesario descargar manualmente estas dependencias. Simplemente Maven se encarga de buscarlas y en los repositorios y nos permite agregarlas facilmente. Si una biblioteca depende de otra, Maven se encargará agregar estas dependencias a nuestro proyecto.