Antes de leer este artículo te recomiendo que le des una mirada a otros post como:

- ***Contenedores e Inyección de Dependencias 01***
- ***Contenedores e Inyección de Dependencias 02***

Si te animas un poco puedes incluso leer un poco sobre Spring framework aquí:

- ***Contenedor Spring 01***

Porque Java EE?

Cuando desarrollamos aplicaciones estas incorporan una cantidad de funcionalidades que no tienen nada que ver con la lógica de negocio de la aplicación en si

Si creamos un software por ejemplo de facturación o de inventarios a parte de escribir el código que realiza la facturación o los inventarios hay que agregarle otras cosas igual de importantes y necesarias. Estas otras cosas que agregamos son funcionalidades de ***seguridad***, ***comunicación de red***, ***logging***, ***integridad de datos***, etc.

Cada aplicación que creamos debe tener todas estas funcionalidades, asi que la idea detrás de Java EE es reunirlas y ofrecerlas en un conjunto con el objetivo de que el desarrollador se concentre en lo que realmente le interesa que sería la lógica de negocios de su aplicación de facturación, inventarios, recursos humanos, etc.

Eso es Java EE, un soporte para desarrollar aplicaciones empresariales que ofrece todos estos servicios de ***seguridad***, ***comunicación***, ***logging***, ***integridad***, etc.

Hoy día con los cambios globales y la expansión tecnológica las empresas deben ser más competitivas y tener presencia en múltiples lugares. Todo esto implica tener procesos de negocios más complejos.

Que necesita una aplicación para satisfacer estas necesidades?

Básicamente una serie de características claves como las siguientes:

- **Segura :** Solo los usuarios autorizados pueden tener acceso.
- **Escalable :** Flexibilidad para crecer y soportar más carga de procesamiento.
- **Robusta :** Alto rendimiento y capacidad de reaccionar antes fallos.
- **Confiable :** Procesar información y mantenerla consistente y sin errores.
- **Distribuida :** Soportar instalación en múltiples lugares y poder comunicarse entre si.

Java EE proporciona todas estas funcionalidades las cuales son definidas a través de algo que conocemos como ***especificaciones***. Una *especificación* es básicamente un API que facilita el desarrollo de aplicaciones distribuidas, robustas, seguras, confiables, escalables y sobre todo ***multicapas y basadas en componentes***.

Ok, pero como se usan? Quien ofrece todas estas funcionalidades?

Servidores de Aplicación

Un servidor es una aplicación que esta ejecutandose y el software que tu escribiste es como un plugin que debe ser instalado en dicho servidor. El servidor proporciona un nivel de abstracción de las características ya mencionadas permitiendo que el desarrollador se concentre en su lógica de negocios.

Java EE al definir y estandarizar todas estas características permite que una aplicación pueda ser instalada en cualquier servidor que soporta las API de Java EE, también conocido como ***estándar***.

Los estándares son un conjunto de especificaciones(normas técnicas) conocidas como los **JSR**(Java Specification Release). Estos JSR son publicados para que empresas privadas y comunidades desarrollen implementaciones acordes a estas APIs.

Arquitectura Java EE

Java EE se basa en tres conceptos claves:

- Servicios

- Contenedores
- Componentes

Los Servicios : Son todas esas características de las que hablamos al comienzo. 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 Contenedores : Son entornos en tiempo de ejecución, es decir un programa que se esta ejecutando y tu aplicación la montas sobre este como si fuera un plugin o el cassette para un consola de juegos. Hay varios tipos de contenedores y la agrupación de ellos forman un ***servidor de aplicaciones***.

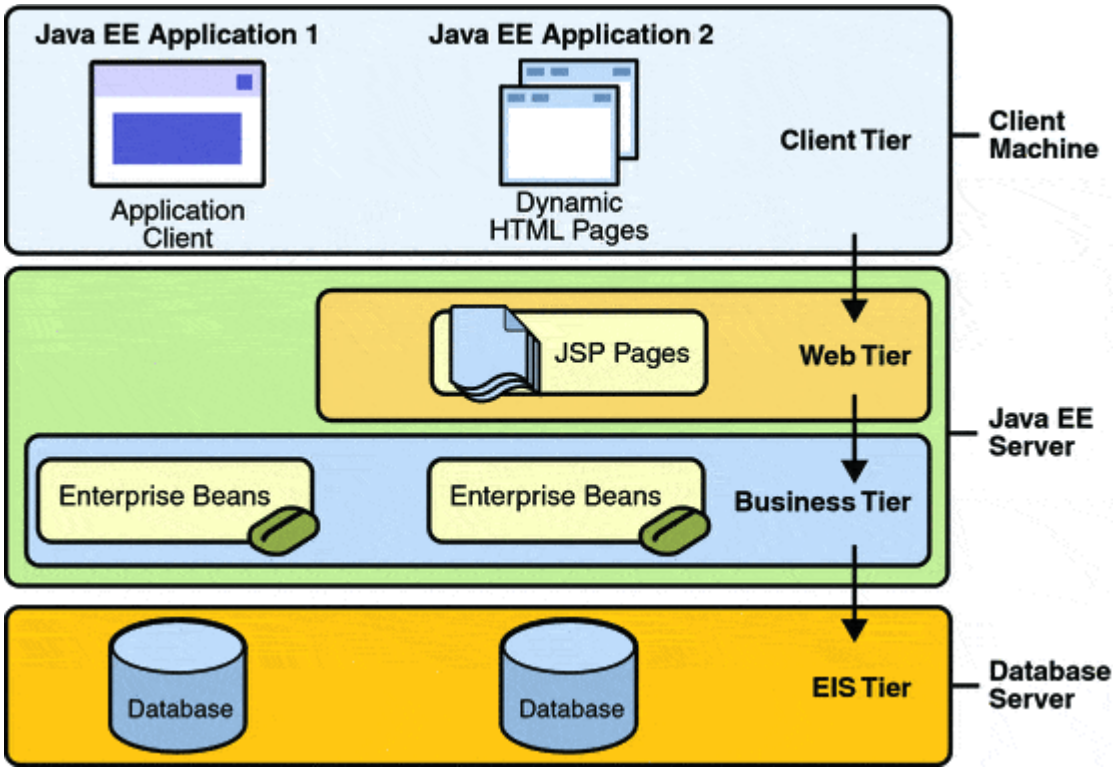
Los Componentes : Son objetos POJO que pueden ser reusados. Estos contienen la lógica de negocio de la aplicación y usan los servicios proporcionados por el contenedor. Hay varios tipos de componentes y según ese tipo son instalados(desplegados) en un contenedor u otro.

Los 3 conceptos anteriores permiten a Java EE definir una arquitectura de ***capas*** y ***distribuida***. Las capas definidas en el estándar Java EE son las siguientes:

1. **Client tier:** Componentes que corren en la máquina del cliente.
2. **Web tier:** Componentes que corren en un **contenedor web**.
3. **Business tier:** Componentes que corren en un **contenedor de negocio**.
4. **EIS tier:** Acceso a bases de datos y software heredado.

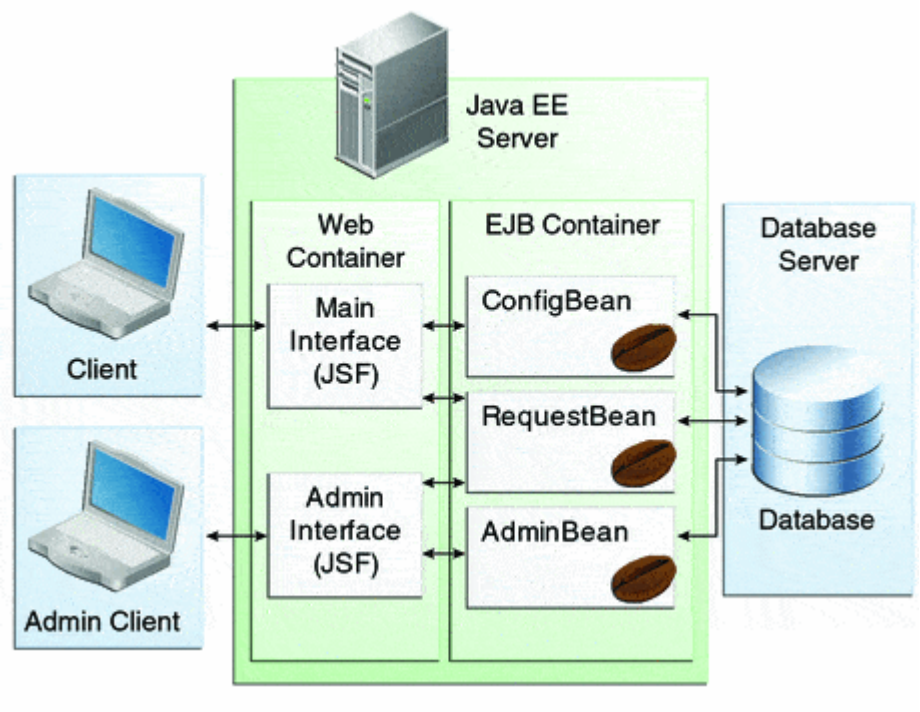
Las capas 2 y 3 son las principales hacia donde se enfoca Java EE, es decir la capa web y la capa de negocio. Ese es el dominio de acción de los servidores de aplicaciones. Recordemos que un servidor de aplicaciones esta formado por varios contenedores de componentes.

Esta es la imagen oficial de los tutoriales de Java EE:



Nota que aquí estamos hablando de una distribución de capas practicamente física. No la confundas con la separación de capas de la arquitectura de tu aplicación. Mira como el servidor de aplicaciones(Java EE Server) abarca la capa web(Web tier) y la capa de negocios(Business tier)

Aquí otra imagen que expresa el mismo concepto de capas distribuidas:



Nota como el servidor de aplicaciones(Java EE Server) tiene un contenedor web(Web container) y un contenedor de negocios(EJB Container).