**JEE**

*“Openandstandard-basedplatformfor developing, deploying and managing n-tier, web-enabled, server-centric, and component based enterprise applications”*

Plataforma abierta y basada en estandares para desarrollo, despliegue y manejo de n-tier, web-enable, server-centric y basado en componentes de aplicaciones empresariales.

*Introduccion a JEE*

* Orientada a aplicaciones empresariales
* Plataforma basada en componenetes
* Provee un conjuto de APIs (Seguridad, transacciones, Persistencia, etc)
* Provee una arquitectura Multicapa (3-tier, n-tier)

Tiene como objetivo proveer un conjunto de APIs para reducir el tiempo de desarrollo, su complejidad y mejorar el rendimiento de las aplicaciones.

También introduce un modelo simplificado de programación mediante el uso de XML y Anotaciones, Programacion basada en POJOs (Plain Old Java Objects) e inyección de dependencias.

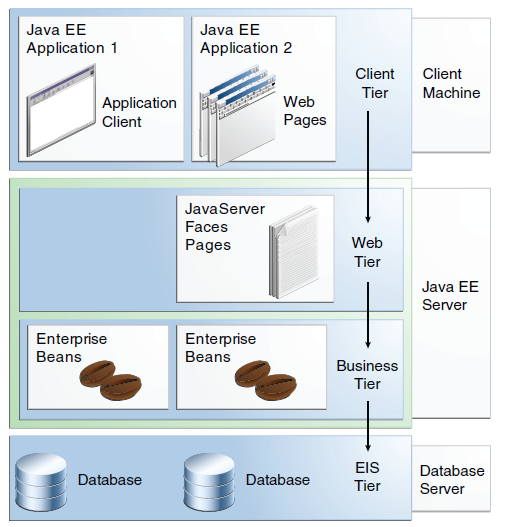
Desventajas

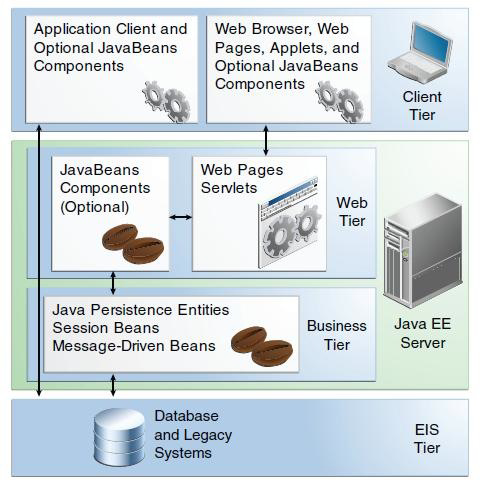
* Se desarrolla unicamente en Java
* Se necesita experiencia para el desarrollo

Ventajas

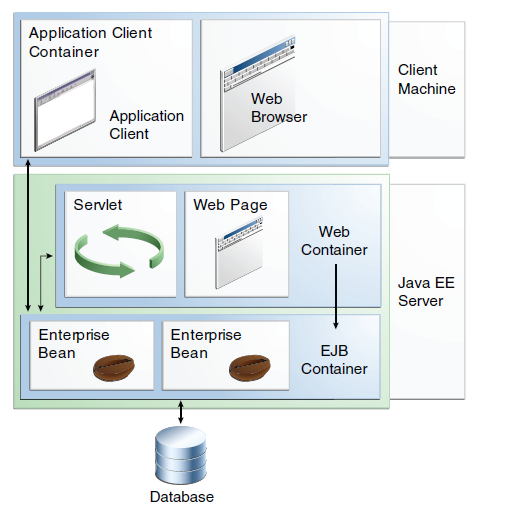
* Soporte de multiples sistemas operativos (Portabilidad)
* Soluciones libres (Software libre)
* Integracion de aplicaciones empresariales
* Reduccion del tiempo de desarrollo
* Mejora el rendimiento de las aplicaciones
* Los servidores JEE proporcionan beneficios a nivel de disponibilidad, seguridad, y escalabilidad entre otros.

*Arquitectura*





*Contenedores*



Los contenedores java son interfaces entre un componente java y las funcionalidades de bajo nivel específicas de la plataforma sobre la que se esta ejecutando.

Application Client Container: Contenedores que manejan la ejecución de los componentes en el cliente.

Web Container: Maneja la ejecucion de paginas web, servlets y algunos componentes EJB de aplicaciones JEE. Los componentes web y sus contenedores se ejecutan en JEE servers.

EJB Container: Maneja la ejecucion de los EJB para aplicaciones JEE. Enterprise beans y sus contenedores se ejecutan en JEE servers.

*Componentes de despliegue*

* Modulo EJB

El ensamblado del modulo contiene los archivos .class del modulo y un conjunto de metadata. Dentro de la metadata se encuentran los archivos descriptores de despliege: MANIFEST.MF y los archivos de despliegye ejb-jar.xml y el archivo glassfish-ejb-jar.xml (este ultimo opcional). Los modulos EJB se empaquetan en archivos .jar

* Modulo Web

Ensamblado computesto de los archivos de clase del modulo web, sus descriptores de despliegue web.xml y glassfish-web.xml (opcional). y un directorio con las librerias requeridas. Este conjunto de elementos es empaquetado en un archivo .war

* Enterprise application

Contiene Modulos Web y EJB y su descriptor de despliegue application.xml. Las enterprise application son empaquetadas en archivos .ear

**EJB**

*Que es EJB?*

Es un componente del lado del servidor que encapsula la logica del negocio de una aplicación.

*Cuando usar y cuando no usar*

Debemos considerar usar EJB si tu aplicación tiene alguno de los siguientes requerimientos:

* La aplicación debe ser escalable. Para ajustarse al número creciente de usuarios será necesario distribuir los componentes a lo largo de varias máquinas. Si bien los beans pueden ejecutarse en distintas maquinas pero su ubicación se mantendrá transparente para los clientes.
* La aplicación debe asegurar integridad de datos. Enterprise Beans soportan transacciones y los mecanismos que manejan la concurrencia del acceso a los objetos compratidos.
* La aplicación va a tener variedad de clientes. Con unas pocas líneas de código clientes remotos fácilmente pueden ubicar el Enterprise bean. Estos clientes pueden ser livianos, varios y numerosos.

*Beneficios*

* EJB containers proveen servicios sobre el sistema a los EnterpriseBeans lo que permite que al desarrollar el Bean uno se preocupe por resolver el problema del negocio mientras que el contenedor se encargara de resolver con el sistema problemas como manejo de transacciones y seguridad.
* Los beans contienen la lógica de negocio de la aplicación por lo que el los desarrolladores del cliente pueden enfocarse en la presentación del cliente, dado que el cliente no debe contener la implementación de las rutinas y procesos se tienen clientes mas livianos lo que cobra mayor importancia cuando el cliente se ejecuta en dispositivos pequeños.
* Los Enterprise beans son componentes portables, se puede realizar un build de nuevas aplicaciones a partir de beans existentes. Estas aplicaciones podrán ser ejecutadas en cualquier servidor JEE que provea las APIs estándar de java.

*Tipos*

* **Session Beans**

Un session bean encapsula logica de negocio que puede ser invocada programaticamente desde un cliente local, remoto o mediante web services

Representa una conversación temporal con un cliente, cuando el cliente finaliza su ejecucion, el bean de sesión y sus datos desaparecen.

Esta compuesto por una clase que implementa interfaces.

Dentro de los session beans tenemos tres subtipos

* Session beans con estado STATEFULL  
  Mantiene el estadon conversacional, para esto la instancia es reservada para el cliente y cada una almacena la informacion del cliente  
  La sesion finaliza si el cliente remueve el bean o finaliza su sesion  
  Anotacion @Stateful
* Session beans sin estado STATELESS  
  No mantiene estado conversacional  
  Son asignados por el contenedor a cualquier cliente de un pool de instancias  
  Ofrecen mejor escalabilidad  
  Pueden implementar web services  
  Annotacion @Stateless
* Singleton  
  Se instancia una vez por cada aplicación  
  Existe durante todo el ciclo de vida de la aplicación  
  Son diseñados para casos en los que una misma instancia de un bean es compartida y accedido concurrentemente por los clientes de la aplicación  
  Annotacion @Singleton
* **Message-driven beans**

***Tipos de acceso a Session beans***

El acceso a los session beans se realiza mediante interfaces (local o remota) o sin interfaces.

Los factores para determinar el acceso remoto o local son:

* Alto o bajo acomplamiento
* Tipo de cliente
* Distribucion de los componentes
* Rendimiento

*Localizacion de EJB (JNDI, inyección de dependencias)*

La localización con JNDI consta de que el EJB realiza un lookup entre los otros beans y recueros que cuenta el sistema y recupera el deseado. Con JNDI es responsabilidad del cliente hacer la localización y obtener la referencia al objeto

Sin interface

* *MiSB miSB= (MiSB)InitialContext.lookup("java:module/MiSB “);*

Con interface local  
- *MiSBLocal miSBLocal = (MiSBLocal )InitialContext.lookup("java:module/MiSBLocal");*

Con interface remota  
- *MiSBRemota miSBRemota = (MiSBRemota )InitialContext.lookup("java:****global****/miApp/ MiSBRemota ");*

Con EJB 3, se puede utilizar inyección de dependencias, dejando que el contenedor se responsabilice de inyectar el objeto requerido basado en la dependencia declarada @EJB

Sin interface

*@EJB  
MiSB miSB;*

Con interface local

*@EJB   
MiSBLocal miSBLocal;*

Con interface remota  
*@EJB*

*MiSBRemota miSBRemota;*

**MDB**

*Que son?*

Son un tipo de EJB que permite a las aplicaciones JEE procesar mensajes de forma asincrónica.

Caracteristicas

Message-driven beans have the following characteristics.

* They execute upon receipt of a single client message.
* They are invoked asynchronously.
* They are relatively short-lived.
* They do not represent directly shared data in the database, but they can access and update this data.
* They can be transaction-aware.
* They are stateless.

Anotacion: @MessageDriven

*Para que sirven?*

Permiten a las aplicaciones procesar mensajes de forma asincrónica a través del servicio JMS. Los MDB controlan las colas y procesan los mensajes que hay en ellas. A partir del mensaje procesado ejecutan los servicios que correspondan.

*Diferencias y semejanzas con Session Bean*

Diferencias

Los clientes no acceden a los MDB mediante interfaces, a diferencia de los sesión beans los MDB solo tienen una clase bean.

Semejanzas

Son muy similares a los stateless sesión beans  
No matienen información o estado conversacional alguno para un cliente especifico

Todas las instancias de un MDB son equivalentes, lo que le permite al contenedor EJB asignarle un mensaje a cualquier instancia del MDB. Con esto se posible tener un pool de estos recursos que permita el procesamiento de mensajes concurrentemente.

Un solo MDB puede procesar mensajes de multiples clientes.

*Explicacion sobre como se realiza la logica de procesamiento*

Cuando un mensaje llega a la cola o tópico a la que el MDB se encuentra registrado el contenedor hace una llamada al método onMessage del message driven bean.

El MDB procesa/parsea el mensaje o puede delegar su procesamiento.

**WS**

*Que es?*

*”Un Servicio Web (Web Service [WS]) es una aplicación software identificada por un URI (Uniform Resource Identifier ),* ***cuyas interfaces se pueden definir, describir y descubrir mediante documentos XML****. Los Servicios Web hacen posible la interacción entre aplicaciones utilizando mensajes XML intercambiados mediante protocolos de Internet.”*[W3]

Un servicio es un procedimiento, un método de un objeto con una interfaz estable y publica que puede ser invocado por un cliente

*Caracteristicas*

* Uso de estándares abiertos (HTTP, XML, SOAP)
* Favorece la Interoperabilidad
* Reduce el acoplamiento
* Expone servicios
* Permite a las aplicaciones compartir información de forma remota
* Utiliza XML para especificar los leguajes y protocolos necesarios, describe los servicios y representa el intercambio de información.

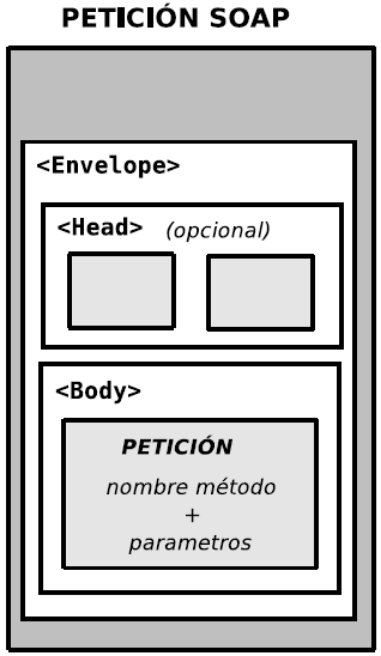
*Cuando usar y cuando no?*

*Protocolos*

SOAP

Simple Object Acces Protocol

Es un protocolo para el intercambio de información basado en XML a través de Internet (HTTP o SMTP). Los mensajes soap están compuestos por un Envelope, Header y un Body



WSDL

Describe los servicios web y sus interfaces mediante documentos XML.

Especifica cada método con sus parámetros de entrada y salida, expone información sobre la localización del servicio (URIs) y describe los mecanismos de interaccion a utilizar (bindings)

Los documentos WSDL un servicio web como una colección de puertos capaces de intercambiar mensajes, en la descripción de cada puerto se tiene una definición abstracta y una definición concreta.  
Elementos abstractos:   
Types (tipos y estrucuras de datos a intercambiar),   
Message: Definicion del mensaje a intercambiar incluyendo su nombre y su contenido  
Operation: Descripción de una acción admitida por el servicio  
Port Type: Define Grupos de Operaciones

Elementos concretos  
Binding: especificación de protocolo y del formato de datos para un tipo de puerto concreto

Port: combinación de un enlace y una dirección de red

Service: colección de puertos relacionados

UDDI

Universal Description, Discovery and Integration

Estandariza la publicación y la búsqueda de servicios web

Facilitar el enlace dinámico, permitiendo consultar referencias y acceder a servicios de interés en tiempo de ejecución.

Informacion ofrecida

Paginas blancas: Identificador y dirección de contacto de la empresa/organización que publica el ws

Paginas amarillas: descripción de los servicios web ofrecidos usando diferentes categorizaciones

Paginas verdes: Informacion técnica sobre los ws (URL de descarga del WSDL)

Transporte HTTP y SMTP

**JMS**

*Que es?*

Una especificacion que describre una forma comun para que los programas java creen, envien, reciban y lean mensajes. Permite una comunicación desacoplada, mensajería asincrónica y un reliable delivery

*Componentes de las aplicaciones JMS*

* Clientes JMS: programas java que envían y reciben mensajes
* Mensajes
* Administred Objects: Objeto JMS preconfigurado para ser usado por los clientes.
* Proveedor JMS: Sistema de mensajería que implementa JMS y funcionalidades administrativas

*Objetos de administración*

* ConnectionFactory  
  Son utilizadas por el cliente para crear una conexión con el proveedor JMS  
  Enapsulan un conjunto de parámetros de configuración que son definidos por el administrador.
* Conexión  
  Encapsula una conexión virtual con el prveedor JMS.  
  La connection se usa para crear una o mas sessions.
* Session  
  Una sesión es un contexto de un solo hilo para enviar y recibir mensajes  
  Las sesiones se usan para crear message producers, message consumers y messages
* Destinos  
  Un destino es el objeto que un cliente usa para especificar el destino del mensaje que produce y/o el origen de los mensajes que consume
  + Topic – Utilizado para PS
  + Queue – Utilizado para P2P
* Message Producer  
  Objeto creado por una sesión que es usado para enviar mensajes a un destinatario
* Message Consumer  
  Objeto creado por una sesión que es usado para recibir mensajes de un destinatario. El proveedor JMS es el encargado de entregar los mensajes a los consumers suscriptos a ese destino.
* Message Listener  
  Un message listener es un objeto que actua como un event handler asíncrono para mensajes. Implementa la interface MessageListener y contiene el método onMessage, este método es el encargado de manejar el mensaje que llega.

*Dominios de mensajes (PTP - PS) y características*

Point To Point

* Construido sobre el concepto cola de mensajes – queues
* Cada mensaje tiene un consumer

Las aplicaciones son construidas con message queues, senders y receivers. Las colas retienen los mensajes hasta que sean consumidos o expiren (tienen un tiempo de vida configurable). Cada mensaje es consumido por un único reciever que indica que es quien proceso el mensaje.

Publish Suscribe systems

* Utiliza Topics para enviar y recibir mensajes
* Cada mensaje tiene multiples consumidores

Los clientes envían mensajes a un Topic. Los Publishers y los Suscribers generalemente son anónimos y pueden publicar o suscribirse dinámicamente al contenido. El topic mantiene el contenido hasta que puede distribuirlo a todos los suscriptores actuales.

*Formas de consumir los mensajes*

Sincronica

El suscriber/receiver explícitamente solicita el mensaje del destino invocando el receive method. El receive metthod queda bloqueado hasta que llegue un mensaje (o tenga un timeout)

Asincronica

Un cliente registra un message listener al destino. Cuando un mensaje llega al destino, el proveedor JMS entrega el mensaje invocando el método del listener onMessage();

**JPA**

*Que es?*

Es la API que provee una herramienta de mapeo de objetos con bases de datos relacionales (ORM) con el objetivo de interactuar con bases de datos relacionales desde aplicaciones java. Para esto se mapean las entidades de la base de datos en POJOs.

JPA esta compusta de 4 partes:

* Java Persistence API
* Query Language JPQL
* Object relational mapping metadata
* Java Persistence Criteria API para realizar queries

*Que es un Entity*

Una entidad es un objeto persistente que representa una tabla en la base de datos. Cada instancia de una entidad representa un registro en la base de datos. Las entidades trabajan como POJOs.  
Se utilizan anotaciones de mapeo para establecer las relaciones entre entidades y para mapear las entidades con las tablas de la base de datos.

*Anotaciones principales y mapeos*

@Entity

@Id: Para identificar el PK

**@Id**

private Long id;

@Table: Mapear las clases con tablas

**@Table(name="CLIENTES“,schema=“DB\_SCHEMA\_CLIENTE”)**

@Column: Mapear las propiedades de las entidades a columnas de tablas

**@Column(name=”CODIGO\_CEDULA”, nullabe=false, unique=true)**

private int cedula;

**@GeneratedValue:** Provee diferentes estrategias para la generación de valores de la clave primaria.

**@Id**

**@GeneratedValue(strategy = GenerationType.AUTO)**

**@Column(name=”CLIENTE\_ID”, nullabe=false)**

private Long id;

**@Temporal:** Se utiliza para indicar que el tipo de formato de fecha que queremos utilizar.

**@Temporal(TemporalType.DATE)**

private Date fechaNacimiento;

Las entidades son asministradas por el EntityManager, este esta representado por instnacias de javax.persistence.EntityManager y cada una de estas intancias esta asociada a un persistence context.

*Persistence context*

Es un conjunto de instancias de entidades administradas que existen en un data store particular. Un persistence context define el ámbito en el cual se determinadas instancias de la entidad son creadas, persistidas y eliminadas. El contenido del persistence context se modifica mediante las operaciones realizadas sobre las entidades mediante el EntityManager.

*Consultas*

Dentro de JPQL se tienen 3 tipos de consulta:

* Dynamic Queries: reciben una cadena de caracteres que se forma dinámicamente para armar la consulta, puede recibir los parámetros por posiscion o nombre (? Por posición, : por nombre). Utilizan las entidades
* Named Queries: Queries definidas a nivel de la entity o en un archivo de configuración (orm.xml) y con compiladas al momento de inicializarse la aplicaion. Tiene un alcance global.
* Native Queries: Consultas nativas en el lenguaje del manejador de base de datos.