élogos

LA ARQUITECTURA JAVA EE

élogos

ÍNDICE

LA ARQUITECTURA JAVA EE

1. Capa Web y capa de negocio

Capas de una aplicación

Capa Web

Sus funciones son:

- Capturar los datos enviados por el cliente.
- Generar las respuestas en el formato apropiado.

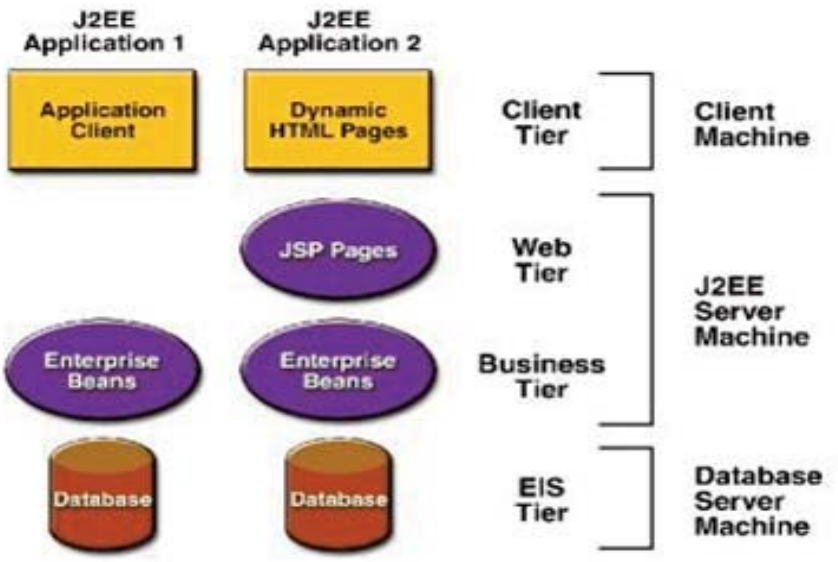
Capa de negocio

Sus funciones son:

- Encapsular toda la lógica de aplicación, incluyendo el acceso a los datos.

La especificación Java EE incluye dos tipos de componentes para la implementación de la capa Web:

Servlets y Java Server Pages



Servlets

Un servlet es un objeto Java que se encarga de gestionar las peticiones que llegan desde el cliente de una forma eficiente. Su ciclo de vida es gestionado por el contenedor Web.

Funciones

- Recibir requerimientos y enviar respuestas.
- Mantener información de sesión.
- Interactuar con otros servlets o componentes.

Java Server Pages(JSP)

Una página JSP contiene:

- Una plantilla de datos para formatear un documento Web (HTML o XML). Separa la presentación del procesamiento dinámico.

- Elementos y scriptlets JSP generan contenido dinámico. Las páginas JSP son transformadas en servlets en tiempo de ejecución EJB.

Los componentes Web son hospedados por Contenedores Web.

El contenedor Web se encarga de gestionar el ciclo de vida de un servlet, proporcionándole todos los servicios necesarios para su funcionamiento. Además, realiza la transformación de páginas JSP en servlet cuando la página es solicitada por primera vez.

Capa de negocio

La capa de negocio incluye todas las operaciones relativas a la lógica de aplicación. Todas las instrucciones de acceso a datos se codificarán en componentes dentro de esta capa.

La capa de negocio de una aplicación puede estar implementada por:

Clases estándares Java. Son clases normales Java en cuyos métodos se define la lógica de negocio de la aplicación, aislando al exterior de los detalles de implementación. No necesitan de ningún contenedor especial para su ejecución.

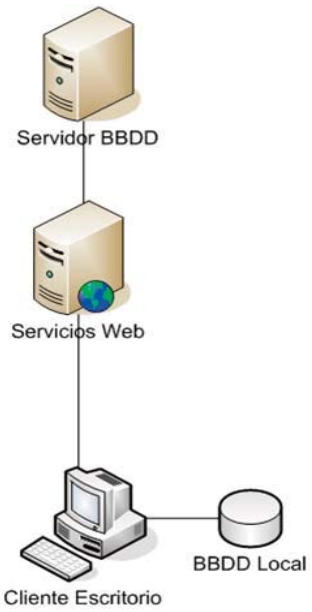
Enterprise JavaBeans (EJB). Se trata de componentes especiales que se ejecutan dentro de un contenedor EJB, el cual proporciona servicios de alto nivel que el programador no tiene que implementar, como la gestión automática de transacciones o el control de acceso a métodos.

En ambos casos se pueden emplear JavaBeans para el intercambio de datos entre las capas.

Servidor de aplicaciones y contenedores

La arquitectura de un servidor de aplicaciones incluye una serie de subsistemas:

- Servidor HTTP: también denominado servidor Web o servidor de páginas. Un ejemplo es el servidor Apache.
- Contenedor de aplicaciones o contenedor Servlet/JSP: un ejemplo de éste es Tomcat (que incluye el servicio anterior sobre páginas)
- Contenedor Enterprise Java Beans: contiene aplicativos Java de interacción con bases de datos o sistemas empresariales. Un ejemplo es JBoss que contiene a los anteriores (servidor de páginas web y contenedor de aplicaciones web).



Protocolo HTTP

El lenguaje básico de interconexión entre capas es el protocolo HTTP. Es un protocolo de aplicación, generalmente implementado sobre TCP/IP.

Es un protocolo sin estado, basado en solicitudes (request) y respuestas (response), que usa por defecto el puerto 80, basado en "peticiones y respuestas".

2. Componentes, contenedores y Servicios Java EE

Las tecnologías de la plataforma J2EE contienen: componentes, servicios y comunicaciones.

Componentes

Los componentes son utilizados por desarrolladores para crear partes esenciales de una Apps empresarial, utilizadas en la interface de usuario y lógica del negocio.

Tecnologías de componentes

Un componente es una unidad de software a nivel de aplicación. Además de los javabeans soportados por

J2SE, J2EE soporta:

- servlets(c)
- JSP
- EJB (s)

Todos los componentes dependen de una entidad de nivel de sistema llamada "**Contenedor**".

Los contenedores proveen a los componentes servicios como ciclo de vida, seguridad, multithreading, etc.

Tipos de componentes

En función de la capa a la que pertenecen, los componentes Java EE se dividen en dos grupos:

- Componentes Web: componentes que proveen respuesta a requerimientos. Genera interfaces de usuario para apps basadas en Web, se trata de los servlets y páginas JSP.

- Componentes EJB.

La arquitectura EJB es una tecnología del lado del servidor para desarrollo y ejecución de componentes para lógica de negocio de una aplicación empresarial. Los EJB son escalables, transaccionales y multiusuario y son de dos tipos:

Session Bean

Es creado para proveer algún servicio detrás de un cliente y normalmente durante una única sesión. Ejecuta operaciones como cálculos o accesos a bases de datos para el cliente.

Aunque un bean de session puede ser transaccional, no es recuperable si el contenedor se cae. Pueden ser stateless o pueden mantener estados conversacionales a través de métodos y transacciones.

El contenedor EJB administra el estado si éste tiene que ser movido de la memoria, pero el mismo objeto session bean tiene que gestionar sus datos.

Entity Bean

Es un objeto persistente que representa el mantenimiento de los datos en un almacenamiento. Está centrado en los datos.

Un EJB puede gestionar su propia persistencia o puede delegar esta función en el contenedor, puede vivir tanto como los datos que éste representa.

Es identificado por una clave primaria. Si el contenedor donde reside un EB cae, el EB, su clave primaria y referencias remotas sobreviven a la caída.

Desde la versión 3 de EJB, este tipo ha sido eliminado. En su lugar, se ha incluido dentro de la plataforma Java EE el api JPA para la gestión de persistencia dentro de una aplicación .

Contenedores Java EE

Los bloques más importantes que componen un servidor de aplicaciones son los contenedores. Un contenedor es un software que se encarga de gestionar el ciclo de vida de un componente y proporcionar al programador la implementación de un API que le permita realizar el desarrollo de dichos componentes.

Los servidores de aplicaciones Java EE proporcionan dos contenedores: El contendor Web y el contenedor EJB

Contenedor Web

Gestiona el ciclo de vida de servlets y JSP. Proporciona una implementación del API servlet, cuyas interfaces y clases principales se encuentran en el paquete javax.servlet y javax.servlet.http.

Contenedor EJB

Gestiona el ciclo de vida del EJB y proporciona una serie de servicios, como la gestión de transacciones o el pool de instancias.

Los servidores de aplicaciones, además de los contenedores y otros servicios, incorporan un servidor Web para el diálogo HTTP con el cliente.

Servicios deployment, transaccionales y de seguridad

Servicios deployment

Permite a componentes y aplicaciones ser personalizadas en el momento de ser empaquetadas.

Llevan información transaccional, de seguridad entre otros.

Servicios transaccionales

Las transacciones dividen una aplicación en una serie de unidades atómicas indivisibles de trabajo. Un sistema con soporte a transacciones asegura que cada unidad completa su trabajo sin interferencia de otros procesos. Éxito: commit, Falla: rollback

Servicios de seguridad

Aseguran que a los recursos sólo pueden acceder los usuarios autorizados para usarlos. Hay dos tipos de servicios de seguridad:

- Autenticación (username y password).
- Autorización: políticas del dominio de seguridad.

JTA y JNDI

Java Transaction API (JTA) permite a las aplicaciones acceder a transacciones de una forma independiente de la implementación específica y Java Naming and Directory Interface (JNDI), ofrece a las aplicaciones acceso a sistemas de directorio, donde se encuentran desplegados determinados tipos de objetos, como EJBs o DataSources.

Java Transaction API (JTA)	Java Naming and Directory Interface (JNDI)
Define interfaces entre gestores de transacciones y sus participantes involucrados en un sistema de transacciones distribuidas. El Java Transaction Service (JTS) especifica la implementación de un gestor de transacciones que soporta JTA e implementa el mapeo del Servicio de Transacciones de CORBA.	Ofrece métodos para operaciones sobre directorios: • Asociación de atributos con objetos. • Búsqueda de objetos basados en atributos. • JNDI es independiente de una implementación específica. • Arquitectura Connector. • Versión futura de J2EE. • Conexión entre la plataforma J2EE y sistemas de información empresarial. Los conectores serán la forma estándar de conectar legacy applications.

Tecnologías de comunicaciones

- Comunicación entre clientes y servidores.
- Comunicación entre objetos de diferentes servidores.
- Tecnologías de comunicaciones.
- Protocolos Internet.
- Protocolos de invocación remota de métodos.
- Protocolos OMG.
- Tecnologías de mensajería.
- Formatos de datos.
- Protocolos Internet.
- TCP/IP.
- HTTP 1.0.
- SSL 3.0.
- Protocolos de invocación de métodos remotos.

RMI utiliza interfaces Java para definir objetos remotos. Combina la serialización Java con el protocolo de métodos remotos de Java (JRMP) para convertir métodos locales en remotos.

Los Protocolos OMG permiten a objetos hospedados en J2EE acceder a objetos remotos desarrollados en CORBA y viceversa. La comunicación entre ORB se hace con

IIOP. Los objetos CORBA son definidos con IDL.

Las tecnologías OMG requeridas por la plataforma J2EE son **Java IDL** y **RMI-IIOP.**

Tecnología de mensajería

Las tecnologías de mensajería para el envío y recepción de mensajes asincrónicos son **Java message Service** (JMS) y **JavaMail API.**

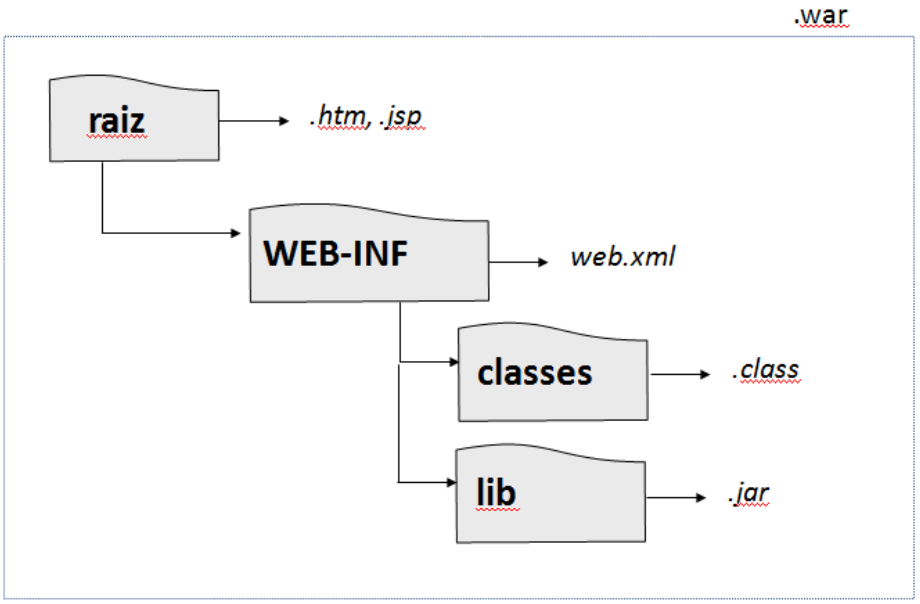
Es el preferido cuando hay:

- Requerimientos de velocidad y confiabilidad.
- Interacciones punto a punto.
- Mensajeria publish-subscribe.

JavaMail provee un conjunto de **clases abstractas** e interfaces que conforman un sistema de correo electrónico. Puede soportar diferentes implementaciones para almacenamiento, formatos y transporte.

3. Estructura de una aplicación Java EE

A la hora de crear una aplicación Web con Java EE, los archivos y componentes que la forman deben organizarse en la estructura de directorios que se indica en la siguiente imagen



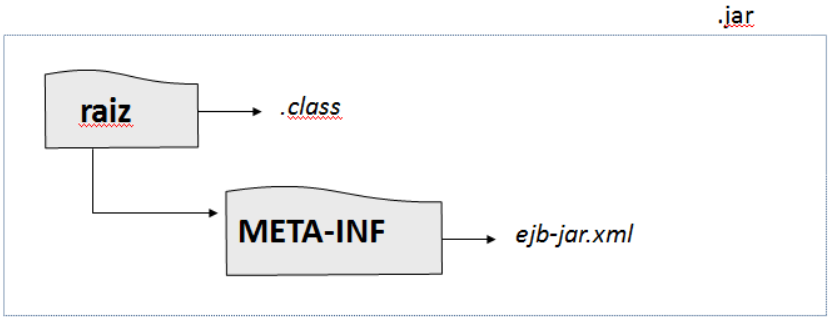
raíz. En el directorio raíz de la aplicación se sitúan las páginas JSP, así como los archivos de páginas estáticas .html, imágenes y hojas de estilo

WEB-INF. El resto de la aplicación deberá colgar de este directorio. En él deberá estar el archivo descriptor de despliegue web.xml

classes. En este directorio se sitúan las clases que componen la aplicación, es decir, servlets, JavaBeans y clases de lógica de negocio. Se deben organizar en paquetes

lib. Este directorio no es obligatorio crearlo. En el se deberán colocar los archivos de librerías externas que nuestra aplicación requiera para funcionar, como por ejemplo, los drivers JDBC.

Estructura de un módulo EJB



De cara a su distribución, la estructura se comprime en archivos .jar

Aplicación empresarial Java EE

Cuando las aplicaciones se componen tanto de módulos Web como de módulos EJB, se pueden comprimir ambos archivos dentro de otro con extensión .ear. Es lo que se conoce como aplicación empresarial.

La mayoría de los servidores de aplicaciones utilizan este tipo de archivos como estándar para la distribución de aplicaciones Java EE

El archivo de configuración web. xml

A través del fichero de configuración web.xml, se suministra al servidor de aplicaciones información sobre la aplicación, como por ejemplo, los servlets que la componen, páginas de bienvenida, configuración de sesiones, etc.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns=http://java.sun.com/xml/ns/javaee
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
   <servlet>
      <servlet-name>EjemploServlet</servlet-name>
      <servlet-class>servlets.EjemploServlet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>EjemploServlet</servlet-name>
      <url-pattern>/EjemploServlet</url-pattern>
   </servlet-mapping>
   <session-config>
      <session-timeout>
         30
      </session-timeout>
   </session-config>
   <welcome-file-list>
      <welcome-file>index.jsp</welcome-file>
   </welcome-file-list>
</web-app>
```

Registro de un servlet

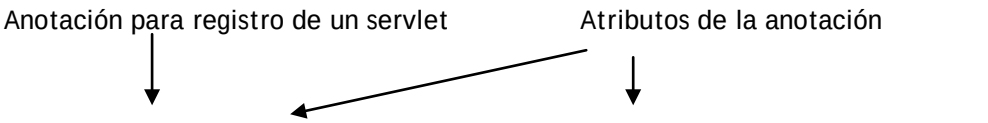
En posteriores módulos tendremos la oportunidad de analizar con detalle algunos de los parámetros más destacados del archivo descriptor de despliegue web.xml. De todos estos parámetros, cabe destacar la información de registro de un servlet:

```
<servlet>
      <servlet-name>EjemploServlet</servlet-name>
      <servlet-class>servlets.EjemploServlet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>EjemploServlet</servlet-name>
      <url-pattern>/EjemploServlet</url-pattern>
   </servlet-mapping>
```

<servlet>. Etiqueta para registro de un servlet. A través de los subelementos <servlet-name> y <servlet-class> se indican el nombre interno y clase de implementación del servlet, respectivamente.

<servlet-mapping>. Asocia al servlet una dirección de recurso. La dirección se indica a través de <url-pattern>

Desde la versión Java EE 6, ya no es obligatorio el uso del archivo descriptor de despliegue web.xml. La información de configuración de la aplicación se puede suministrar al servidor de aplicaciones a través de anotaciones incluidas en el propio código:




```
@WebServlet(name = "EjemploServlet", urlPatterns = {"/EjemploServlet"})
public class EjemploServlet extends HttpServlet {
   :
}
```