Introducción

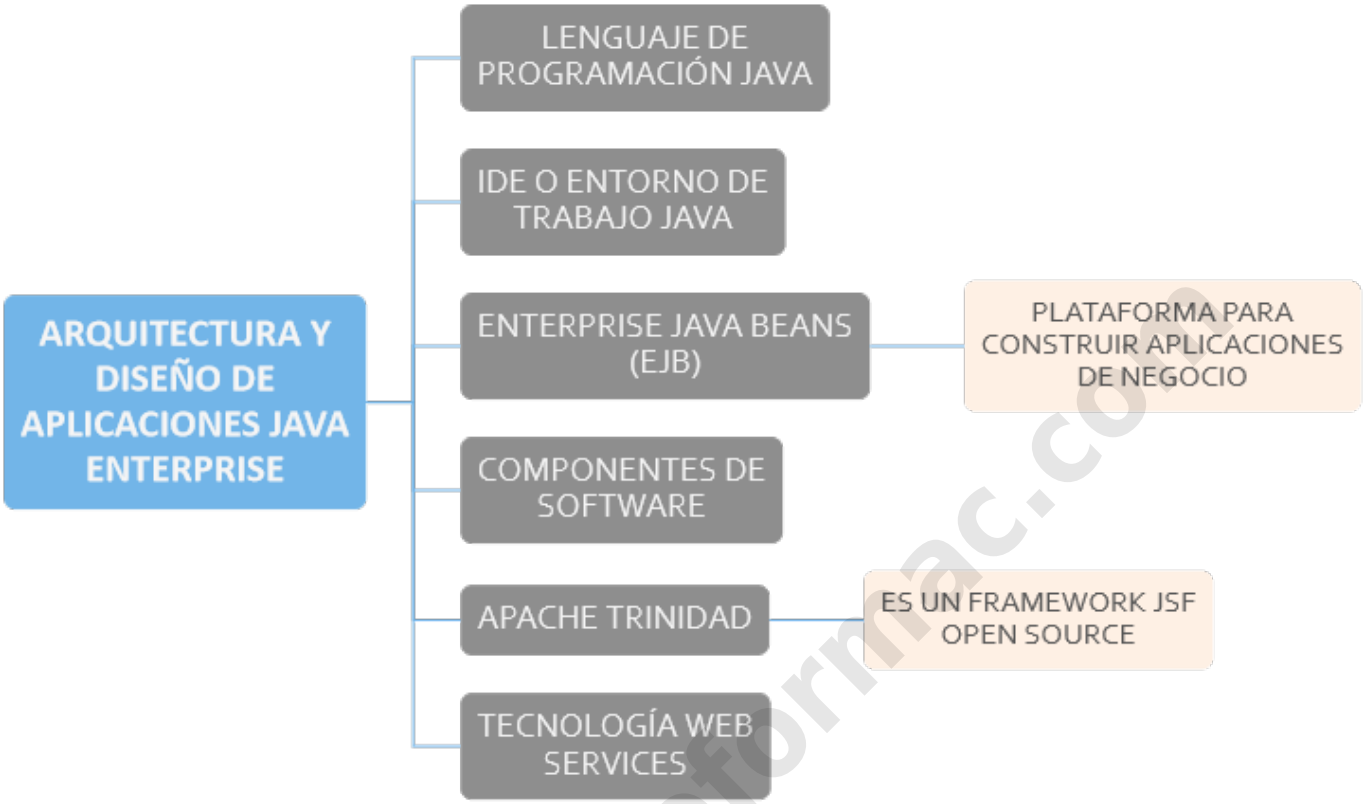
Java Enterprise Edtion o JAVA EE es una plataforma de programación para desarrollar y ejecutar aplicaciones en entorno Java, basada en arquitectura multicapa utilizando componentes modulares.

A lo largo de esta unidad de aprendizaje, conoceremos como Java EE tiene varias especificaciones de API, tales como JDBC, RMI, e-mail, JMS, Servicios Web, XML, etc y define cómo coordinarlos. Java EE también configura algunas especificaciones únicas para Java EE para componentes. Estas incluyen Enterprise JavaBeans, servlets, portlets (siguiendo la especificación de Portlets Java), JavaServer Pages y varias tecnologías de servicios web. Ello permite al desarrollador crear una aplicación de empresa portable entre plataformas y escalable, a la vez que integrable con tecnologías anteriores.

Objetivos

- Aprender a crear una clase Java
- Conocer las diferentes Ediciones de Java
- Estudiar las diferentes herramientas para ejecutar programas en Java
- Identificar los distintos modelos de componentes de Software

Mapa Conceptual



Lenguaje de programación Java

Java es un lenguaje de programacion muy completo, el cual es utilizado para la creacion de aplicaciones de todo tipo, desde aplicaciones para dispositivos moviles como aplicaciones empresariales de tipo web, aplicaciones de escritorio y aplicaciones de automatizacion de procesos.

Java como lenguaje es muy potente y versátil y ha sido adoptado en todas las industrias como:

- Financiera: Sistemas de automatización para aplicaciones de procesamiento de ordenes de compra e intercambio de valores en el mercado.

Estas aplicaciones son aplicaciones que requieren el procesamiento de un gran número de transacciones por segundo.

En esta industria también se generan aplicaciones de servicio al cliente para acceso a servicios bancarios y financieros:

- Defensa: El departamento de defensa de los Estados Unidos por ejemplo usa sistemas desarrollados en Java como controladores de diferentes dispositivos en tiempo real.
- Educación, Ingeniería, Biogenética, Información Geográfica, eCommerce, Logistica, ERP, etc.

Respecto a que herramientas y patrones de desarrollo debemos utilizar, todo depende del tipo de proyecto que vayamos a implementar.

En el mundo de Java para programar, no necesitamos más que un editor de texto y el compilador de Java javac.

Pero para ser más rápidos y eficaces en las tareas, es recomendable utilizar un IDE como Eclipse, NetBeans, IntelliJ, etc.

Eclipse y NetBeans son gratis y tienen una gran cantidad de componentes adicionales que facilitan el desarrollo de todo tipo de aplicaciones.

Con respecto a arquitectura, todo depende del tipo de aplicación y requerimientos.

Dada la flexibilidad de Java podemos crear aplicaciones desktop usando SWING o la plataforma de Eclipse.

Para Web podemos usar un servidor de aplicaciones open source o comercial y desarrollar un proyecto web application usando JSP, Servlets, y una serie de frameworks como Spring.

Plataforma Java EE

Existen varias Ediciones de Java, cada una de ellas diseñada para cierto entorno en particular.

Las ediciones de Java más importantes son:

- Java Standard Edition (Java SE).
- Java Micro Edition (Java ME).
- Java Entreprise Edition (Java EE).
- Java Card.

Java Standard Edition es la edición que se emplea en computadoras personales (desktops y laptops). Se le conoce también como Java Desktop (escritorio) y es la versión que necesitamos instalar para poder programar en Java en el ordenador.

A pesar de que su utilidad sirva para crear aplicaciones de escritorio, su enfoque está destinado a comprender el lenguaje Java y los conceptos básicos del lenguaje.

J2SE es una colección de APIs del lenguaje de programación Java útiles para muchos programas de la Plataforma Java.

La Plataforma Java Enterprise Edition incluye todas las clases en el entorno Java SE, además de librerías que son utilizados en cualquier desarrollo Java, independientemente del tipo de desarrollo.

Java Micro Edition es la edición de Java que se emplea en dispositivos móviles, tales como los teléfonos móviles.

Es una versión reducida de Java SE standard con ciertas extensiones enfocadas a las necesidades particulares de este tipo de dispositivos.

La plataforma Java Micro Edition, o Java ME (anteriormente J2ME), es una colección de APIs en Java orientadas a productos de consumo móviles, tales como PDAs, teléfonos móviles o electrodomésticos.

Java ME se ha convertido en una gran opción en el momento de crear un desarrollo para dispositivos móviles, como pueden ser juegos en PDAs.

Ofrece un entorno de desarrollo en el que podemos emular en un PC todo el entorno gráfico durante la fase debug del proyecto y, posteriormente, implementar el proyecto ya creado en el dispositivo móvil.

Al utilizar tecnologías Java en el desarrollo de aplicaciones o juegos con estas APIs, resulta bastante sencillo de migrar las aplicaciones a otros dispositivos, debido a que J2ME esuna "mini" edición de Java Enterprise Edition.

J2EE es un grupo de especificaciones diseñadas por Sun que permiten la creación de aplicaciones empresariales enfocadas al entorno web.

J2EE es solamente una especificación sobre un conjunto de "reglas" que deben seguir las páginas web de servidor para crear aplicaciones empresariales. Algunos productos siguen todas las especificaciones, pero no todos cumplen es estándar completo.

Java Card es la versión de Java enfocada a aplicaciones que se ejecutan en tarjetas de crédito con chip.

Es una versión muy recortada de Java. Un Java Card es una tarjeta capaz de ejecutar mini-aplicaciones Java. En este tipo de tarjetas el sistema operativo es una pequeña máquina virtual Java (JVM) y en ellas se pueden cargar dinámicamente aplicaciones desarrolladas específicamente para este entorno.

Existen diferentes versiones de Java que han ido apareciendo a lo largo del tiempo y dónde el entorno de desarrollo ha ido evolucionando, incluyendo mejoras en cada uno de las plataformas existentes.

Las versiones de Java existentes son las siguientes:

- **Java 1**
    - **Java 1.0** (1996): 8 paquetes, 212 clases - Es la primera versión pública.

La presión hizo que se hiciera pública demasiado pronto, lo cual significa que el diseño del lenguaje no es demasiado pronto y aparecieron multitud de errores en su entorno de clases. Dicha versión

creaba un código poco fiable debido a que apareció demasiado temprano y no solucionaron problemas del lenguaje.

- **Java 1.1** (1997): 23 paquetes, 504 clases - Mejoras de rendimiento en JVM, es decir, máquina virtual de Java, nuevo modelo de eventos en AWT, clases anidadas, serialización de objetos, API de JavaBeans, archivos jar, internacionalización, API Reflection (Reflexión), JDBC (Java Data base Connectivity), RMI (Remote Method Invocation).

Se implementó la firma del código y la autentificación. Fue la primera versión lo suficientemente estable y robusta.

- **Java 2**
    - **Java 1.2** (1998): 59 paquetes, 1520 clases - JFC (Swing), Drag and Drop, Java2D, Corba, API Collections.

Se produjeron notables mejoras en todos los niveles. Para darle más importancia a las mejoras, Sun lo renombró como "Java 2".

El JDK (Java Development Kit) se renombra al término SDK (Software Development Kit).

Se divide en J2SE, J2EE y J2ME.

- **Java 1.3** (2000) - 77 paquetes, 1595 clases: Orientada sobre todo a la resolución de errores y a la mejora del rendimiento. Se producen algunos cambios menores como la inclusión de JNDI (Java Naming and Directory Interface) y la API Java Sound.

También incluye un nuevo compilador de alto rendimiento JIT (Just In Time).

- **Java 1.4** (2002) - 103 paquetes, 2175 clases: También conocido como Merlin. Mejora notablemente el rendimiento y añade entre otros soporte de expresiones regulares, una nueva API de entrada/salida de bajo nivel (NIO, New I/O), clases para el trabajo con Collections, procesado de XML; y mejoras de seguridad como el soporte para la criptografía mediante las

Java Cryptography Extension (JCE), la inclusión de la Java Secure Socket Extension (JSSE) y el Java Authentication and Authorization Service (JAAS).

- **Java 1.5** (2004) - 131 paquetes, 2656 clases: También conocido como Tiger, renombrado por motivos de marketing como Java 5.0.

Incluye como principales novedades: tipos genéricos (generics), autoboxing/unboxing conversiones implícitas entre tipos primitivos y los wrappers correspondientes, Enumerados, Bucles simplificados, printf, funciones con número de parámetros variable, Metadatos en clases y métodos.

- **Java SE 6** (2006): También conocido como Mustang.

Sun cambió el nombre "J2SE" por Java SE y eliminó el ".0" del número de versión.

Los cambios más importantes introducidos en esta versión son: Incluye un nuevo marco de trabajo y APIs que hacen posible la combinación de Java con lenguajes dinámicos como PHP, Python, Ruby y JavaScript. Incluye el motor Rhino, de Mozilla, una implementación de Javascript en Java. Incluye un cliente completo de Servicios Web y soporta las últimas especificaciones para Servicios Web, como JAX-WS, JAXB, STAX y JAXP. Mejoras en la interfaz gráfica y en el rendimiento.

- **Java SE 7** - Nombre en clave Dolphin: En el año 2006 aún se encontraba en las primeras etapas de planificación. Su lanzamiento fue en julio de 2011. Las mejoras dentro de esta nueva versión son:

- Soporte para XML dentro del propio lenguaje.
- Un nuevo concepto de superpaquete.
- Soporte para closures.
- Introducción de anotaciones estándar para detectar fallos en el software.

- **Java 8** - la nueva fecha de lanzamiento para Java 8 final es el 18 de marzo de 2014. Java 8 es la próxima versión del lenguaje, y contendrá muchas mejoras. Su característica más conocida son los lambdas, que se pueden usar para instancias interfaces funcionales (interfaces con 1 solo método) con una sintaxis concisa. Sin embargo, Java 8 también incluye muchas otras

novedades interesantes.

Además de los cambios en el lenguaje Java, con el paso de los años se han efectuado muchos más cambios drásticos en la biblioteca de clases de Java (Java class library) que ha crecido de unos pocos cientos de clases en su primera versión JDK 1.0 hasta más de tres mil incluidas en la actualidad.

APIs completamente nuevas, como Swing y Java2D, han sido introducidas y muchos de los métodos y clases originales de JDK 1.0 han quedado obsoletas para el lenguaje y ya no se utilizan en los desarrollos.

Herramientas de desarrollo y servidor de aplicaciones

Un JDK|SDK ofrece las herramientas para compilar y ejecutar programas en Java. Podemos compilar las clases sin necesidad de tener un entorno gráfico sobre el que desarrollar, pero dicha tarea sería muy costosa dependiendo del tipo de proyecto, ya que existen multitud de configuraciones posibles dependiendo del tipo de aplicación empresarial que estemos desarrollando.

Un IDE o entorno de trabajo de Java, ofrece un ambiente de trabajo para proyectos complejos, es decir, si compilamos una o dos clases en un paquete o individualmente, es posible que el comando javac ofrecido en los JDK/SDK sea suficiente, pero si nuestros proyecto está compuesto por 100 o 200 clases, javac sería muy lento y costoso.

Los IDE's (Integrated Development Environment) ofrecen un entorno gráfico en el que se tiene acceso a mayor número de herramientas no ofrecidas en los SDK's.

Ofrece complidarores más elaborados, check-points dentro de la compilación, creación de WAR's (Web-Archives), "Wizards" para acelerar desarrollo, entre otras cosas.

Los entornos de desarrollo más utilizados hoy en día son:

- NetBeans (http://www.netbeans.org) Open-Source.
- Eclipse (http://www.eclipse.org) Open-Source
- Sun Java Studio (http://developers.sun.com/jsenterprise/index.jsp) de Sun
- JBuilder (http://www.codegear.com/products/jbuilder) de Borland
- WebSphere Studio (http://www306.ibm.com/software/awdtools/developer/application/) de IBM
- JDeveloper (http://www.oracle.com/technology/products/jdev/index.html) de Oracle

JavaBeans, anotaciones y registro

JavaBeans

Enterprise Java Beans (EJB) es una plataforma para construir aplicaciones de negocio portables, reutilizables y escalables usando el lenguaje de programación Java.

Desde el punto de vista del desarrollador, un EJB es una porción de código que se ejecuta en un contenedor EJB, que no es más que un ambiente especializado (runtime) que provee determinados componentes de servicio.

Los EJBs pueden ser vistos como componentes, desde el punto de vista que encapsulan comportamiento y permite reutilizar porciones de código, pero también pueden ser vistos como un framework, ya que, desplegados en un contenedor, proveen servicios para el desarrollo de aplicaciones empresariales, servicios que son invisibles para el programador y no ensucian la lógica de negocio con funcionalidades trasversales al modelo de dominio.

En la especificación 3.0, los EJB no son más que POJOs (clases planas comunes y corrientes de Java) con algunos poderes especiales implícitos, que se activan en tiempo de ejecución (runtime) cuando son ejecutados en un contenedor de EJBs.

Anotaciones

Para poder configurar correctamente un bean, debemos indicar al contenedor de EJB qué servicios debe proveer a nuestro Bean, y para ello disponemos de los descriptores de despliegue o de las anotaciones en las clases bean.

Vamos a visualizar las anotaciones básicas para los Session Beans.

@Stateful

Indica que el Bean de Sesión es con estado. Sus atributos son:

- Name: por defecto es el nombre de la clase pero se puede especificar otro nombre diferente.
- MappedName: Indica si deseamos que el contenedor maneje el objeto de forma específica. Si incluimos esta opción nuestra aplicación podría no ser portable y no funcione en otro servidor

de aplicaciones.

- Description: Indica la descripción de la anotación.

@Stateless

Indica que el Bean de Sesión es sin estado. Sus atributos son:

- Name: Por defecto el nombre de la clase pero se puede especificar otra diferente.
- MappedName: Si deseamos que el contenedor maneje el objeto de manera específica. Al igual que @StateFul esta opción podría hacer que nuestra aplicación no sea portable y no funcione en otro servidor de aplicaciones.
- Description: descripción de la anotación.

@Init

Especifica que el método se corresponde con un método create de un EJBHome o EJBLocalHome de EJB 2.1.

Sólo se podrá llamar una única vez a este método. Sus atributos son:

- Value: indica el nombre del correspondiente método create de la interfaz home adaptada. Sólo se debe utilizar cuando se utiliza el bean anotado con un bean con estado de la especificación 2.1 o anterior y que disponga de más de un método create.

@Remove

Indica que el contenedor debe llamar al método cuando quiera destruir la instancia del Bean. Sus atributos son:

- RetainIfException: indica si el Bean debe mantenerse activo si se produce una excepción. Por defecto su valor es false.

@Local

Indica que la interfaz es local.

@Remote

Indica que la interfaz es remota.

@PostActivate

Invocado después de que el Bean sea activado por el contenedor.

@PrePassivate

Invocado antes de que el Bean esté en estado passivate.

Normalmente las anotaciones más empleadas en las aplicaciones serán @Stateless y @Stateful, para indicar el tipo de EJB que estemos utilizando. El resto de anotaciones se utilizarán en casos más particulares.

Modelo de componentes Web

Definición de componente:

Una unidad de composición con interfaces especificadas de manera contractual y dependencias de contexto completamente explícitas.

Un componente de software puede desplegarse independientemente y es sujeto de composición por parte de terceros.

Los componentes de software posibilitan la reutilización de partes de software y la amortización de inversiones a lo largo de múltiples aplicaciones.

Existen otras partes para su reutilización, tales como bibliotecas de código fuente, diseños o arquitecturas.

Los componentes de software son unidades binarias de producción, adquisición y despliegue independientes, que al interactuar, forman un sistema en funcionamiento.

Los puntos de acceso a los componentes permiten a los clientes acceder a los servicios proporcionados por dicho componente.

Un componente puede tener varias interfaces, una por cada punto de acceso: uso, administración y configuración. No conviene tener varias interfaces similares o redundantes.

La especificación de las interfaces es un contrato. El respeto de este contrato por cliente y componente asegura el éxito de la interacción.

Modelos de componentes de Software:

- Microsoft .NET (COM): Perspectiva PC
- SUN J2EE: Perspectiva Internet
- OMG CORBA Component Model: Perspectiva de corporaciones empresariales. Es un superconjunto multi-lenguaje de la especificación EJB.

Los componentes más utilizados podemos diferenciarlos entre los siguientes

- Enterprise JavaBeans y el modelo de componentes de Java EE
- La estructura de componentes de .NET

La plataforma J2EE soporta un modelo de aplicación distribuida multinivel basado en componentes escritos en Java:

- Componentes cliente: aplicaciones de cliente y applets
- Componentes web: servlets y JavaServer Pages (JSP)
- Componentes de negocio: Enterprise JavaBeans (EJB)

Los componentes J2EE pueden incluir componentes basados en JavaBeans, pero éstos no son considerados parte de la especificación J2EE.

Objetivos por componentes de la arquitectura J2EE:

- Definir una arquitectura de componentes estándar para la construcción de aplicaciones distribuidas basadas en Java
- Separar los aspectos de lógica de negocio de otros soportados por la plataforma: transacciones, seguridad, ejecución multihilo y otros elementos de bajo nivel
- Filosofía java: Escribir una vez y ejecutar en cualquier parte
- Cubrir los aspectos de desarrollo, despliegue y ejecución del ciclo de vida de una aplicación.

Un contenedor es un proceso donde se ejecutan los componentes.

- Gestiona los componentes de la aplicación
- Ciclo de vida
- Proporciona acceso a servicios de la plataforma, como por ejemplo, transacciones, seguridad, conectividad.

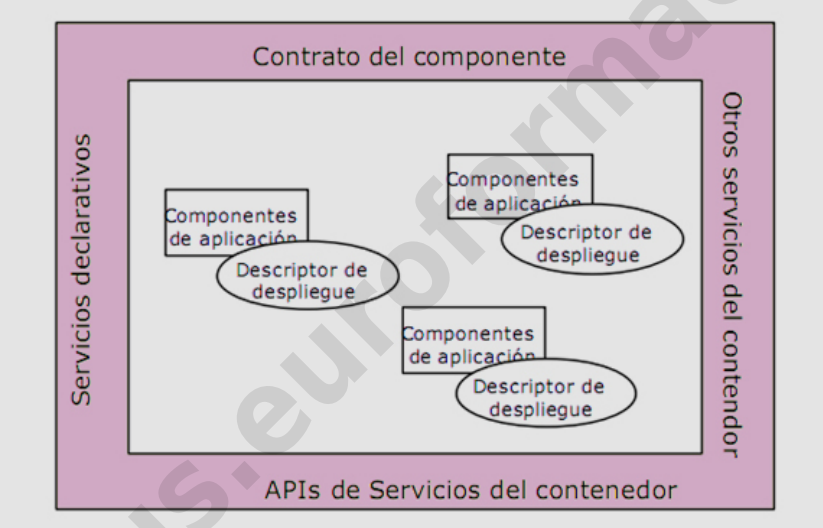
Cuando un desarrollador crea un modelo de arquitectura basado en la plataforma JEE, debe especificar los siguientes elementos en el contenedor:

- Los componentes de la aplicación

- Servlets
- JSPs (Java Server Pages)
- EJBs (Enterprise Java Beans)

- Los descriptores de despliegue

- Ficheros XML que describen los componentes de la aplicación.



Servlets y JavaServer Pages (JSP) son los componentes del nivel de presentación.

Permiten generar páginas web dinámicas

Servlets: Utilizan el código Java puro. Más fácil de controlar el flujo de acciones.

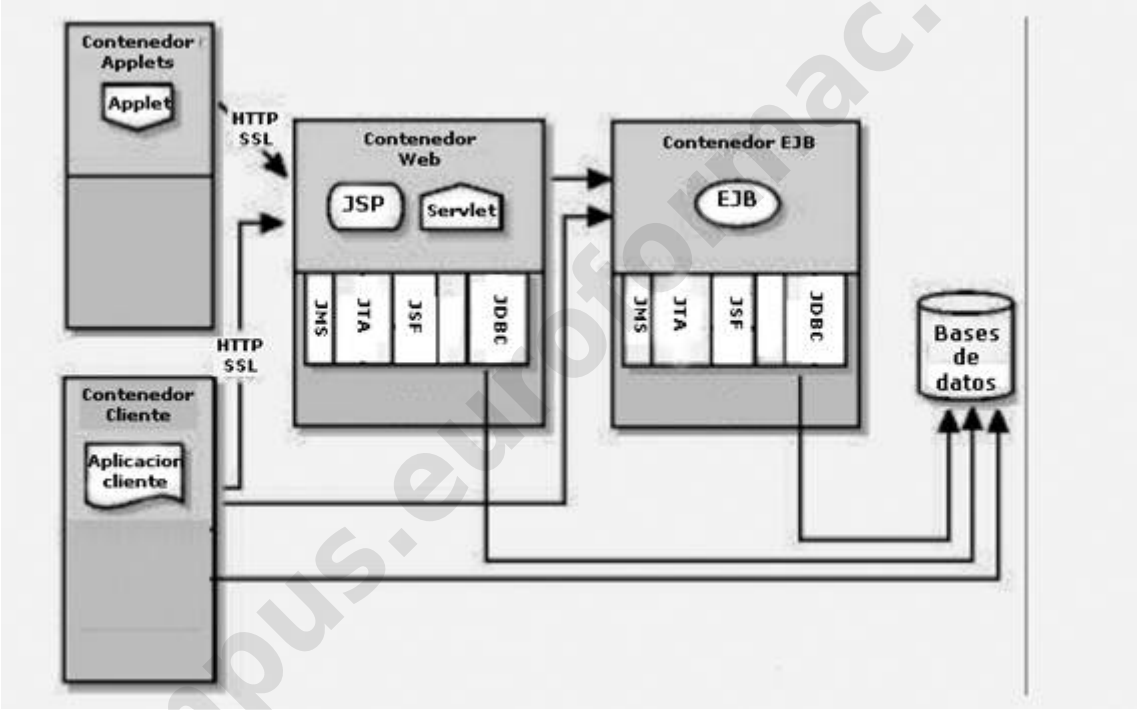
JSP: Lenguaje de marcado basado en etiquetas, es más fácil representar información en el lado del cliente.

Enterprise JavaBeans (EJB) es una completa especificación de arquitectura para componentes de servicio.

Objetivos de la arquitectura de componentes EJB:

- Facilitar el desarrollo de aplicaciones, concentrándose en la lógica de negocio: desarrollo, aplicación y aspectos de tiempo de ejecución.
- Lograr la independencia del proveedor de componentes mediante la especificación de interfaces.
- Lograr independencia de la plataforma
- Asegurar la compatibilidad con Java-APIs existentes, con sistemas de servidor de terceros y con protocolos de CORBA y de servicios Web.

Estructura de una aplicación JEE mediante componentes y contendores:



Desarrollo con tecnología Java Server Faces (JSF)

Java Server Faces (JSF) es un Framework de aplicaciones web basado en java, está orientado a la interfaz grafica del usuario (GUI), esto facilita su uso y construcción de las capas de controlador y vista. También basado en el patrón MVC (Modelo – Vista - Controlador).

Entre sus elementos principales encontramos un juego de componentes UI, modelo de programación de eventos y modelo de componentes que permiten suministrar elementos adicionales de terceros.

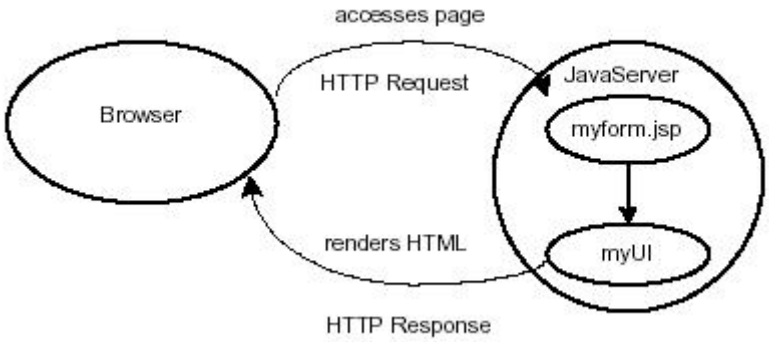
Sus ventajas son:

- Es más fácil la construcción de una interfaz para el usuario.
- Simplifica la migración de datos de las aplicaciones hacia y desde la interfaz del usuario.
- Controla el estado de la interfaz del usuario con solicitudes al servidor.
- Ofrece una separación entre el comportamiento y la presentación tradicionalmente ofrecida por la arquitectura UI del lado del cliente.
- Separación de la lógica – presentación, para que cada miembro de desarrollo web se centre en su parte a realizar.
- Su mayor objetivo es mejorar los conceptos de componente-UI y capa-Web sin limitarlos a una tecnología de script particular.

A continuación se muestra una tabla con las versiones de Java Server Faces:

Versiones	Fecha de lanzamiento	Descripción
JSF 1.0	11-03-2004	Lanzamiento inicial de las especificaciones de JSF.
JSF 1.1	27-05-2004	Lanzamiento que solucionaba errores. Sin cambios en las especificaciones ni en el renderkit de HTML.
JSF 1.2	11-05-2006	Lanzamiento con mejoras y corrección de errores.
JSF 2.0	12-08-2009	Lanzamiento con mejoras de funcionalidad, rendimiento y facilidad de uso.
JSF 2.1	22-10-2010	Lanzamiento de mantenimiento, con mínimos cambios.
JSF 2.2	16-04-2013	Lanzamiento que introduce soporte a HTML 5, Faces Flow, Stateless views y Resource library contracts.

Su estructura básica, como se puede apreciar a continuación, la interface de usuario que creamos con la tecnología JSF se ejecuta en el servidor y se rederiza en el cliente:



Myform.jsp, la pagina JSP, dibuja los componentes del interface de usuario con etiquetas personalizadas por JSF, el UI se representa por myUI y maneja los objetos referenciados por la pagina JSP.

Uso de AJAX y composición de componentes con JSF

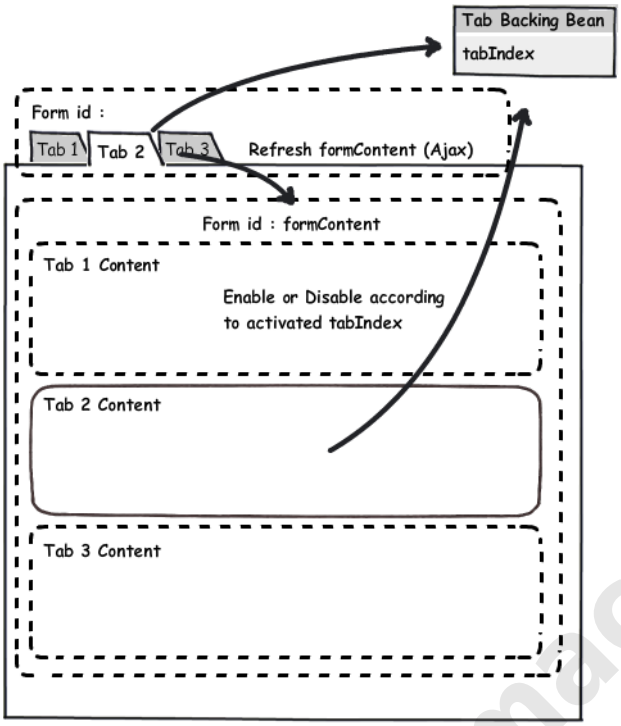
JavaServer Faces tiene soporte para Ajax en su versión 2.0, tiene una librería para ello. Con Ajax, las aplicaciones web pueden enviar datos y recuperarlos de un servidor de forma asíncrona, sin que tenga que intervenir con la visualización y el comportamiento de la página que existe.

Para usar Ajax con JSF se incluye el tag y definir los atributos que necesitemos, a continuación teneis una lista de sus atributos:

- Disabled: la acción Ajax no se ejecutara si se le da el valor true.
- Event: aquí se indica el evento que desencadenara la petición Ajax.
- Execute: se define el id del componente que se quiere enviar al servidor para procesarlo.
- Inmediate: su utilidad se basa en que si queremos que la acción se realice en la fase de inicio predefinida, fase 5 (invoke Aplication) o en la fase 2 (Apply request values) si lo establecemos a true.
- Listener: nombre del método que se ejecutara en respuesta a la petición Ajax.
- Onevent: nombre de la función JavaScript que se encarga de manejar el evento.
- Onerror: nombre de la función JavaScript que se encarga de manejar los errores.
- Render: aquí se indica el nombre del componente o componentes que se va a actualizar como consecuencia de la petición.

Para que se entienda mejor este concepto de uso de Ajax con JSF, vamos a centrarnos en el siguiente boceto donde tenemos tres pestañas y tres contenidos.

- Cuando el usuario hace clic en cada uno de los tab, se hace una llamada asíncrona a “tab manager backing vean” con el índice del tab que el usuario ha pulsado.
- El mismo botón envía una solicitud de actualización.
- Los tres contenidos le piden al Tab Backing Bean el ser visibles o no, de acuerdo al índice tab pulsado anteriormente.



Componentes JSF Apache Trinidad y desarrollo mñovil

Apache Trinidad un Framework JSF open source, es uno de los más completos, estable, probado con los 3 de apache, incluye una gran biblioteca de componentes de alta calidad, con muy buena solidez actualmente y demostrado con el paso de los años. Es resultado de una donación por parte de Oracle a ADF Faces de Apache.

El proyecto Apache MyFaces contiene varios subproyectos que están relacionados con la tecnología JSF. Contiene los siguientes elementos:

- Implementación con JavaServer Faces.
- Varias librerías de componentes “widgets UI” para la construcción de aplicaciones web.
- Paquete de extensiones para JavaServer Faces.
- Módulos de integración para otras tecnologías y estándares.

Entre sus ventajas se encuentran:

- Renderizado parcial de la página para todo el conjunto de componentes.
- Un framework para los diálogos de pantalla.
- pageFlowScope, comunicación entre páginas.
- Compatible con otras librerías.
- Funciona perfectamente bien en las aplicaciones de producción y tiene un desarrollo y evolución constante.

Requisitos

Precisa de los mismos requisitos que JavaServer Faces, debemos descargar e instalar una implementación de Java (JDK).

Devekionebt Kit para nuestra plataforma de sistema operativo, para la compilación se necesita la versión 1.4 JDK.

Contenedor de Servlet, tendremos que descargar e instalar un contenedor servlet, cualquiera en el que se pueda ejecutar la versión 1.1 de JSF.

A continuación tenéis una tabla con entornos soportados por los distintos componentes de Apache MyFaces: