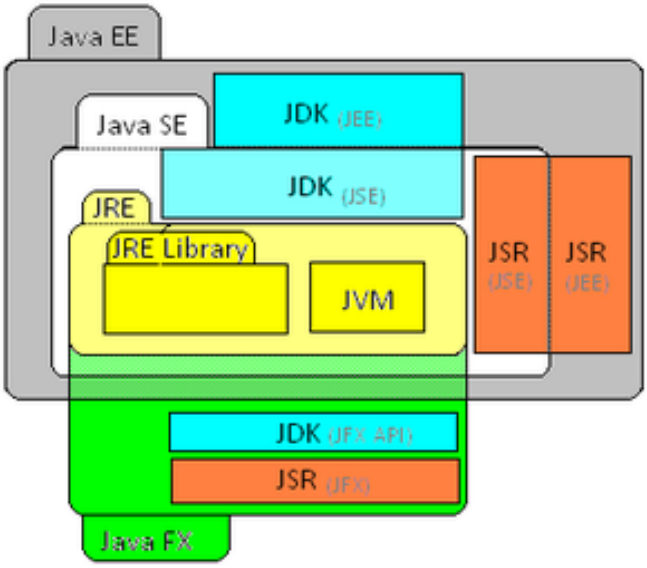
Temas de Investigación

7 Plataformas de Java y Tecnologías de Integración



La plataforma es así llamada la plataforma Java (antes conocida como Plataforma Java 2), e incluye:

- Plataforma Java, Edición Estándar (Java Platform, Standard Edition), o Java SE
- Plataforma Java, Edición Empresa (Java Platform, Enterprise Edition), o Java EE
- Plataforma Java, Edición Micro (Java Platform, Micro Edition), o Java ME

Las tecnologías que existen en la plataforma Java son:

- Java SE
- Java EE
- Java ME
- Java Card

Java SE

Java Platform, Standard Edition (Java SE) es una especificación que describe una plataforma Java abstracta. Proporciona una base para la compilación y despliegue de aplicaciones empresariales céntricas en la red que abarcan desde el sistema PC de escritorio al servidor de grupo de trabajo. Java SE lo implementa el kit de desarrollo de software (SDK) Java.

Rule Execution Server puede ejecutar conjuntos de reglas con código Java SE 100%. Existen muchos casos de uso para la ejecución Java SE pura, como por ejemplo la ejecución de lotes o la ejecución de reglas desde un proveedor JMS o un bus de servicio empresarial (ESB) que no sea Java EE.

Java EE

Java Platform, Enterprise Edition (Java EE) se basa en la especificación Java SE. Representa una colaboración entre diversos proveedores y líderes del sector y proporciona el soporte de infraestructura para las aplicaciones.

En la infraestructura Java EE, añade reglas:

- En la capa de la aplicación, para gestionar lógica empresarial dinámica y el flujo de tareas.
- En la capa de la presentación, para personalizar el flujo de páginas y el flujo de trabajo, y para construir páginas personalizadas basadas en estado de sesión.
- Java EE es portable y escalable, y da soporte a la integración con versiones anteriores y componentes basados en arquitectura EJB. Java EE simplifica las aplicaciones empresariales definiendo y especificando un complejo conjunto de servicios estándar comunes, como denominación, gestión de transacciones, simultaneidad, seguridad y acceso a base de datos.

Java EE también define un modelo de contenedor, que aloja y gestiona instancias de componentes de aplicaciones Java EE. Los contenedores están a su vez alojados en servidores Java EE.

Java ME

La plataforma Java Micro Edition (Java ME), o anteriormente Java 2 Micro Edition (J2ME), es una especificación de un subconjunto de la plataforma Java orientada a proveer una colección certificada de API de desarrollo de software para dispositivos con recursos restringidos. Está orientado a productos de consumo como PDAs, teléfonos móviles o electrodomésticos.

Java ME fue una buena opción para crear juegos en teléfonos móviles debido a que se puede emular en un PC durante la fase de desarrollo y luego subirlos fácilmente al teléfono. Al utilizar tecnologías Java el desarrollo de aplicaciones o videojuegos con estas API resulta bastante económico de portar a otros dispositivos. Sin embargo, pocos dispositivos actualmente utilizan la tecnología por la que poco a poco esta se ha ido al olvido.

Java ME fue desarrollado mediante el Java Community Process bajo la especificación JSR 68. La evolución de la plataforma ha propiciado el abandono de las Java Specification Request (peticiones de especificación para Java) en favor de JSRs separadas para las distintas versiones de Java ME.

Java Cards

Java Card es una tecnología que permite ejecutar aplicaciones basadas en Java (applets) en tarjetas inteligentes y dispositivos pequeños similares con recursos muy limitados. Java Card ofrece a los desarrolladores una forma estándar de desarrollar y desplegar applets en estos dispositivos.

La tecnología Java Card se basa en la plataforma Java, Standard Edition, y utiliza un subconjunto del lenguaje y las API de Java. Los applets de la tarjeta Java se ejecutan en una máquina virtual (VM) que se implementa en el hardware de la tarjeta inteligente. La VM proporciona un entorno independiente de la plataforma para los applets.

Los applets de la tarjeta Java pueden crearse para ejecutarse en cualquier plataforma de tarjeta Java, independientemente del hardware o el software subyacente. Esto permite portar los applets a diferentes tarjetas inteligentes y otros dispositivos con relativa facilidad.

7.1 Principales características de las tres plataformas de Java: J2SE, J2ME, y J2EE

J2SE

Java Platform, Standard Edition o Java SE (conocido anteriormente hasta la versión 5.0 como Plataforma Java 2, Standard Edition o J2SE), es una colección de API del lenguaje de programación Java útiles para muchos programas de la Plataforma Java. La Plataforma Java 2, Enterprise Edition incluye todas las clases en el Java SE, además de algunas de las cuales son útiles para programas que se ejecutan en servidores sobre workstations.

Comenzando con la versión J2SE 1.4 (Merlin), la plataforma Java SE ha sido desarrollada bajo la supervisión del Java Community Process. JSR 59 la especificación para J2SE 1.4 y JSR 176 especificó J2SE 5.0 (Tiger). En 2006, Java SE 6 (Mustang) está siendo desarrollada bajo el JSR 270.

J2ME

Java 2 Platform Micro Edition (J2ME) es una plataforma para desarrollar e implantar aplicaciones para dispositivos pequeños y con recursos limitados, como teléfonos móviles y PDA. J2ME consta de un conjunto de API de lenguaje de programación Java y una máquina virtual (VM) de tamaño reducido para ejecutar programas escritos en el lenguaje Java.

Las aplicaciones J2ME suelen estar escritas en lenguaje Java y utilizan las API de Java ME para acceder a las capacidades del dispositivo. Las aplicaciones J2ME se suelen empaquetar como archivos Java Archive (JAR) y se despliegan en los dispositivos mediante aprovisionamiento por aire (OTA).

Los dispositivos J2ME suelen tener una memoria, una capacidad de procesamiento y una duración de la batería limitadas, y pueden carecer de una interfaz gráfica de usuario (GUI). Las aplicaciones J2ME deben estar diseñadas para ejecutarse en estos dispositivos limitados.

La plataforma Java ME era conocida anteriormente como Java 2 Platform, Micro Edition (J2ME).

J2EE

Java Platform, Enterprise Edition o Java EE (anteriormente conocido como Java 2 Platform, Enterprise Edition o J2EE hasta la versión 1.4; traducido informalmente como Java Empresarial) es una plataforma de programación —parte de la Plataforma Java— para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java. Permite utilizar arquitecturas de N capas distribuidas y se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. La plataforma Java EE está definida por una especificación. Similar a otras especificaciones del Java Community Process (JCP), Java EE es también considerado informalmente como un estándar debido a que los proveedores deben de cumplir ciertos requisitos de conformidad para declarar que sus productos son conformes a Java EE; estandarizado por JCP.

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. Otros beneficios añadidos son, por ejemplo, que el servidor de aplicaciones puede manejar transacciones, la seguridad, escalabilidad, concurrencia y gestión de los componentes

desplegados, significando que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes en lugar de en tareas de mantenimiento del bajo nivel.

7.2 Beneficios y principales características de Java RMI

RMI (Java Remote Method Invocation) es un mecanismo ofrecido por Java para invocar un método de manera remota. Forma parte del entorno estándar de ejecución de Java y proporciona un mecanismo simple para la comunicación de servidores en aplicaciones distribuidas basadas exclusivamente en Java. Si se requiere comunicación entre otras tecnologías debe utilizarse CORBA o SOAP en lugar de RMI.

RMI se caracteriza por la facilidad de su uso en la programación por estar específicamente diseñado para Java; proporciona paso de objetos por referencia (no permitido por SOAP), recolección de basura distribuida (Garbage Collector distribuido) y paso de tipos arbitrarios (funcionalidad no provista por CORBA).

A través de RMI, un programa Java puede exportar un objeto, con lo que dicho objeto estará accesible a través de la red y el programa permanece a la espera de peticiones en un puerto TCP. A partir de ese momento, un cliente puede conectarse e invocar los métodos proporcionados por el objeto.

La invocación se compone de los siguientes pasos:

- Encapsulado (marshalling) de los parámetros (utilizando la funcionalidad de serialización de Java).
- Invocación del método (del cliente sobre el servidor). El invocador se queda esperando una respuesta.
- Al terminar la ejecución, el servidor serializa el valor de retorno (si lo hay) y lo envía al cliente.
- El código cliente recibe la respuesta y continúa como si la invocación hubiera sido local.

Así esta tecnología que permite a los programadores crear aplicaciones distribuidas en Java, en las que los métodos de objetos Java remotos pueden ser invocados desde otras máquinas virtuales Java, posiblemente en diferentes hosts.

7.3 Beneficios y principales características de JDBC, SQL, y tecnologías RDBMS

JDBC significa Java™ EE Database Connectivity (conectividad de bases de datos Java). En desarrollo de Java EE se trata de una tecnología muy conocida que se utiliza de forma habitual para implementar la interacción de bases de datos. JDBC es una API de nivel de llamada, lo que significa que las sentencias SQL se pasan como series a la API que, posteriormente, se encarga de ejecutarlas en RDMS. Por ello, el valor de estas series se puede modificar durante el tiempo de ejecución, haciendo que JBDC sea dinámica.

Mientras que los programas JDBC se ejecutan de forma más lenta que sus equivalentes SQLJ, una ventaja de este método es un concepto denominado "Write once, call anywhere" (escribir el código una sola vez y ejecutarlo en cualquier plataforma). Esto significa que, puesto que no se necesita ninguna interacción hasta el tiempo de ejecución, un programa JDBC es muy portable y se puede emplear entre dos sistemas distintos sin ningún tipo de preocupación.

- JDBC. permite que cualquier comando SQL pueda ser pasado al driver. directamente, con lo que una aplicación Java puede hacer uso.

- Con. el objetivo de conseguir que un driver sea compatible con SQL (SQL. compliant), se obliga a que al menos, el driver cumpla el Estándar.

Un sistema de gestión de bases de datos relacionales (RDBMS) es una base de datos que almacena elementos de datos y conjuntos de datos en función de sus conexiones con otros elementos. Estos sistemas de gestión de bases de datos utilizan tablas para mostrar las asociaciones entre los distintos componentes de los datos, que pueden utilizarse para organizar grandes conjuntos de datos con numerosos elementos. Las tablas de un RDBMS también pueden ser ventajosas porque almacenan los datos de forma que los usuarios puedan obtenerlos posteriormente. Los RDBMS se utilizan con frecuencia en los sistemas informáticos modernos y en las aplicaciones para ordenadores y dispositivos móviles.

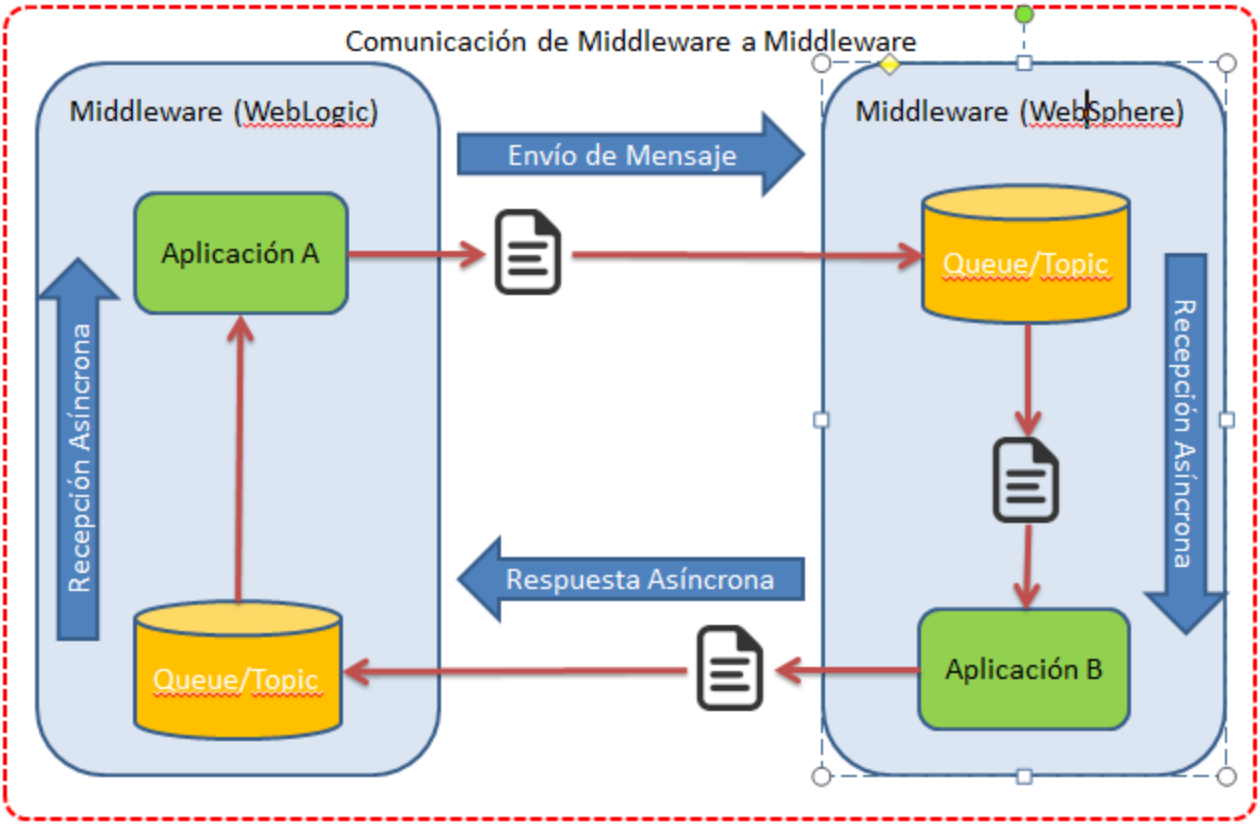
7.4 Beneficios y principales características de JNDI y tecnologías JMS

JNDI permite que las aplicaciones distribuidas busquen servicios de una manera abstracta e independiente de los recursos. El caso de uso más común es configurar un grupo de conexión de base de datos en un servidor de aplicaciones Java EE. Cualquier aplicación que se implemente en ese servidor puede obtener acceso a las conexiones que necesita utilizando el nombre JNDI java:comp/env/FooBarPoolsin tener que conocer los detalles sobre la conexión.Esto tiene varias ventajas:

- Si tiene una secuencia de implementación donde las aplicaciones se mueven de los devl->int->test->prodentornos, puede usar el mismo nombre JNDI en cada entorno y ocultar la base de datos real que se está utilizando. Las aplicaciones no tienen que cambiar a medida que migran entre entornos.
- Puede minimizar la cantidad de personas que necesitan conocer las credenciales para acceder a una base de datos de producción. Solo el servidor de aplicaciones Java EE necesita saber si usa JNDI.

Java Message Service es una especificación de la Java Community Process(JRS 914) la cual fue desarrollada originalmente para permitir a las aplicaciones Java comunicarse con los proveedores MOM o Middleware Orientado a Mensajes la cual buscaba los siguientes objetivos:

- Brindas un mecanismo de enrutamiento y entrega de mensajes.
- Entrega fiable de mensajes
- Soportar los patronos de entrega de mensajes Punto a Punto(P2P) y Publicador/Subscriptor (P/S)
- Brindar un mecanismo asíncrono de entrega de mensajes.



En el mercado podemos encontrar diferentes proveedores de JMS los cuales se encuentran embebidos en los principales Application Server como el caso de WebSphere, Wildfly, WebLogic, Glassfish, entre otros, sin embargo todos trabajan bajo el mismo estándar lo que permite que la comunicación entre ellos sea totalmente compatible.

Aunque JMS es un potente proveedor de Mensajería requiere de la configuración de Colas o Temas para poder depositar lo mensajes de entrada.

Tanto las colas como los temas son muy utilizados en los ambientes distribuidos donde se requerimos una comunicación asíncrona o One Wey(Solo ida) entre los distintos componentes que conforman nuestra infraestructura sin embargo estos se diferencias de los servicios web debido a que los mensajes son colocados en Cola o Temas para luego ser distribuidos a los destinatarios por lo que cambia un poco la forma en que los mensajes son administrados.

Java SE | Oracle Technology Network | Oracle

Oracle Java Technologies | Oracle

Oracle Java Technologies | Oracle

Java Platform, Micro Edition (Java ME) (oracle.com)

Wayback Machine (archive.org)

Aplicaciones Java SE y Java EE - Documentación de IBM

Tarjeta Java - Definición y explicación (techlib.net)

Java 2 Platform Micro Edition (J2ME) - Definición y explicación (techlib.net)

Java RMI - Introduction (tutorialspoint.com)

8 Tecnologías del lado del Cliente

Las tecnologías del lado del cliente con el lenguaje JAVA son aquellas que permiten crear aplicaciones web interactivas que se ejecutan en el navegador del usuario. Estas tecnologías incluyen los applets de JAVA, que son pequeños programas que se descargan desde el servidor y se ejecutan en una máquina virtual de JAVA integrada en el navegador; y los servlets de JAVA, que son componentes que se ejecutan en el servidor y generan contenido dinámico para el navegador. Ambas tecnologías utilizan el lenguaje JAVA, que es un lenguaje orientado a objetos, multiplataforma y con una amplia biblioteca de clases y métodos.

8.1 Beneficios del uso de Tecnologías del lado del cliente con HTML y JavaScript

HTML es el lenguaje que define la estructura y el contenido de una página web. Con HTML podemos crear elementos como encabezados, párrafos, listas, tablas, imágenes, enlaces, formularios, etc. HTML es la base de cualquier sitio web y es imprescindible para el desarrollo web.

CSS es el lenguaje que define el estilo y la apariencia de una página web. Con CSS podemos modificar el color, la fuente, el tamaño, el posicionamiento, el diseño, las animaciones y las transiciones de los elementos HTML. CSS nos permite crear páginas web atractivas y adaptativas a diferentes dispositivos y resoluciones.

JavaScript es el lenguaje que define el comportamiento y la funcionalidad de una página web. Con JavaScript podemos manipular los elementos HTML y CSS, crear eventos, validar formularios, hacer peticiones al servidor, mostrar datos dinámicos, usar APIs de terceros, etc. JavaScript nos permite crear páginas web interactivas y responsivas a las acciones del usuario.

El uso de estas tecnologías del lado del cliente tiene varios beneficios, entre los que podemos destacar:

- Mejora la experiencia de usuario, ya que se reduce el tiempo de carga de las páginas y se evita recargar la página completa cada vez que se realiza una acción.
- Aumenta la accesibilidad, ya que se puede adaptar el contenido y el diseño a las preferencias y necesidades del usuario, así como a las características del dispositivo y del navegador.
- Facilita la integración con otros servicios y plataformas, ya que se puede usar APIs de terceros para acceder a datos o funcionalidades externas, como Google Maps, Twitter, Facebook, PayPal, etc.
- Potencia la creatividad y la innovación, ya que se puede usar la combinación de HTML, CSS y JavaScript para crear efectos visuales, animaciones, juegos, aplicaciones web, etc.

Los frameworks de JavaScript son una parte esencial del desarrollo web front-end moderno, los cuales proveen a los desarrolladores herramientas probadas y testeadas para la creación de aplicaciones web interactivas y escalables. Muchas empresas modernas utilizan frameworks como parte estándar de sus

herramientas, por lo que muchos trabajos de desarrollo front-end en la actualidad requieren experiencia en frameworks.

8.2 Beneficios y principales características del uso de J2ME MIDlets

La máquina virtual de Java (JVM) interpreta el código de bytes de Java generado cuando se compila el programa. De este modo un programa Java puede ser ejecutado en cualquier dispositivo provisto de la máquina virtual de Java y de las librerías de clases apropiadas.

Las configuraciones están compuestas de una máquina virtual de Java y de un conjunto mínimo de librerías de clases. La JVM normalmente se encuentra en la parte más alta del sistema operativo del dispositivo en cuestión. La configuración define la funcionalidad mínima que debe cubrir una categoría o grupo de dispositivos particular. Define con respecto a la máquina virtual de Java las posibilidades mínimas que debe ofrecer y cuáles son sus requisitos para todos los dispositivos de una determinada categoría o grupo.

Actualmente, hay dos configuraciones de J2ME: Connected Limited Device Configuration (CLDC) enfocada a dispositivos con restricciones de procesamiento y memoria, y Connected Device Configuration (CDC) enfocada a dispositivos con más recursos.

J2ME (Java 2 Micro Edition) es una plataforma para desarrollar aplicaciones Java para dispositivos electrónicos con capacidades computacionales y gráficas muy limitadas, como teléfonos móviles, PDAs o electrodomésticos inteligentes. El objetivo principal de J2ME es descargar dinámicamente aplicaciones que aprovechen las posibilidades de cada dispositivo. J2ME proporciona el poder y los beneficios de la tecnología Java (portabilidad de código, programación orientada a objetos y ciclo de desarrollo rápido) a dispositivos pequeños. La plataforma está modularizada y personalizada con configuraciones, perfiles y paquetes opcionales que permiten abarcar la diversidad de dispositivos.

Un MIDlet es un tipo de aplicación en J2ME que se puede programar utilizando la configuración (CLDC) y el perfil (MIDP) para programar Midlets en teléfonos móviles. Los beneficios del uso de J2ME incluyen un fuerte uso del proceso de desarrollo trasplantado, portabilidad de código, programación orientada a objetos y ciclo de desarrollo rápido .

- Portabilidad: La portabilidad del código es una de las principales ventajas de J2ME. Puedes escribir tu código una vez y ejecutarlo en cualquier dispositivo que soporte J2ME, independientemente del sistema operativo subyacente.
- Programación Orientada a Objetos: Al igual que con cualquier aplicación Java, puedes aprovechar los principios de la Programación Orientada a Objetos (OOP) al desarrollar MIDlets. Esto puede hacer que tu código sea más modular, más fácil de mantener y reutilizar.
- Desarrollo Rápido: J2ME proporciona un conjunto rico de APIs que puedes utilizar para desarrollar aplicaciones. Esto puede acelerar significativamente el tiempo de desarrollo.
- Seguridad: J2ME proporciona un entorno seguro para ejecutar tus aplicaciones. Puedes controlar el acceso a los recursos del dispositivo y proteger tus datos.
- Interactividad: Los MIDlets pueden interactuar con el usuario a través de la interfaz gráfica de usuario (GUI). Puedes crear interfaces ricas e interactivas para tus aplicaciones.
- Conectividad: Los MIDlets pueden conectarse a Internet y comunicarse con servidores remotos. Esto te permite desarrollar aplicaciones que pueden acceder a datos en tiempo real.

- Pequeño tamaño de la aplicación: Dado que los MIDlets están destinados a dispositivos con recursos limitados, generalmente son pequeños en tamaño. Esto significa que puedes desarrollar aplicaciones que pueden ejecutarse en dispositivos con memoria limitada.
- Soporte para múltiples perfiles: J2ME soporta diferentes perfiles para diferentes tipos de dispositivos. Esto significa que puedes desarrollar aplicaciones específicas para ciertos tipos de dispositivos.

8.3 Beneficios y principales características del uso de Applets en Java del lado del cliente

Un applet es un programa Java™ diseñado para incluirse en un documento web HTML. Puede escribir su applet Java e incluirlo en una página HTML, de la misma forma que se incluye una imagen. Cuando utiliza un navegador habilitado para Java para ver una página HTML que contiene un applet, el código del applet se transfiere al sistema y lo ejecuta la máquina virtual Java del navegador.

El documento HTML contiene etiquetas, que especifican el nombre del applet Java y su URL (Uniform Resource Locator). El URL es la ubicación en la que residen los bytecodes del applet en Internet. Cuando se visualiza un documento HTML que contiene una etiqueta de applet Java, un navegador web habilitado para Java descarga los códigos de bytes Java de Internet y utiliza la máquina virtual Java para procesar el código desde dentro del documento web. Estos applets de Java son los que permiten que las páginas web contengan gráficos animados o contenido interactivo.

También puede escribir una aplicación Java que no requiera el uso de un navegador web.

Las aplicaciones son programas autónomos que no requieren la utilización de un navegador. Las aplicaciones Java se ejecutan iniciando el intérprete de Java desde la línea de mandatos y especificando el archivo que contiene la aplicación compilada. Generalmente, las aplicaciones residen en el sistema en el que se despliegan. Las aplicaciones acceden a los recursos del sistema y están restringidas por el modelo de seguridad Java.

Un Applet de Java es un programa que puede incrustarse en un documento HTML, es decir, en una página web1. Cuando un navegador carga una página web que contiene un Applet, este se descarga en el navegador web y comienza a ejecutarse. Esto permite crear programas que cualquier usuario puede ejecutar con tan solo cargar la página web en su navegador.

- Multiplataforma: Los Applets de Java son multiplataforma, lo que significa que funcionan en Linux, Windows, OS X y en cualquier sistema operativo para el cual exista una Java Virtual Machine.
- Independencia del Navegador y Sistema Operativo: La principal ventaja de utilizar Applets consiste en que son mucho menos dependientes del navegador que los scripts en Javascript, incluso independientes del sistema operativo del ordenador donde se ejecutan.
- Potencia de Java: Java es más potente que JavaScript, por lo que el número de aplicaciones de los Applets podrá ser mayor.
- Invocación Remota de Objetos: Los Applets pueden proporcionar invocación remota de objetos que se encuentran en Máquinas Virtuales diferentes. También pueden soportar llamadas a los servidores desde los Applets.
- Integración del Modelo de Objetos Distribuidos: Los Applets pueden integrar el modelo de objetos distribuidos en el lenguaje Java de una manera natural, conservando en medida de lo posible la semántica de los objetos Java.

- En resumen, los Applets de Java ofrecen una plataforma potente y flexible para el desarrollo de aplicaciones web del lado del cliente.

8.4 Beneficios y principales características del uso de Java Swing

Como caracteristicas de Swing en Java, también se le conoce como parte dentro de la librería Java Foundation Classes por sus siglas (JFC), fue un intento de resolver la mayor parte de las deficiencias que presentaba parte de la librería AWT también formada por Java. En Swing, Sun creó un muy bien diseño dando muchas caracteristicas; principalmente una vista más agradable, flexible y potente conjunto de herramientas. Desafortunadamente, esto significa que Swing necesita tiempo para aprender, y es a veces demasiado complejo para situaciones comunes. Swing está construido sobre las partes de AWT. Todas las partes de Swing son también parte de AWT. Swing utiliza el modelo AWT de evento y clases de apoyo, tales como colores, imágenes y gráficos. Para tener buenas bases de este lenguaje dejamos este curso de java presencial y 100% práctico recomendado por muchos usuarios que han tomado este curso de swing.

Java Swing nos brinda ciertas facilidades para la construcción de interfaces graficas de usuario en esta entrada vamos a conocer a nivel general algunos de los principales componentes que podemos usar en nuestras GUI's. Hasta el momento hemos visto lo que es Swing, las diferencias entre JFrame y JDialog así como un pequeño ejemplo sobre la creación de Ventanas incluyendo también 2 componentes tales como los botones y las etiquetas de texto. En esta entrada enunciaremos algunos de los principales componentes, tal vez no podamos trabajarlos todos pero es bueno saber que existe, conociendo nuevas opciones para vincular en nuestros desarrollos. Es importante para poder realizar todos estos elementos con Swing, tener una base solida de la programación en Java por ello dejamos este curso de java ampliamente recomendable.

Java Swing es una herramienta de interfaz gráfica de usuario (GUI) ligera que incluye un amplio conjunto de widgets1. Es parte de la JFC (Java Foundation Classes) en la plataforma Java2. La JFC proporciona facilidades para ayudar a las personas a construir GUIs2. Swing abarca componentes como botones, tablas, marcos, etc.

- Diseño en Java Puro: El diseño en Java puro posee menos limitaciones de plataforma. Esto significa que puedes desarrollar aplicaciones que funcionan en cualquier sistema operativo para el cual exista una Máquina Virtual Java.
- Desarrollo Activo: El desarrollo de componentes Swing es más activo. Esto significa que la comunidad está constantemente mejorando y añadiendo nuevas características a Swing.
- Soporte de Características: Los componentes de Swing soportan más características. Esto te permite crear interfaces de usuario ricas e interactivas.
- Doble Buffer Incorporado: El swing de Java proporciona principalmente el doble buffer incorporado. Esto puede mejorar el rendimiento de tus aplicaciones al reducir el parpadeo.
- Soporte para la Depuración: Swing también proporciona soporte para la depuración. Esto puede ayudarte a encontrar y solucionar problemas en tu código más rápidamente.
- Cambio de Apariencia: Los componentes de oscilación cambian principalmente su apariencia que se ve y se siente de la interfaz de usuario en función del paquete que se está utilizando.
- Independencia de la Plataforma: La biblioteca Swing está construida sobre el conjunto de herramientas de widgets abstractos, lo que la hace independiente de la plataforma.

¿Qué es JDBC? - Documentación de IBM

Cuales son las ventajas de la tecnologia JDBC? – RESPUESTASRAPIDAS

▶ ¿Qué es el RDBMS? (con definición y características principales) (historiadelaempresa.com)

¿Qué es el JNDI? ¿Cuál es su uso básico? Cuando se usa (qastack.mx)

Entendiendo los frameworks de JavaScript del lado del cliente - Aprende desarrollo web | MDN (mozilla.org)

Capítulo+3+-+Java+en+dispositivos+telefónicos+móviles.J2ME.pdf (us.es)

IBM Documentation

Caracteristicas de Swing - Cursos de Java (buscaminegocio.com)

Enterprise JavaBeans - Wikipedia, la enciclopedia libre

9 Tecnologías del lado del Servidor

En java existen varias tecnologías con las que se pueden construir sitios web:

- Java Servlet API
- Java Server Pages Technology
- JavaServer Faces Technology
- JDBC API
- Java Message Service API
- Java API for XML processing
- Java Naming and Directory Interface (JNDI)
- Java Persistence API

Java Servlet es la primera tecnología web del lado del servidor de Java que permite definir clases específicas de HTTP. La clase servlet amplía el poder y la capacidad de los servidores y aloja la aplicación. Estas aplicaciones se pueden acceder con el modelo de edición de solicitud-respuesta. Aunque un servlet puede responder a cualquier solicitud, el objetivo principal de usar un servlet es extender las aplicaciones alojadas por los servidores web. Por ejemplo, puede usar servlets para obtener datos de entrada de una aplicación en línea o incluso extenderla para que aparezca en la pantalla o en una página HTML.

Java Server Pages (JSP) es una tecnología muy popular entre los desarrolladores y proporciona una forma fácil de crear páginas web dinámicas. JSP es similar a las páginas HTML, pero también contienen código Java ejecutado en el lado del servidor.

JavaServer Faces (JSF) es un marco de trabajo para construir interfaces de usuario web para aplicaciones Java EE. JSF simplifica el desarrollo de componentes personalizados y proporciona una forma fácil de crear interfaces de usuario reutilizables.

Java Database Connectivity (JDBC) es una API para conectarse a bases de datos relacionales desde Java. JDBC proporciona una forma fácil y eficiente de interactuar con bases de datos relacionales desde aplicaciones Java.

Java Message Service (JMS) es una API para enviar mensajes entre diferentes componentes de aplicaciones distribuidas. JMS proporciona una forma fácil y eficiente de enviar mensajes entre diferentes componentes de aplicaciones distribuidas.

Java API for XML Processing (JAXP) es una API para procesar documentos XML desde Java. JAXP proporciona una forma fácil y eficiente de procesar documentos XML desde aplicaciones Java.

Java Naming and Directory Interface (JNDI) es una API para acceder a servicios basados en nombres y directorios desde Java. JNDI proporciona una forma fácil y eficiente de acceder a servicios basados en nombres y directorios desde aplicaciones Java.

Java Persistence API (JPA) es una API para interactuar con bases de datos relacionales desde aplicaciones Java EE. JPA proporciona una forma fácil y eficiente de interactuar con bases de datos relacionales desde aplicaciones Java EE.

9.1 Descripción de las características principales de las siguientes tecnologías: EJB, servlets, JSP, JMS, JNDI, SMTP, JAX-RPC, Web Services (incluyendo SOAP, UDDI, WSDL, y XML), y Java Mail

EJB

Las Enterprise JavaBeans (también conocidas por sus siglas EJB) son una de las interfaces de programación de aplicaciones (API) que forman parte del estándar de construcción de aplicaciones empresariales J2EE (ahora JEE) de Oracle Corporation (inicialmente desarrollado por Sun Microsystems).

Su especificación detalla cómo los servidores de aplicaciones proveen objetos desde el lado del servidor, que son precisamente los EJB:

- Comunicación remota utilizando CORBA.
- Transacciones.
- Control de la concurrencia.
- Eventos utilizando JMS (Java Messaging Service).
- Servicios de nombres y de directorio.
- Seguridad.
- Ubicación de componentes en un servidor de aplicaciones.
- La especificación de EJB define los papeles jugados por el contenedor de EJB y los EJB, además de disponer los EJB en un contenedor.

Los EJB proporcionan un modelo de componentes distribuido estándar del lado del servidor. El objetivo de los EJB es dotar al programador de un modelo que le permita abstraerse de los problemas generales de una aplicación empresarial (concurrencia, transacciones, persistencia, seguridad, etc.) para centrarse en el desarrollo de la lógica de negocio en sí. El hecho de estar basado en componentes permite que éstos sean flexibles y sobre todo reutilizables.

No hay que confundir los Enterprise JavaBeans con los JavaBeans. Los JavaBeans también son un modelo de componentes creado por Oracle - Sun Microsystems para la construcción de aplicaciones, pero no pueden utilizarse en entornos de objetos distribuidos al no soportar nativamente la invocación remota (RMI).

Existen tres tipos de EJB:

- EJB de Entidad (Entity EJB): su objetivo es encapsular los objetos del lado del servidor que almacena los datos. Los EJB de entidad presentan la característica fundamental de la persistencia: (nota: en la documentación de Java para JEE 5.0, los entity beans desaparecen, porque son remplazados por Java Persistence API o JPA)1
    - Persistencia gestionada por el contenedor (CMP): el contenedor se encarga de almacenar y recuperar los datos del objeto de entidad mediante el mapeo o vinculación de las columnas de una tabla de la base de datos con los atributos del objeto.
    - Persistencia gestionada por el bean (BMP): el propio objeto entidad se encarga, mediante una base de datos u otro mecanismo, de almacenar y recuperar los datos a los que se refiere, por lo cual la responsabilidad de implementar los mecanismos de persistencia es del programador.
- EJB de Sesión (Session EJB): gestionan el flujo de la información en el servidor. Generalmente sirven a los clientes como una fachada de los servicios proporcionados por otros componentes disponibles en el servidor. Puede haber dos tipos:
    - Con estado (stateful): en un bean de sesión con estado, las variables de instancia del bean almacenan datos específicos obtenidos durante la conexión con el cliente. Cada bean de sesión con estado, por tanto, almacena el estado conversacional de un cliente que interactúa con el bean. Este estado conversacional se modifica conforme el cliente va realizando llamadas a los métodos de negocio del bean. El estado conversacional no se guarda cuando el cliente termina la sesión.
    - Sin estado (stateless): los beans de sesión sin estado son objetos distribuidos que carecen de estado asociado permitiendo por tanto que se los acceda concurrentemente. No se garantiza que los contenidos de las variables de instancia se conserven entre llamadas al método.
- EJB Dirigidos por Mensajes (Message-driven EJB): son los únicos beans con funcionamiento asíncrono. Usando el Java Messaging System (JMS), se suscriben a un tema (topic) o a una cola (queue) y se activan al recibir un mensaje dirigido a dicho tema o cola. No requieren de su instanciación por parte del cliente.

Los EJB se disponen en un contenedor EJB dentro del servidor de aplicaciones. La especificación describe cómo el EJB interactúa con su contenedor y cómo el código cliente interactúa con la combinación del EJB y el contenedor.

Cada EJB debe facilitar una clase de implementación Java y dos interfaces Java. El contenedor EJB creará instancias de la clase de implementación Java para facilitar la implementación EJB. Las interfaces Java son utilizadas por el código cliente del EJB. Las dos interfaces, conocidas como interfaz "home" e interfaz remota, especifican las firmas de los métodos remotos del EJB. Los métodos remotos se dividen en dos grupos:

- métodos que no están ligados a una instancia específica, por ejemplo aquellos utilizados para crear una instancia EJB o para encontrar una entidad EJB existente. Estos métodos se declaran en la interfaz "home".
- métodos ligados a una instancia específica. Se ubican en la interfaz remota.

Dado que se trata simplemente de interfaces Java y no de clases concretas, el contenedor EJB genera clases para esas interfaces que actuarán como un proxy en el cliente. El cliente invoca un método en los

proxies generados que a su vez sitúa los argumentos método en un mensaje y envía dicho mensaje al servidor EJB. Los proxies usan RMI-IIOP para comunicarse con el servidor EJB.

El servidor llamará a un método correspondiente a una instancia de la clase de implementación Java para manejar la llamada del método remoto.

Servlets

Un servlet es una clase en el lenguaje de programación Java, utilizada para ampliar las capacidades de un servidor. Aunque los servlets pueden responder a cualquier tipo de solicitudes, estos son utilizados comúnmente para extender las aplicaciones alojadas por servidores web, de tal manera que pueden ser vistos como applets de Java que se ejecutan en servidores en vez de navegadores web. Este tipo de servlets son la contraparte Java de otras tecnologías de contenido dinámico Web, como PHP y ASP.NET.

La palabra servlet deriva de otra anterior, applet, que se refiere a pequeños programas que se ejecutan en el contexto de un navegador web.

El uso más común de los servlets es generar páginas web de forma dinámica a partir de los parámetros de la petición que envíe el navegador web.

1. Inicializar el servlet

Cuando un servidor carga un servlet, ejecuta el método init del servlet. El proceso de inicialización debe completarse antes de poder manejar peticiones de los clientes, y antes de que el servlet sea destruido.

Aunque muchos servlets se ejecutan en servidores multi-thread, los servlets no tienen problemas de concurrencia durante su inicialización. El servidor llama sólo una vez al método init al crear la instancia del servlet, y no lo llamará de nuevo a menos que vuelva a recargar el servlet. El servidor no puede recargar un servlet sin primero haber destruido el servlet llamando al método destroy.

2. Interactuar con los clientes

Después de la inicialización, el servlet puede dar servicio a las peticiones de los clientes. Estas peticiones serán atendidas por la misma instancia del servlet, por lo que hay que tener cuidado al acceder a variables compartidas, ya que podrían darse problemas de sincronización entre requerimientos simultáneos.

3. Destruir el servlet

Los servlets se ejecutan hasta que el servidor los destruye, por cierre del servidor o bien a petición del administrador del sistema. Cuando un servidor destruye un servlet, ejecuta el método destroy del propio servlet. Este método sólo se ejecuta una vez y puede ser llamado cuando aún queden respuestas en proceso, por lo que hay que tener la atención de esperarlas. El servidor no ejecutará de nuevo el servlet hasta haberlo cargado e inicializado de nuevo.

JSP

JavaServer Pages (JSP) es una tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas basadas en HTML y XML, entre otros tipos de documentos. JSP es similar a PHP, pero usa el lenguaje de programación Java.

El rendimiento de una página JSP es el mismo que tendría el servlet equivalente, ya que el código es compilado como cualquier otra clase Java. A su vez, la máquina virtual compilará dinámicamente a código de máquina las partes de la aplicación que lo requieran. Esto hace que JSP tenga un buen desempeño y sea más eficiente que otras tecnologías web que ejecutan el código de una manera puramente interpretada.

La principal ventaja de JSP frente a otros lenguajes es que el lenguaje Java es un lenguaje de propósito general que excede el mundo web y que es apto para crear clases que manejen lógica de negocio y acceso a datos de una manera prolija. Esto permite separar en niveles las aplicaciones web, dejando la parte encargada de generar el documento HTML en el archivo JSP.

Otra ventaja es que JSP hereda la portabilidad de Java, y es posible ejecutar las aplicaciones en múltiples plataformas sin cambios. Es común incluso que los desarrolladores trabajen en una plataforma y que la aplicación termine siendo ejecutada en otra.

Los servlets y Java Server Pages (JSPs) son dos métodos de creación de páginas web dinámicas en servidor usando el lenguaje Java. En ese sentido son similares a otros métodos o lenguajes tales como el PHP, ASP o los CGIs, programas que generan páginas web en el servidor. Sin embargo, se diferencian de ellos en otras cosas.

Para empezar, los JSPs y servlets se ejecutan en una máquina virtual Java, lo cual permite que, en principio, se puedan usar en cualquier tipo de ordenador, siempre que exista una máquina virtual Java para él. Cada servlet (o JSP, a partir de ahora lo usaremos de forma indistinta) se ejecuta en su propio hilo, es decir, en su propio contexto; pero no se comienza a ejecutar cada vez que recibe una petición, sino que persiste de una petición a la siguiente, de forma que no se pierde tiempo en invocarlo (cargar programa + intérprete). Su persistencia le permite también hacer una serie de cosas de forma más eficiente: conexión a bases de datos y manejo de sesiones, por ejemplo.

Las JSPs son en realidad una forma alternativa de crear servlets ya que el código JSP se traduce a código de servlet Java la primera vez que se le invoca y en adelante es el código del nuevo servlet el que se ejecuta produciendo como salida el código HTML que compone la página web de respuesta.

JMS

La API Java Message Service (en español servicio de mensajes Java), también conocida por sus siglas JMS, es la solución creada por Sun Microsystems para el uso de colas de mensajes. Este es un estándar de mensajería que permite a los componentes de aplicaciones basados en la plataforma Java2 crear, enviar, recibir y leer mensajes. También hace posible la comunicación confiable de manera asíncrona.

El servicio de mensajería instantánea también es conocido como Middleware Orientado a Mensajes (MOM por sus siglas en inglés) y es una herramienta universalmente reconocida para la construcción de aplicaciones empresariales.

Dicha API es parte integral de la versión 2 de Java.

Existen dos modelos de la API JMS, los cuales son:

- Modelo Punto a Punto (point to point) (P2P): Este modelo cuenta con solo dos clientes, uno que envía el mensaje y otro que lo recibe. Este modelo asegura la llegada del mensaje ya que si el

receptor no está disponible para aceptar el mensaje o atenderlo, de cualquier forma se le envía el mensaje y este se agrega en una cola del tipo FIFO para luego ser recibido según haya entrado.

- Modelo Publicador/Suscriptor (Publish/subscribe): Este modelo cuenta con varios clientes, unos que publican temas o eventos, y los que ven estos temas, a diferencia del modelo punto a punto este modelo tiende a tener más de un consumidor.

JNDI

La Interfaz de Nombrado y Directorio Java (Java Naming and Directory Interface) es una Interfaz de Programación de Aplicaciones (API) de Java para servicios de directorio. Permite a los clientes descubrir y buscar objetos y datos a través de un nombre. Como todas las APIs de Java que hacen de interfaz con sistemas host, es independiente de la implementación subyacente. Adicionalmente, especifica una interfaz de proveedor de servicio (SPI) que permite que las implementaciones del servicio de directorio sean integradas en el framework. Las implementaciones pueden hacer uso de un servidor, un fichero, o una base de datos; la elección depende del desarrollador.

JNDI organiza sus nombres en una jerarquía. Un nombre puede ser cualquier string tal como "com.mydomain.ejb.MyBean". Un nombre también puede ser un objeto que soporte la interfaz Name, sin embargo la forma más común de nombrar a un objeto es un string. Un nombre se asocia a un objeto en el directorio almacenando el objeto o una referencia JNDI al objeto en el servicio de directorio identificado por el nombre.

La API JNDI define un contexto que especifica donde buscar un objeto. El contexto inicial se usa normalmente como punto de partida.

En el caso más simple, un contexto inicial debe crearse usando la implementación específica y los parámetros extra requeridos por la implementación. El contexto inicial será usado para buscar un nombre. El contexto inicial es análogo a la raíz de un árbol de directorios para un sistema de ficheros. Debajo hay un ejemplo de cómo crear un contexto inicial:

SMTP

El Protocolo Simple de Transferencia de Correo (en inglés: Simple Mail Transfer Protocol o SMTP) es un protocolo de red utilizado para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos (PDA, teléfonos móviles, impresoras, etc.). Se encuentra en la capa de aplicación del modelo OSI1, la última de este modelo, en la que se dispone la interfaz entre las aplicaciones de comunicación y la red que transmite los mensajes2. Fue definido inicialmente en agosto de 1982 por el RFC 821 (para la transferencia) y el RFC 822 (para el mensaje), dos estándares oficiales de Internet que fueron reemplazados respectivamente por el RFC 2821 y el RFC 2822, posteriormente destituidos por los estándares RFC 5321 y RFC 5322.3

El funcionamiento de este protocolo se da en línea, de manera que opera en los servicios de correo electrónico. Sin embargo, posee algunas limitaciones en cuanto a la recepción de mensajes en el servidor de destino (cola de mensajes recibidos), por lo que se ejecuta normalmente en relación con otros, como POP3 o IMAP, otorgando a SMTP la tarea específica de enviar correos y delegando la de recibirlos a los protocolos antes mencionados.

El correo electrónico es presentado por un cliente de correo (MUA, agente de usuario de correo) a un servidor de correo (MSA, agente de sumisión de correo) usando SMTP a través del puerto 587. Una gran

parte de los proveedores de correo todavía permiten el envío a través del puerto 25. Desde allí, el MSA entrega el correo a su agente de transferencia postal mejor conocido como el MTA (Mail Transfer Agent, Agente de Transferencia de Correo). En algunas ocasiones, estos dos agentes son casos diferentes aunque hay que destacar que provienen del mismo software de donde fueron lanzados sólo que presentan opciones diferentes dentro de la misma máquina.

El procesamiento local que se presenta puede ser realizado en una sola máquina o partido entre varias aplicaciones; en este segundo caso, los procesos implicados pueden compartir archivos; aquí SMTP es usado para la transferencia de mensajes internamente, con cada uno de los hosts configurados para usar la siguiente aplicación como un anfitrión elegante. Para lograr la localización del servidor objetivo, el MTA divisorio tiene que usar el sistema de nombre de dominio (DNS) para lograr la búsqueda del registro interno de cambiado de correo conocido como registro MX para la esfera del recipiente (la parte de la dirección a la derecha). Es en ese instante cuando el registro de MX devuelto contiene el nombre del anfitrión objetivo.[cita requerida]

Luego el MTA se une al servidor de cambio como un cliente SMTP. Una vez que MX acepta el mensaje entrante, este a su vez se lo da a un agente de entrega de correo (MDA) para luego ser llevado a la entrega de correo local. El MDA, además de entregar mensajes es también capaz de salvar mensajes en un buzón de formato, y la recepción de correo puede ser realizada usando muchas computadoras. Hay dos formas en que un MDA puede entregar mensajes: ya sea enviándolos directamente al almacenamiento, o expedirlos sobre una red usando SMTP. Una vez entregado al servidor de correo local, dicho correo es almacenado para la recuperación de la hornada. Su recuperación se logra por medio de las aplicaciones de usuario final, conocidas como clientes de correo electrónico, usando el Protocolo de Acceso de Mensaje de Internet (IMAP), este protocolo que facilita tanto el acceso para enviar, como el manejo de correo almacenado.

Los administradores de servidor pueden elegir si los clientes utilizan TCP puerto 25 (SMTP) o el puerto 587 (Presentación) para retransmitir el correo saliente a una inicial del servidor de correo.4 Las especificaciones y muchos servidores soportan ambos. Aunque algunos servidores soportan el puerto 465 para el legado SMTP seguro en violación de las especificaciones, es preferible utilizar los puertos estándar y comandos SMTP estándar de acuerdo con RFC 3207, si se debe utilizar una sesión segura entre el cliente y el servidor.

Algunos servidores están configurados para rechazar toda la retransmisión en el puerto 25, pero los usuarios válidos de autenticación en el puerto 587 pueden retransmitir correo a cualquier dirección válida.[cita requerida] Algunos proveedores de servicios de Internet interceptan el puerto 25, volviendo a dirigir el tráfico a su propio servidor SMTP, independientemente de la dirección de destino. Esto significa que no es posible para sus usuarios acceder a un servidor SMTP fuera de la red del ISP a través del puerto 25.

Algunos servidores SMTP soportan el acceso autenticado en otro puerto que no sea 587 o 25 para permitir a los usuarios conectarse a ellos, incluso si el puerto 25 está bloqueado, pero 587 es el puerto estándar y ampliamente apoyada por los usuarios enviar correo nuevo. Microsoft Exchange Server 2013 SMTP puede escuchar en los puertos 25, 587, 465, 475, y 2525, en función de servidor y si los roles se combinan en un solo servidor. Los puertos 25 y 587 se utilizan para proporcionar la conectividad del cliente con el servicio de transporte en la parte delantera de la función de servidor de acceso de cliente (CAS). Los puertos 25, 465 y 475 son utilizados por el servicio de transporte de buzón de correo. Sin

embargo, cuando la función de buzón se combina con la función de CAS en un único servidor, el puerto 2525 se utiliza por la función de buzón de SMTP desde el servicio de transporte de extremo delantero del CAS, CAS, mientras que continúa para utilizar el puerto 25. Puerto 465 es utilizado por el servicio de transporte de buzón de correo para recibir las conexiones de cliente proxy de la función CAS. Puerto 475 es utilizado por la función de buzón para comunicarse directamente con otras funciones de buzón, la transferencia de correo entre el servicio de envío de transporte de buzón de correo y el servicio de entrega de transporte buzón.

JAX-RPC

JAX-RPC (Java API for XML-based RPC, Java Application Programming Interface for Extensible Markup Language - based Remote Procedure Call) permite a una aplicación Java invocar un servicio web (Web Service o WS) basado en Java con una descripción conocida sin dejar de ser consistente con su descripción Web Services Description Language (WSDL). Se puede ver como Java Remote Method Invocation (Java RMI) sobre servicios web.

JAX-RPC 1 está en desuso con Java EE 6.1 El servicio JAX-RPC utiliza estándares del W3C (World Wide Web Consortium) como WSDL.2

JAX-RPC 2.0 fue renombrado a JAX-WS 2.0 (Java API for XML Web Services).

SOAP

SOAP (originalmente las siglas de Simple Object Access Protocol) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Este protocolo deriva de un protocolo creado por Dave Winer en 1998, llamado XML-RPC. SOAP fue creado por Microsoft, IBM y otros. Está actualmente bajo el auspicio de la W3C. Es uno de los protocolos utilizados en los servicios Web.

SOAP es un paradigma de mensajería de una dirección sin estado, que puede ser utilizado para formar protocolos más completos y complejos según las necesidades de las aplicaciones que lo implementan. Puede formar y construir la capa base de una "pila de protocolos de web service", ofreciendo un framework de mensajería básica en el cual los web services se pueden construir. Este protocolo está basado en XML y se conforma de tres partes:

- Sobre (envelope): el cual define qué hay en el mensaje y cómo procesarlo.
- Conjunto de reglas de codificación para expresar instancias de tipos de datos.
- La Convención para representar llamadas a procedimientos y respuestas.

El protocolo SOAP tiene tres características principales:

- Extensibilidad (seguridad y WS-routing son extensiones aplicadas en el desarrollo).
- Neutralidad (bajo protocolo de transporte TCP puede ser utilizado sobre cualquier protocolo de aplicación como HTTP, SMTP o JMS).
- Independencia (permite cualquier modelo de programación).

Como ejemplo de cómo el modelo SOAP pueda ser utilizado, consideraremos un mensaje SOAP que podría ser enviado a un web service para realizar la búsqueda de algún precio en una base de datos, indicando para ello los parámetros necesitados en la consulta. El servicio podría retornar un documento en formato XML con el resultado, un ejemplo, precios, localización o características. Teniendo los datos

de respuesta en un formato estandarizado procesable (en inglés "parsable"), éste puede ser integrado directamente en un sitio Web o aplicación externa.

La arquitectura SOAP está formada por varias capas de especificación: MEP (Message Exchange Patterns) para el formato del mensaje, enlaces subyacentes del protocolo de transporte, el modelo de procesamiento de mensajes, y la capa de extensibilidad del protocolo. SOAP es el sucesor de XML-RPC, a pesar de que toma el transporte y la neutralidad de la interacción, así como el envelope / header / body, de otros modelos (probablemente de WDDX).

Un mensaje SOAP es un documento XML ordinario con una estructura definida en la especificación del protocolo. Dicha estructura la conforman las siguientes partes:

- Envelope (obligatoria): raíz que de la estructura, es la parte que identifica al mensaje SOAP como tal.
- Header: esta parte es un mecanismo de extensión ya que permite enviar información relativa a cómo debe ser procesado el mensaje. Es una herramienta para que los mensajes puedan ser enviados de la forma más conveniente para las aplicaciones. El elemento "Header" se compone a su vez de "Header Blocks" que delimitan las unidades de información necesarias para el header.
- Body (obligatoria): contiene la información relativa a la llamada y la respuesta.
- Fault: bloque que contiene información relativa a errores que se hayan producido durante el procesado del mensaje y el envío desde el "SOAP Sender" hasta el "Ultimate SOAP Receiver".

UDDI

UDDI son las siglas del catálogo de negocios de Internet denominado Universal Description, Discovery and Integration. El registro en el catálogo se hace en XML. UDDI es una iniciativa industrial abierta (sufragada por la OASIS) entroncada en el contexto de los servicios Web. El registro de un negocio en UDDI tiene tres partes:

- Páginas blancas - dirección, contacto y otros identificadores conocidos.
- Páginas amarillas - categorización industrial basada en taxonomías.

  Páginas verdes - información técnica sobre los servicios que aportan las propias empresas.

UDDI es uno de los estándares básicos de los servicios Web cuyo objetivo es ser accedido por los mensajes SOAP y dar paso a documentos WSDL, en los que se describen los requisitos del protocolo y los formatos del mensaje solicitado para interactuar con los servicios Web del catálogo de registros.

WSDL

WSDL, las siglas de Web Services Description Language, es un formato del Extensible Markup Language (XML) que se utiliza para describir servicios web (WS). La versión 1.0 fue la primera recomendación por parte del W3C y la versión 1.1 no alcanzó nunca tal estatus. La versión 2.0 se convirtió en la recomendación actual por parte de dicha entidad.

WSDL describe la interfaz pública a los servicios Web. Está basado en XML y describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Las operaciones y mensajes que soporta se describen en abstracto y se ligan después al protocolo concreto de red y al formato del mensaje.

Así, WSDL se usa a menudo en combinación con SOAP y XML Schema. Un programa cliente que se conecta a un servicio web puede leer el WSDL para determinar qué funciones están disponibles en el servidor. Los tipos de datos especiales se incluyen en el archivo WSDL en forma de XML Schema. El cliente puede usar SOAP para hacer la llamada a una de las funciones listadas en el WSDL.

El WSDL nos permite tener una descripción de un servicio web. Especifica la interfaz abstracta a través de la cual un cliente puede acceder al servicio y los detalles de cómo se debe utilizar.

XML

XML, siglas en inglés de eXtensible Markup Language, traducido como 'Lenguaje de Marcado Extensible' o 'Lenguaje de Marcas Extensible', es un metalenguaje que permite definir lenguajes de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible. Proviene del lenguaje SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML) para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí o integrar información.

XML no ha nacido únicamente para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande, con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

La tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible. Que la información sea estructurada quiere decir que se compone de partes bien definidas, y que esas partes se componen a su vez de otras partes. Entonces se tiene un árbol de trozos de información. Ejemplos son un tema musical, que se compone de compases, que están formados a su vez por notas. Estas partes se llaman elementos, y se las señala mediante etiquetas.

Una etiqueta consiste en una marca hecha en el documento, que señala una porción de este como un elemento. Un pedazo de información con un sentido claro y definido. Las etiquetas tienen la forma <nombre>, donde nombre es el nombre del elemento que se está señalando.

Java Mail

JavaMail es una API Java que facilita el envío y recepción de correo electrónico desde código java a través de protocolos SMTP, POP3 y IMAP. JavaMail está integrado en la plataforma Java EE, pero también proporciona un paquete opcional para su uso en Java SE.

La última versión liberada bajo la identificación de Java EE es la 1.6.2, publicada en agosto de 2018. Existe otra implementación JavaMail de código abierto - GNU JavaMail - aunque sólo soporta la versión 1.3 de la especificación JavaMail, además solo proporciona un único backend gratuito de NNTP, que permite utilizar esta tecnología para leer y enviar artículos de grupos de noticias.

A partir del 14 de septiembre de 2018 el proyecto de JavaMail fue movido de Oracle a Eclipse Foundation como parte de EE4J project,2 en el cual la última versión estable es la versión 1.6.3, liberada el 26 de noviembre de 2018.

9.2 Descripción de las características básicas del uso de servlets y JSP para HTML del lado del cliente

Las características básicas del uso de Servlets y JSP para HTML del lado del cliente son las siguientes:

- Independencia de la plataforma: Al estar escritos en Java, los Servlets y JSP son independientes de la plataforma.
- Eficiencia: Consumen menos recursos porque solo son cargados la primera vez que se solicitan sus servicios. Las siguientes peticiones crean hilos de ejecución.
- Seguridad y portabilidad: Se ejecutan bajo la misma máquina virtual de Java, lo que los hace seguros y portables.
- No requieren soporte para Java en el explorador del cliente: Operan en el dominio en el servidor y envían los resultados en HTML.
- Leer cualquier dato enviado por el usuario: Los datos normalmente se introducen por medio de la página Web, pero también pueden obtenerse a partir de un applet Java.
- Obtener otra información sobre la petición: Esta información se refiere, por ejemplo, a los cookies, el nombre del host de donde proviene la petición, etc.
- Generar los resultados: Esta parte puede requerir acceder a una base de datos, ejecutar una llamada RMI o CORBA, invocar a una aplicación o simplemente computar los datos de entrada.
- Generar un documento con los resultados: Debemos establecer el tipo de documento que va a ser devuelto (una página HTML, una imagen, un archivo comprimido, etc.).
- Establecer los parámetros apropiados para la respuesta.
- Enviar la respuesta al cliente: Una vez que tenemos el formato del documento que entregaremos como respuesta y tenemos establecidos los parámetros de la comunicación enviamos la respuesta al cliente.
- Cuando un Servlet acepta una llamada de un cliente, recibe dos objetos: Un ServletRequest, que encapsula la comunicación desde el cliente al servidor; y un ServletResponse, que encapsula la comunicación de vuelta desde el Servlet hacia el cliente1. Estos son interfaces definidos en el paquete javax.servlet.

9.3 Beneficios y principales características del uso de EJB

Los Enterprise JavaBeans (EJB) son una de las interfaces de programación de aplicaciones (API) que forman parte del estándar de construcción de aplicaciones empresariales J2EE (ahora JEE) de Oracle Corporation. Aquí te presento algunos beneficios y características principales del uso de EJB:

- Reutilización y compartición de lógica: Los EJB permiten reutilizar y compartir la lógica en múltiples aplicaciones o clientes con acoplamiento flexible.
- Escalabilidad y confiabilidad: Si se aplican una gran cantidad de solicitudes de varios mensajes, procesos o hilos de llamada, primero se distribuyen entre las instancias EJB disponibles en el grupo y luego se ponen en cola.
- Gestión de concurrencia: Los EJB manejan automáticamente la concurrencia, lo que permite a múltiples usuarios acceder a los recursos al mismo tiempo sin conflictos.

- Manejo automatizado de transacciones: Los EJB proporcionan un manejo automatizado de transacciones, lo que facilita el seguimiento y la gestión de las transacciones.
- Independencia de la plataforma: Al estar escritos en Java, los EJB son independientes de la plataforma.
- Seguridad: Los EJB operan bajo la misma máquina virtual de Java, lo que los hace seguros1.

9.4 Describir los beneficios fundamentales y desventajas del uso de la tecnología J2EE en servidores

La tecnología J2EE (Java 2 Platform, Enterprise Edition) proporciona un estándar de desarrollo de aplicaciones de empresa multinivel. Aquí te presento algunos beneficios y desventajas del uso de la tecnología J2EE en servidores:

- Beneficios:
    - Estandarización: J2EE proporciona un modelo de programación que mejora la productividad del desarrollo, estandariza la plataforma para alojar aplicaciones de empresa y asegura la portabilidad de las aplicaciones.
    - Multinivel: Permite el desarrollo profesional de aplicaciones empresariales distribuidas sobre una arquitectura multicapa.
    - Independencia de la plataforma: Al estar escritos en Java, los componentes J2EE son independientes de la plataforma.
    - Seguridad: Los componentes J2EE operan bajo la misma máquina virtual de Java, lo que los hace seguros.
- Desventajas: Aunque J2EE tiene muchos beneficios, también tiene algunas desventajas que deben tenerse en cuenta:
    - Complejidad: J2EE puede ser complejo para los desarrolladores que son nuevos en Java o en el desarrollo de aplicaciones empresariales.
    - Rendimiento: Aunque Java ha mejorado mucho en términos de rendimiento, todavía puede ser más lento en comparación con otros lenguajes como C++.
    - Costo: Aunque Java es gratuito, algunos servidores de aplicaciones J2EE pueden ser costosos.

JavaServer Pages Technology (oracle.com)

Welcome to The Apache Software Foundation!

Java Message Service (JMS) (oracle.com)

Java Naming and Directory Interface (oracle.com)

Programación en castellano. Sistema de Nombrado en Java (JNDI) [Parte I] (archive.org)

SMTP: el requisito para enviar correos electrónicos - IONOS

Capa 7 OSI | Capa de Aplicación - El Taller del Bit

STD1 (archive.org)

Protocolos de mensajería (SMTP, POP3 e IMAP4) - CCM

In First Year, DMARC Protects 60 Percent of Global Consumer Mailboxes – dmarc.org

Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language (w3.org)

Extensible Markup Language (XML) (w3.org)

Tutorial de XML | Abrirllave.com

XML Extended Data Type | Microsoft Learn

El Lenguaje XML # (archive.org)

JavaMail (archive.org)

Empezar con JavaMail (chuidiang.org)