Introducción a las arquitecturas de componentes y a Java EE

Autores: Simon Pickin
Natividad Martínez Madrid
Pablo Basanta Val

Dirección: Departamento de Ingeniería Telemática
Universidad Carlos III de Madrid
España

Versión: 1.0

Software de Comunicaciones

Agradecimientos

- "Enterprise JavaBeans. Grundlagen - Konzepte - Praxis. EJB2.0/2.1"
  Martin Backschat, Otto Gardon
  Spektrum Akademischer Verlag, 2002
  ISBN: 3-8274-1322-2

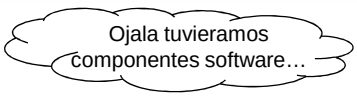
Software de Comunicaciones

Contenidos

- Introducción
  - desarrollo basado en componentes
  - aplicaciones y arquitecturas empresariales
- Arquitecturas cliente-servidor y multi-tier
  - conceptos básicos
  - nivel de cliente
  - nivel de presentación
  - nivel de negocio
  - nivel de datos
- La plataforma Java Enterprise Edition
  - introducción
  - la arquitectura y los contenedores de Java EE
  - los componentes de Java EE
  - los servicios de Java EE

Software de Comunicaciones

Desarrollo basado en componentes: origen



- 1968: Congreso de la OTAN sobre Ing. del software, Garmisch, Alemania
  - estudiar cómo contrarrestar la “crisis del software”
  - ver: `http://homepages.cs.ncl.ac.uk/brian.randell/NATO/`
- Charla de Doug McIlroy en este congreso

  título: *Mass produced software components*
  (Componentes de software fabricados en serie)
  - tema: el software debería construirse a partir de componentes prefabricados
  - texto completo `http://cm.bell-labs.com/cm/cs/who/doug/component$.txt`

Software de Comunicaciones

Definición de componente (1/5)

- Una unidad de composición con interfaces especificadas de manera contractual y dependencias de contexto completamente explícitas. Un componente software puede desplegarse independientemente y es sujeto de composición por parte de terceros.

  Clemens Szyperski
  Component Software: Beyond Object-Oriented Programming

Software de Comunicaciones

Definición de componente (2/5)

- Los componentes software posibilitan el reuso práctico de partes de software y la amortización de inversiones a lo largo de múltiples aplicaciones. Existen otras unidades de reuso, tales como bibliotecas de código fuente, diseños o arquitecturas. Por tanto, para ser específico, los componentes software son unidades binarias de producción, adquisición y despliegue independientes, que al interactuar, forman un sistema en funcionamiento.

  Clemens Szyperski
  Component Software : Beyond Object-Oriented Programming

Software de Comunicaciones

Definición de componente (3/5)

- … representa una parte modular de un sistema que encapsula sus contenidos y cuya manifestación es reemplazable dentro de su entorno. Un componente define su comportamiento en términos de interfaces ofrecidas y requeridas. Como tal, un componente sirve como tipo cuya conformidad se define mediante esas intefaces ofrecidas y requeridas (abarcando su semántica tanto estática como dinámica)… Un componente se modela a través de todo el ciclo de vida del desarrollo y se refina succesivamente hasta el despliegue y tiempo de ejecución

OMG
UML superstructure version 2

Definición de componente (4/5)

- ¿Qué es un componente?: Un módulo software que:
  - está orientado al desarrollo de aplicaciones por ensamblaje de módulos existentes
  - facilita la división del trabajo (responsabilidades claras)
  - constituye una unidad de reuso: se puede escoger "de la estantería", listo para su empleo (COTS: *Components "Off-The-Shelf"*)
  - constituye una unidad de despliegue: se compila y despliega de manera independiente

Definición de componente (5/5)

- Podemos requerir también de un componente que:
  - forme parte de una aplicación distribuida
  - sea de uso flexible ⇒ múltiples interfaces
  - pueda comunicarse de manera flexible
    - comunicación síncrona por invocación de métodos
    - comunicación asíncrona por canales (eventos)

Software de Comunicaciones

Modelos de componentes

- ¿Qué incluye un modelo de componentes?
  - una noción de componente individual
  - una definición de cómo ensamblar componentes
    - conectando interfaces compatibles
  - una definición de un entorno de componentes
    - es decir, un entorno de despliegue y ejecución de componentes
- ¿Qué proporciona un modelo de componentes?
  - diseño y desarrollo por ensamblaje: reuso
  - visión clara de la arquitectura de una aplicación
  - separación de los aspectos funcionales y no funcionales
- No cumplen las condiciones anteriores:
  - modelos basados en objetos, p.e. RMI,...
  - modelos basados en servicios, p.e. CORBA 2, Jini,...

Software de Comunicaciones

Entornos de componentes distribuidos (1/2)

- ¿Qué es un entorno de componentes distribuidos?
  - un entorno concebido para el despliegue y ejecución de aplicaciones distribuidas basadas en componentes

Software de Comunicaciones

Entornos de componentes distribuidos (2/2)

- ¿Qué conlleva un entorno de componentes distribuidos?
  - la separación de los aspectos funcionales y no funcionales
  - la gestión y soporte *implícitos* por el entorno de ejecución (vía perfiles estándar) de los aspectos no funcionales tales como:
    - seguridad (autenticación, autorización,...)
    - transacciones (definición declarativa o vía API)
    - control de concurrencia
    - persistencia (gestionada por el entorno o vía API)
    - gestión de ciclo de vida
    - nombrado (*naming*), *trading*, búsqueda de componentes
    - activación / desactivación
    - protocolos de comunicación
    - administración de componentes
  - soporte para el despliegue

Software de Comunicaciones

Desarrollo basado en componentes (CBD)

- Actualmente, pretenden realizar CBD:
  - Enterprise JavaBeans y el modelo de componentes de Java EE
  - la estructura de componentes de .NET
  - CORBA 3 y el modelo de componentes de CORBA (CCM)
- Pero ninguna satisface del todo las definiciones
  - Szyperski: “con… dependencias de contexto completamente explícitas”
  - UML 2.0: “especifica un contrato formal de los servicios… que requiere de otros componentes”
- Arquitectura de componentes / marco de componentes
  - sinónimos del término modelo de componentes (normalmente)

Software de Comunicaciones

Componentes vs. objetos

- Tienen en común:
  - modularidad (bajo acoplamiento externo y alta cohesión interna)
  - encapsulación/ocultación de información, abstracción,...
- Pero los componentes:
  - tienen granularidad menos fina
  - son elementos más asociados a la aplicación
    - no tienen por qué corresponderse con elementos del mundo real
  - hacen explícitas sus dependencias del contexto
    - implementaciones actuales: no todas las dependencias
  - interactúan vía protocolos de interacción bien definidos
  - unidad de despliegue: pueden desplegarse de manera independiente
  - unidad de re-uso (¿y las bibliotecas de clases, no lo son?) en todo caso, deberían conducir a un reuso mucho mayor

Software de Comunicaciones

Aplicaciones empresariales: requisitos (1/2)

- ***Almacenamiento y acceso de datos (*Back-end integration*):*** empleo de sistemas de bases de datos (DBMS), conexión a la base de datos, representación de los datos en la base de datos
- ***Mapping de datos y persistencia*:** representación de los datos en los programas (clases) y correspondencia (mapping) con su representación en la base de datos, actualización de la base de datos tras cambios por el programa
- ***Consistencia de datos*:** control de acceso concurrente a los datos, monitores de transacción
- ***Interacción con el usuario*:** autentificación, control de acceso, coordinación de accesos concurrentes
- ***Acceso a datos comunes*:** aislamiento de los distintos accesos, caché de datos

Software de Comunicaciones

Aplicaciones empresariales: requisitos (2/2)

- ***Prestaciones*:** tiempo de respuesta, interacción eficiente entre los distintos componentes
- ***Escalabilidad*:** posibilidad de incorporar nuevos servidores, distribución de carga
- ***Disponibilidad*:** seguridad frente a caídas de la aplicación (ideal disponibilidad 24 x 7), sistemas de tolerancia a fallos, *clustering* de servidores y datos
- ***Diseño software*:** mantenibilidad y portabilidad → modularidad, diseño en niveles, reducción de dependencias externas (por ejemplo, en la base de datos)
- ***Independencia de plataforma?***

Software de Comunicaciones

Contenidos

- Introducción
  - desarrollo basado en componentes
  - aplicaciones y arquitecturas empresariales
- Arquitecturas cliente-servidor y multi-nivel
  - conceptos básicos
  - nivel de cliente
  - nivel de presentación
  - nivel de negocio
  - nivel de datos
- La plataforma Java Enterprise Edition
  - introducción
  - la arquitectura y los contenedores de Java EE
  - los componentes de Java EE
  - los servicios de Java EE

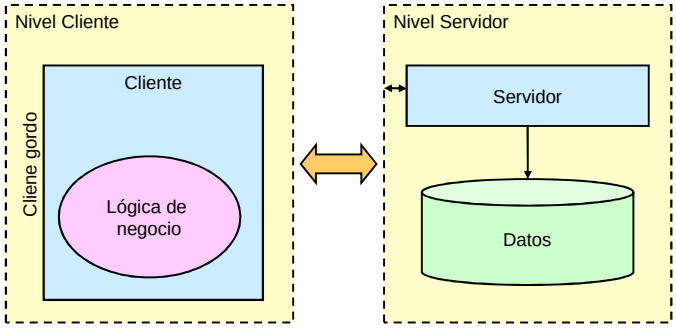
Software de Comunicaciones

Arquitectura cliente/servidor clásica

- Reparto funcional y físico (subsistemas) de la aplicación en dos niveles: cliente y servidor
  - los datos de negocio – en una base de datos o, en general, en el *Enterprise Information System* (EIS) – residen en el servidor
- Dos tipos principales de arquitectura según donde residen la lógica de ejecución, la preparación y presentación de información, la interacción con el usuario etc.
  - **cliente gordo:** parte principal de la aplicación se ejecuta en el cliente
  - **cliente delgado:** parte principal de la aplicación se ejecuta en el servidor
- Cliente y servidor están débilmente acoplados y se comunican sólamente mediante mensajes
- Papel del servidor es pasivo: comunicación iniciada por el cliente (con una solicitud de servicio); servidor responde

Software de Comunicaciones

Arquitectura cliente/servidor: cliente gordo



Software de Comunicaciones

19

Ventajas de los clientes delgados

- Menos infraestructura en el lado cliente
  - reduce costes puesto que hay muchos clientes, pocos servidores
- Administración más fácil
  - es decir, configuración, mantenimiento, despliegue,…
  - puesto que hay menos servidores que clientes
- Menos tráfico en la red
  - debido a un nivel de servicio más abstracto ofrecido al cliente
- Gestión de recursos centralizado
  - ayuda a asegurar la integridad de los datos
  - mayor nivel de seguridad
  - mejor detección de fallos
  - más control sobre las transacciones
  - …
- Más evolutivo, p.e. frente a un cambio del SGBD

Software de Comunicaciones

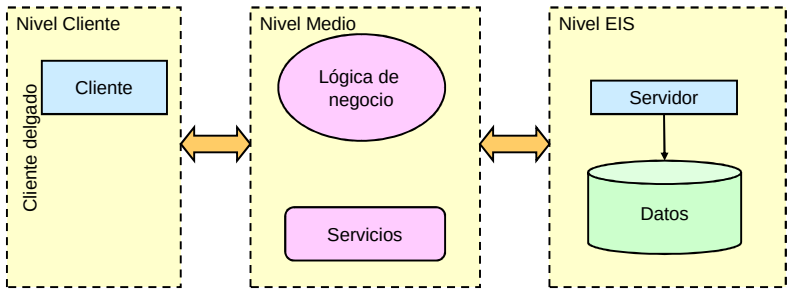
20

Arquitecturas *multi-tier* (multi-nivel) (1/2)

- En las arquitecturas multi-tier, se añaden niveles (*tiers*) de software adicionales que se encargan de realizar ciertas tareas críticas
  - los clientes son clientes delgados (*thin clients*)
- Los niveles intermedios extienden la responsabilidad del lado del servidor
  - aunque pueden estar situados en ordenadores o sistemas independientes
- Cada nivel se comunica
  - sólo con los niveles contiguos
  - a través de interfaces claramente definidas

Software de Comunicaciones

Arquitecturas *multi-tier* (multi-nivel) (2/2)



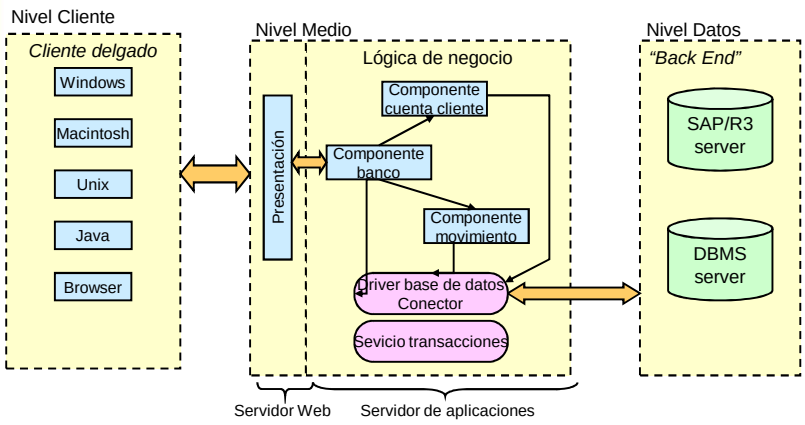
Software de Comunicaciones

Ventajas de las arquitecturas multi-nivel

- Todas las ventajas del cliente delgado
- Más flexibilidad y escalabilidad
- Niveles pueden actualizarse / remplazarse independientemente
  - con cambios de requisitos o de tecnología
- Un control más fino de la carga del servidor
  - evitar sobrecarga del servidor
  - equilibrar la carga entre servidores
  - conseguir tiempo de respuesta más bajo (en general)
- Más facilidad para depurar errores
  - debido a una mayor modularidad

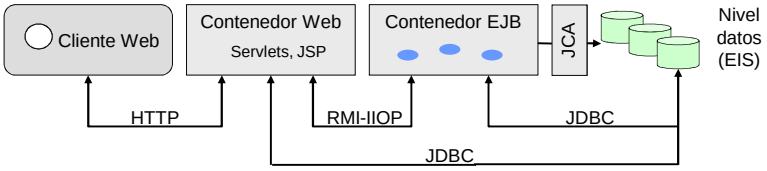
Software de Comunicaciones

Ejemplo: arquitectura multi-nivel



Software de Comunicaciones

Arquitectura de aplicación Web Java EE



Software de Comunicaciones

Servidor de aplicaciones (+ contenedor)

- Sistema de soporte para componentes de servidor
  - entorno de desarrollo para los componentes
  - los componentes de servidor utilizan los servicios del servidor de aplicaciones
- Tareas de infraestructura:
  - instanciación de componentes
  - comunicación
  - sincronización de accesos concurrentes
  - preparación de un entorno seguro
  - disponibilidad
  - seguridad de transacciones

Software de Comunicaciones

Nivel de cliente

- Es la parte de la aplicación que se ejecuta en la máquina del cliente
- Suele tener sólo las siguientes funciones:
  - mostrar la información del servidor
  - recoger datos de entrada
- Típicamente un navegador Web; puede incluir:
  - applets para mostrar información gráfica
  - javascript para pre-procesar entradas
  - plugins (como Flash)

o puede ser una aplicación Java

Software de Comunicaciones

Nivel medio

- Servidor de aplicaciones: parte principal de la aplicación
  - lógica de la aplicación y de negocio
  - preparación de la información para el usuario
- *Middleware*:
  - suele incluir software especializado para la realización de determinadas tareas (servicios corporativos estándares) :
    - monitores, sistemas de nombres, sistemas de colas de mensajes, etc.
    - e.g. CORBA y CORBA services

Software de Comunicaciones

Nivel medio: lógica de presentación

- Recibe las peticiones de los clientes
  - Extrae los datos del cliente
  - Extrae información adicional (cabeceras de la petición)
- Realiza un pre-procesado de la petición
  - Decide qué servicios del nivel de negocio son necesarios
  - Llama a dichos servicios
- Prepara las respuestas al cliente
  - Cabeceras de la respuesta
  - Contenido de la respuesta (típicamente HTML)

Software de Comunicaciones

Nivel medio: lógica de negocio

- Implementa la lógica de negocio
  - es decir, la funcionalidad propiamente dicha
- Servidor de aplicaciones
  - puede integrar también la parte de presentación
- El servidor de aplicaciones contiene:
  - un contenedor donde “viven” los componentes de negocio
    - componentes de sesión (representan procesos)
    - componentes de entidad (representan datos)
  - el resto de servicios *middleware*
    - procesado de datos basado en transacciones
    - acceso seguro
    - monitores
    - sistemas de nombres, etc.

Software de Comunicaciones

Nivel de datos

- Bases de datos o *Enterprise Information Systems*
- Responsable de la administración, acceso rápido a, y persistencia de, los datos
- Accedido por el nivel de negocio
  - mapeo entre la representación de datos en el nivel de negocio y en el de datos
- Nota: las aplicaciones web muy sencillas pueden no tener nivel de negocio
  - el nivel de presentación incluye la lógica de la aplicación
  - el nivel de presentación se comunica directamente con el nivel de datos

Contenidos

- Introducción
  - desarrollo basado en componentes
  - aplicaciones y arquitecturas empresariales
- Arquitecturas cliente-servidor y multi-tier
  - conceptos básicos
  - nivel de cliente
  - nivel de presentación
  - nivel de negocio
  - nivel de datos
- La plataforma Java Enterprise Edition
  - introducción
  - la arquitectura y los contenedores de Java EE
  - los componentes de Java EE
  - los servicios de Java EE

Platforma Java Enterprise Edition

- Colección de especificaciones y directivas de programación para facilitar el desarrollo de aplicaciones de servidor distribuidas multi-nivel, alineada fuertemente con Internet

Un poco de historia…

- 1996: Java Development Kit (JDK) 1.02: colección ordenada de bibliotecas de clases y paquetes
- 1999: JDK 1.2 → Java 2 Platform: adicional al JDK, paquetes opcionales para mensajes, generación dinámica de páginas Web o programas de email en Java. Dividida en 3 ediciones:
  - Java Standard Edition (Java SE): contiene el SDK actual y las APIs estándar; pensado para aplicaciones de desktop y applets
  - Java Enterprise Edition (Java EE): basada en Java SE, extiende el lado del servidor; v1.3 (finales de 2001), v1.4 (mediados de 2003), JEE5 (feb. 2006)
  - Java Micro Edition (Java ME): pensado para sistemas móviles y enmarcados: teléfonos móviles, *palmtops*, etc.

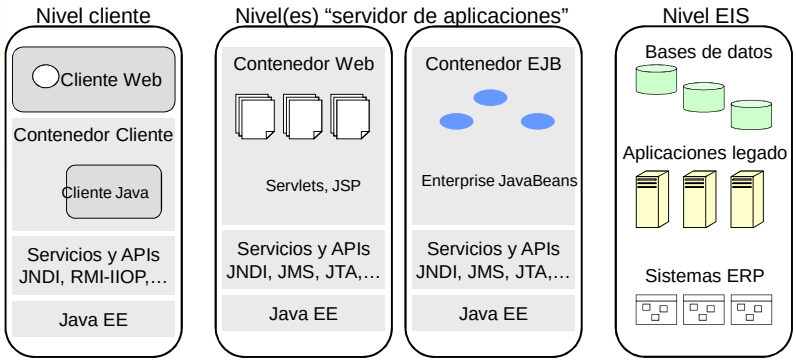
Software de Comunicaciones

Elementos de la especificación Java EE

- **Java EE Platform**: estándar representado por un conjunto de APIs y directivas, soportadas por un servidor de aplicación (`java.sun.com/javaee/`)
- **Java EE Server**: implementación de referencia de un servidor de aplicaciones para Java EE (`java.sun.com/javaee/`)
- **Java EE Testsuite**: Java EE Compatibility Testsuite (CTS), certificación de implementaciones de Java EE etc. (`java.sun.com/javaee/overview/compatibility.jsp`)
- **Java EE Blueprints**: consejos para el desarrollo de aplicaciones Java EE, patrones de diseño y un ejemplo de aplicación (`java.sun.com/reference/blueprints/index.html`)

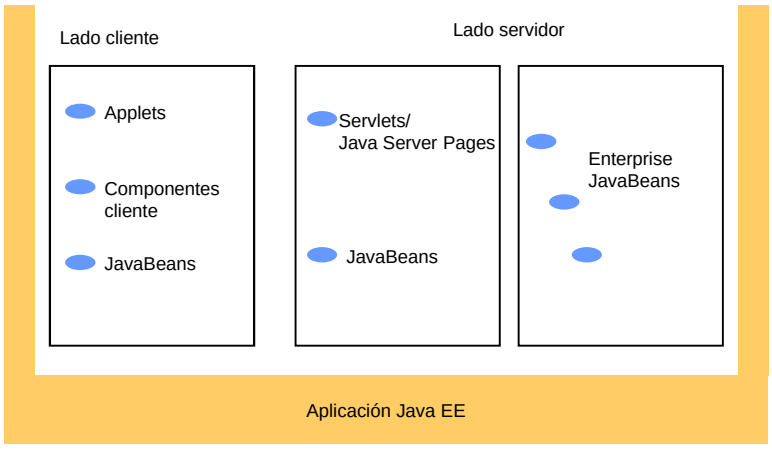
Software de Comunicaciones

Arquitectura de aplicación Java EE



Software de Comunicaciones

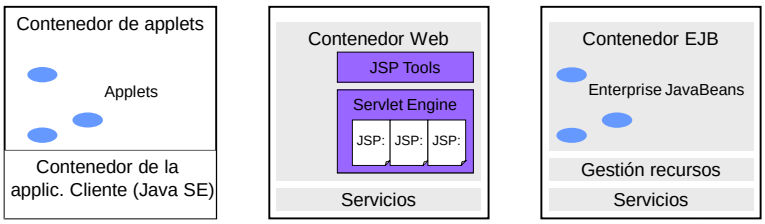
Tipos de componentes en Java EE



Software de Comunicaciones

Contenedores (*containers*)

- Ofrecen un entorno de ejecución para los compts. de la aplicación
- Proporcionan una vista uniforme de los servicios requeridos por ellos
  - tal como especificados en los descriptores (espec. de dependencias)
- Proporcionan herramientas de despliegue
  - para la instalación y configuración de componentes (también en tiempo de ejecución)
- Las tareas principales de los contenedores del lado del servidor:
  - la gestión de recursos y del ciclo de vida



Software de Comunicaciones

37

Servlets y JavaServer Pages

- Servlets y JavaServer Pages (JSP) son los componentes del nivel de presentación
- Permiten generar páginas web dinámicas
- Servlets:
  - Java code: más fácil controlar el flujo de acciones
- JSP:
  - lenguaje de marcado basado en etiquetas: más fácil representar información
- Equivalentes (una JSP se traduce a un servlet)

Software de Comunicaciones

38

Enterprise JavaBeans

- 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 gracias al principio: *Write Once Run Anywhere* (WORA) y a su realización en Java
  - asegurar la compatibilidad con Java-APIs existentes, con sistemas de servidor de terceros y con protocolos de CORBA y de *servicios Web*

Software de Comunicaciones

¿Qué son los Enterprise JavaBeans?

- Véase la definición de un modelo de componentes (página 9)
- Una noción de componente individual. Los EJBs son
  - componentes usados como parte de aplicaciones corporativas distribuidas
  - partes encapsuladas parte de la lógica de negocio de la aplicación
- Una definición de cómo ensamblar componentes
  - se comunica con gestores de recursos, y con otros EJBs
  - accedido por distintos tipos de clientes: EJBs, servlets, clientes de aplicación, etc.
- Entorno de ejecución (el contenedor EJB)
  - el contenedor EJB proporciona el aceso a servicios para seguridad, transacciones, despliegue (*deployment*), control de concurrencia, gestión del ciclo de vida
  - una aplicación puede tener uno o varios EJBs en uno o varios contenedores EJB

Software de Comunicaciones

Tipos de EJBs

- De entidad (*Entity Beans*):
  - modelan conceptos de negocio como objetos persistentes asociados a datos. Ej. Cuenta bancaria, producto, pedido…
- De Session (*Session Beans*):
  - representan procesos ejecutados en respuesta a una solicitud del cliente. Ej. Transacciones bancarias, cálculos, realización de pedidos…
- Activados por mensaje (*Message-Driven Beans*):
  - representan procesos ejecutados como respuesta a la recepción de un mensaje

Software de Comunicaciones

Servicios Java EE (1/2)

- **Servicio de nombres**: acceso a componentes y recursos mediante nombres lógicos
  - portabilidad y mantenibilidad
  - *Java Naming and Directory Interface (JNDI)*
- **Servicio de transacciones**: ejecución de una serie de pasos de forma atómica y aislada
  - *demarcación de transacciones declarativa*: concepto declarativo de límite de transacción mediante descriptores
  - *demarcación de transacciones programática*: posibilidad de un control de las transacciones más fino mediante una interfaz de programación
  - *Java Transaction Service (JTS)*
- **Servicio de seguridad**: directivas de seguridad para recursos protegidos
  - control de acceso en dos pasos: autentificación y autorización
  - realización declarativa o programada
  - *Java Athentication & Authorization Service (JAAS)*

Software de Comunicaciones

Servicios Java EE (2/2)

- **Persistencia**: almacenamiento persistente de objetos y estados de objetos, normalmente realizado en bases de datos relacionales
  - declarativa: persistencia gestionada por el contenedor (CMP)
  - programática: persistencia gestionada por *bean* (BMP) via API JDBC
- **Comunicación**: distintas técnicas de comunicación, proporcionadas por el proveedor de servicio de aplicación o de contenedores
  - comunicaciones Web: TCP/IP, UDP/IP, HTTP y HTTPS (con SSL/TLS)
  - procesamiento de objetos distribuidos, RMI (Remote Method Invocation)
    - Java Remote Method Protocol (JRMP)
    - CORBA-IIOP para interoperabilidad entre Java EE y sistemas CORBA
    - JAX-RPC para interoperabilidad entre Java EE y Web Services
- **Servicios de configuración y administración**: empaquetamiento, instalación y configuración flexible de componentes y la administración de aplicaciones
  - descripción mediante esquemas XML de las características de servidores, contenedores, aplicaciones, componentes y servicios.

Software de Comunicaciones