TEMA 8

CARACTERISTICAS DE LOS MECANISMOS DE COMUNICACION

Permiten la comunicación entre procesos ejecutandose en ordenadores distintos. Son ofrecidos por los middlewares de comunicación. Caracteristicas:

UTILIZACIÓN

A Con primitivos básicos de comunicación
- *Operaciones de envio y recepción.
- *Ejemplos: Sockets, colas de mensajes

B Mediante construcciones de lenguajes de programación
- * Mayor nivel de abstracción
- * Envio/recepción de mensajes transparente al programador
- *Ejemplos: (RPC) remote procedure call, (ROI) remote object invocation

ESTRUCTURA DE LOS MENSAJES

A No estructurados solo contenido: contenido en formato libre. Ej: sockets

B Estructurados en cabecera + contenido: cabecera de un conjunto de campos estandar y el contenido de formato libre. Ejemplo: colas de mensajes

C Estructura transparente al programador: determinada por el middleware de comunicaciones. Ejemplo: RCP, ROI

CONTENIDO DE LOS MENSAJES

A Bytes:
- *Ventajas: eficiente y compacto
- *desventajas: difícil de procesar, la representación en binario no es igual entre arquitecturas y lenguajes de prog

B Texto:
- *ventajas: independiente de la arquitectura y lang de programación
- *desventajas: menos eficiente
- * Se suele usar un lenguaje con amplia disponibilidad de bibliotecas para su procesamiento (XML, JSON)

cambiar con Typex

DIRECCIONAMIENTO

A Directo: el ordenador emisor envía los mensajes directamente al receptor. Ej: Socket web, RPC, ROI

B Direccionamiento indirecto: los mensajes se envían a través de un intermediario (Broker)
Ejemplo: Colas de mensajes

SINCRONIZACIÓN

A Comunicación asincrónica: el middleware responde al emisor con la confirmación, tras almacenarlo en sus buffers (la + rápida, ni siquiera espera a q llegue a receptor)

B Comunicación sincrónica (entrega): El middleware responde cuando el receptor ha confirmado la entrega correcta del mensaje (tarda una vez de llegar, antes de procesarlo)

C Comunicación sincrónica (respuesta): El middleware responde al emisor tras recibir aviso del receptor de haber procesado el mensaje. (tarda + en responder)

PERSISTENCIA

A Comunicación persistente: el middleware puede guardar los mensajes perdidos de entrega. Ejemplo: colas de mensajes (hace de buzón)

B Comunicación no persistente: El middleware no es capaz de ~~transmitir~~ mantener los msjs que deben transmitirse. Ejemplo: sockets, servicios web, RPC/ROI

Estas características son útiles para categorizar y comprender los diferentes mecanismos de comunicación existentes, pero la realidad es compleja y algunos mecanismos no encajan exactamente

INVOCACIÓN A OBJETO REMOTO (ROI)

Objeto remoto → Un objeto que puede ser invocado desde otros espacios de direcciones. Este se instancia en un servidor y puede responder a la invocación de objetos remotos y locales

Los aplicaciones se organizan en colecciones dinámicas de objetos que pueden estar en diferentes nodos. Ellos y los objetos pueden invocar métodos de otros objetos en remoto
Asumiremos que todo objeto existe modelos está completamente contenido en un único nodo, aunque existen modelos de objetos distribuidos tmb.

VENTAJAS DEL MODELO DE OBJETOS DISTRIBUIDOS

- Aprovecha la expresividad, capacidad de abstracción y flexibilidad del paradigma OO

— Transparencia de ubicación
  * La sintaxis de invocación de los métodos de un objeto no depende del espacio de direcciones en el que reside dicho objeto
  * Podemos ubicar los objetos de acuerdo a diferentes criterios: seguridad, restricciones, ...
— Podemos reutilizar aplicaciones "heredadas" (legacy) encapsulandolas en objetos usando el patrón de diseño "wrapper"
— Escalabilidad: los objetos pueden distribuirse sobre una red, teniendo en cuenta la demanda actual

TIPOS DE INVOCACIONES

— Invocación local: objetos invocador e invocado residen en el mismo proceso
— Invocación remotas (ROI): los objetos invocador e invocado residen en procesos distintos o en nodos distintos

ANTECEDENTES: RPC

— No contempla el concepto de objeto
— El ordenador remoto ofrece un catálogo de procedimientos que pueden ser llamados de forma transparente (como si fueran locales) desde ordenadores cliente. Mecanismo implementado por los stubs: stub cliente y stub servidor

3ero 3/5 A:23

PASOS

1° Invocación del procedimiento local
2° Empaquetado de argumentos de entrada (marshalling) en un mensaje
3° Envío del mensaje al servidor y espera de respuesta
4° Desempaquetado y extracción de argumentos
5° Llamada al procedimiento
6° Ejecución del procedimiento
7° Retorno de control al stub servidor
8° Empaquetado de argumentos de salida y resultado en un mensaje
9° Envío de mensaje de respuesta
10° desempaquetado del mensaje y extracción de respuesta + argumentos
11° Retorno de control al código que invocó el código local

INVOCACIÓN A OBJETO REMOTO (ROI)

Enmascara la invocación a objeto remoto como invocación local.
Similar a RPC (proporciona transparencia de ubicación) pero aplicado al modelo de objetos (paradigma OO)

— Aplicación = colección de objetos repartidos en distintos nodos
— Podemos invocar metodos de otros objetos de forma remota

ELEMENTOS QUE INTERVIENEN EN UNA ROI

— Proxy
  * Ofrece la misma interfaz que el objeto remoto
  * Contiene una referencia al objeto remoto
    - Proporciona acceso al objeto remoto y a su interfaz
  * Se crea en tiempo de ejecución cuando se accede por 1ª vez al obj remoto

— Esqueleto
  * Recibe las peticiones de los clientes
  * Realiza los verdaderos llamados a los metodos del objeto remoto
  * Se crea en tiempo de ejecución cuando se crea el objeto remoto

— OBject Request Broker (ORB)
  * Componente principal de un middleware orientado a objetos
  * Se encarga de:
    - Identificar y localizar los objetos
    - Realizar las invocaciones remotas de los proxies a los ~~objetos~~ esqueletos
    - Gestionar el ciclo de vida de los objetos (creación, registro, activación y eliminación)

PASOS DE UNA ROI

1° El proceso cliente invoca el metodo del proxy local relacionado con el objeto remoto
2° El proxy empaqueta los argumentos y usando la referencia al objeto, llama al ORB. El ORB gestiona la invocación, haciendo que el mensaje llegue al esqueleto
3° El esqueleto desempaqueta los argumentos e invoca el metodo solicitado, quedando a la espera de que finalice
4° El método llamado finaliza y se desbloquea el ~~objeto~~ esqueleto

5º El esqueleto empaqueta los resultados y llama al ORB, el que hace llegar el mensaje al proxy
6º El proxy desempaqueta los resultados y los devuelve al proceso cliente ~~hace llegar el objeto al proxy~~

PASO DE OBJETOS COMO ARGUMENTOS

PASO POR REFERENCIA

- Basta con copiar la referencia del nodo invocador al nodo invocado.
- No importa que el objeto pertenezca al nodo invocador o que pertenezca a un tercer nodo

PASO POR VALOR

— El estado del objeto origen se empaqueta mediante un proceso denominado serialización
— El objeto serializado se transmite al nodo destino, donde a partir de dicha información se crea un nuevo objeto copia del original
— Ambos objetos evolucionan por separado

CREACIÓN DE OBJETOS

En ROI la creación de objetos (y su registro en el ORB) puede realizarse mediante 2 procedimientos distintos:

— Por iniciativa del cliente
  * El cliente solicita a una factoría crear el objeto
  * Una factoría es un objeto servidor que crea objetos de un tipo, los registra en el ORB y devuelve una copia de la referencia al cliente
— Por iniciativa del servidor
  * Un proceso servidor crea un objeto y lo registra en el ORB, obteniendo una referencia al objeto
  * El proceso servidor registra la referencia en el servidor de nombres, para que otros clientes puedan buscarla, mediante el nombre lógico (cadena de texto) proporcionado por el s. de nombres.
  * Cualquier proceso que conozca el nombre lógico del objeto puede contactar con el servidor de nombres y obtener una referencia a dicho objeto

JAVA RMI

Es un middleware de comunicación orientado a objetos que proporciona una solución para un lenguaje OO específico (Java) que da soporte a la portabilidad

- No es multilenguaje, pero sí multi-plataforma
- Permite invocar métodos de objetos java de otra JVM, y pasar objetos Java como argumentos cuando se invocan dichos métodos

El componente RMI se incorpora de forma automática a un proceso Java cuando se usa su API (escucha peticiones que llegan en un puerto TCP)

Un objeto es invocable de forma remota si implementa una interfaz que extiende la interfaz Remote.
Para cada uno de estos objetos remotos, RMI crea dinamicamente un objeto llamado esqueleto (no accesible por el usuario)
Para invocar un objeto remoto desde otro proceso se usa un proxy. Su interfaz es idéntica a la de un objeto remoto

* Dirección IP + puerto + qué objeto
* Permite localizar al esqueleto del objeto remoto

EL SERVIDOR DE NOMBRES DE JAVA RMI

- Almacena para cada objeto: nombre simbólico + referencia
- Puede residir en cualquier nodo y es accesible desde el cliente o el servidor usando la interfaz local llamada Registry
- En las distribuciones Oracle de Java el servidor de nombres se lanza usando la orden rmiregistry

Soto 4/5 10:50

DESARROLLO DE UNA APLICACIÓN JAVA RMI → REGLAS PARA OBJS REMOTOS

- Interfaz de objeto Remoto
  * Debe extender la interfaz java.rmi.Remote (extends Remote)
  * Los métodos de dicha interfaz deben indicar que puede generarse la excepción RemoteException
  * A partir de la definición de la interfaz, el compilador de java genera proxies y esqueletos
- Clase de objetos remotos:
  * Debe implementar la interfaz remota
  * Debe extender java.rmi.server.UnicastRemoteObject

Esto permite la registrar los objetos en el ORB de Java

Soto 4/5 10:52

PASO DE OBJETOS COMO ARGUMENTOS EN JAVA RMI

— Cuando se invoca un método podemos pasar objetos como argumentos. Si el objeto que se pasa como argumento implementa la interfaz Remote entonces se pasa por referencia y el objeto es compartido por las referencias previas y nuevas. Sin embargo, si no implementa Remote: se serializa y pasa por valor y se crea un objeto en la maquina virtual destino totalmente independiente al original (3.b 4/5 11:12)

CARACTERISTICAS DE LA COMUNICACIÓN EN JAVA RMI

— Utilización: los llamados a métodos remotos hacen transparente al programador el uso de los primitivos basicos de la comunicación
— Estructura y contenido de los mensajes: Determinados por el compilador de Java, transparente al programador
— Direccionamiento: Directo al ordenador donde reside el objeto remoto
— Sincronización: Sincrónica en la respuesta. Se espera a que el método termine
— Persistencia: No persistente. El objeto remoto debe estar activo

SERVICIOS WEB

— Sistema Software diseñado para proporcionar interacciones ordenador-ordenador usando la red
— Normalmente basado en una arquitectura cliente-servidor (petición respuesta) implementada sobre el protocolo HTTP
— No se solicitan paginas web, sino consultas y acciones

Existen numerosos variables. Los más representativos son:

— Servicios Web basados en SOAP y WSDL
  * Conocidos como servicios web
  * SOAP (Simple Object Access Protocol): Especificación XML de la información que se intercambia
  * WSDL (Web Services Description Language): Especificación XML que describe la funcionalidad ofrecida por un servicio web

- Servicios web RESTful
  * Alternativa más simple y flexible, lo que ha hecho que tengan gran aceptación en la actualidad desbancando a los servicios clásicos
  * JSON es el formato más habitual para el intercambio de información
  * No se requiere ningún lenguaje de descripción de funcionalidad. No obstante se están empezando a usar siendo OAS (Open Api especification) el más usado.

Alternativas para utilizar servicios web
- Construcción y procesado directo del contenido de los mensajes HTTP
- Generación automática de código proporcionada por frameworks de desarrollo y despliegue de servicios web, tanto para el lado cliente como el lado servidor a partir de la definición del servicio. (Los servicios web podrían considerarse como una forma de RPC.

¿QUÉ ES UN SERVICIO WEB RESTFUL?
- Constituyen una de las tecnologías más importantes para el desarrollo de aplicaciones web.
- Están disponibles en la mayoría de lenguajes de programación y frameworks de desarrollo.
- Gran parte de su éxito se debe a su sencillez
- Estrictamente no es un estándar, sino un conjunto de convenciones de uso
  * Datos y funcionalidad se consideran recursos → se accede a ellos mediante URLs
  * Se actúa sobre recursos utilizando un conjunto de operaciones simples y bien definidos (basados en HTTP)
  * Se emplea la arquitectura cliente/servidor
  * Diseñado para utilizar un protocolo de comunicación sin estado (que usa caché)

REFERENCIAS A RECURSOS EN REST

— Identificación de recursos a través de la URI: proporcionan un espacio de dirección global para el descubrimiento de recursos y servicios

Fotos (4/5 12:44)

OPERACIONES EN REST

GET → Leer un recurso

POST → Crear un recurso

PUT → Modificar un recurso

Delete → Borrar un recurso

CODIGOS DE ESTADO EN REST

200 → OK

201 → Recurso creado

400 → Solicitud incorrecta

404 → Recurso no encontrado

500 → fallo en el proveedor de servicio

Gerph: Google Drive Web Api (mirar págs 65-67)

CARACTERISTICAS DE LA COMUNICACIÓN EN REST pág