Taller práctico sobre CORBA en GNOME

Rodrigo Moya

<rodrigo@gnome-db.org>

Este documento constituye la propuesta para un taller (2 o más horas) sobre el uso de CORBA en GNOME, de forma que pueda servir para ver lo sencillo que es usar esta potente tecnología en aplicaciones desarrolladas para el proyecto GNOME. Dicho taller sería impartido por Rodrigo Moya y Diego Gonzalez.

Tabla de contenidos

1. Taller práctico sobre CORBA en GNOME

CORBA es una tecnología con una extremada potencia, que permite el desarrollo de aplicaciones distribuidas. Sin embargo, si bien potente, el acceso a ella es, cuando menos, algo dificultoso en el inicio, debido a la gran variedad de términos y conceptos con que debe enfrentarse quien se acerca a CORBA por primera vez.

Debido a estas dificultades, mucha gente sufre de pánico ante la idea de usar CORBA en sus aplicaciones. Por esta razón, en el proyecto GNOME siempre se ha buscado el hacer lo más sencillo posible el uso de esta potentísima tecnología, en la que se basa una buena parte del proyecto GNOME, y para ello, se han creado nuevas interfaces que hacen el acceso a la tecnología de CORBA un juego de niños. En este taller, vamos a intentar quitar los miedos a usar CORBA, mediante el uso de ejemplos prácticos de uso de dicha tecnología en aplicaciones GNOME.

1.1. Teoría sobre CORBA

La teoría de CORBA está repleta de nuevos términos que, en muchas ocasiones, no hacen sino crear confusión. Pero, una vez analizada dicha teoría, se encuentran multitud de parecidos con la terminología habitual de la programación orientada a objetos.

1.1.1. ORB, el núcleo

El ORB (Object Request Broker) es la parte del estándar CORBA que se encarga la implementación, tanto de las comunicaciones, como de los mapeos a los distintos lenguajes de programación soportados. Es la parte básica de la arquitectura CORBA, y es totalmente imprescindible.

1.1.2. IIOP/GIOP

CORBA define, como protocolo de comunicaciones entre los distintos objetos, el GIOP, que especifica el formato de las comunicaciones CORBA sin especificar ningún transporte específico. Para eso se define IIOP, que no es más que GIOP sobre TCP/IP.

Estos protocolos son unos protocolos de comunicaciones muy eficientes, pensados para aligerar las comunicaciones lo más posible. En él, simplemente se define el formato de los mensajes, y luego, cada ORB (implementación de CORBA) es responsable de convertir dichos mensajes en datos para la máquina en la que se está ejecutando, permitiendo así la interacción entre distintas implementaciones de CORBA ejecutándose en máquinas distintas, con sistemas operativos distintos, e, incluso, con representación de datos distinta.

1.1.3. El lenguaje IDL

A la hora de desarrollar aplicaciones CORBA, uno de los primeros pasos que es escribir los interfaces IDL de la aplicación. IDL es un lenguaje que permite definir una serie de interfaces para la comunicación entre dos o más aplicaciones. El IDL se escribe en un fichero, normalmente con la extensión IDL, en el cual se definen los interfaces y los métodos (funciones) de esos interfaces.

Un fichero IDL podría ser el equivalente a un fichero de cabecera en C++, en el que se definen las clases (el equivalente a los interfaces CORBA) y las propiedades (variables de la clase) y métodos de esas clases. De hecho, IDL NO ES un lenguaje para implementar interfaces CORBA, sino que es un lenguaje PARA DEFINIR interfaces CORBA, por ello la equivalencia con un fichero de cabecera de C++, donde, normalmente, no se incluye ninguna implementación, sino simplemente definiciones.

1.1.4. Cabos y esqueletos

Una vez creado el fichero IDL describiendo los interfaces de los objetos, se usa lo que se llama un compilador IDL, que genera los cabos y esqueletos necesarios para el uso e implementación de dichos interfaces. Los cabos son el

código generado para ser usado desde la parte cliente (la aplicación que va a hacer uso del objeto CORBA), mientras que los esqueletos son, valga la redudancia, el esqueleto de la implementación del objeto.

Existen compiladores IDL para distintos lenguajes de programación, por lo que, una vez escrito el fichero IDL, se pueden generar los cabos y esqueletos, y por tanto implementar cada una de las partes, en cualquier lenguaje de programación que soporte CORBA, ayudando de esta forma a la transparencia del lenguaje de implementación de cada una de las partes que componen la aplicación.

Existe tambien la posibilidad de saltarse el paso de compilación del fichero IDL, y generar esos cabos y esqueletos en tiempo de ejecución, a partir de dicho fichero IDL. Esto es lo que hace orbit-perl, por ejemplo, que permite el uso de ORBitSección 1.2 desde el lenguaje Perl.

1.2. ORBit, el ORB de GNOME

Como base de todos los desarrollados relacionados con CORBA en GNOME, tenemos a ORBit, el ORB (Object Request Broker) desarrollado por el equipo de programadores de GNOME, en el año 1997, ante la falta de implementaciones de CORBA existentes por aquel entonces que cumplieran los tres requisitos que eran precisos para poder ser usados en GNOME: libre, rápido y ligero. Con el paso del tiempo, ORBit ha demostrado ser una de las mejores implementaciones de CORBA existentes hasta la fecha, no ya dentro de las libres, si no incluyendo las implementaciones no libres tambien. Es especialmente digna de destacar la velocidad y ligereza de ORBit, que gana con diferencia al resto de implementaciones en cuanto a velocidad y rendimiento se refiere.

Si bien no es una implementación completa de CORBA (sólo implementa uno de los 14 servicios CORBA, aunque hay que destacar que ninguna implementación de CORBA los implementa todos), si es una implementación especialmente indicada para cubrir las necesidades de GNOME Y de muchas otras aplicaciones que no precisen de un uso ultra-extensivo (uso de todos los servicios CORBA, por ejemplo, algo realmente raro) de CORBA. ORBit es capaz de comunicarse con otros ORB's que cumplan el estándar CORBA sin ningun problema.

ORBit está desarrollado en C, y por tanto es este el lenguaje mejor soportado e indicado para el desarrollo de aplicaciones con ORBit, aunque tambien existe soporte para otros lenguajes, como C++, Perl o Python.

1.3. Activación de componentes

Una de las lagunas del estándar CORBA es en lo referente a la activación de componentes bajo demanda, de forma que un cliente pueda solicitar la activación de un objeto CORBA sin necesidad de que el proceso que lo implementa tenga que estar ya en ejecución. Esto es una seria limitación para la que el proyecto GNOME desarrolló bonobo-activation. Bonobo-activation es un sistema para la activación de objetos bajo demanda, que funciona mediante la existencia de un demonio (uno por usuario) que lee la información de los componentes instalados para activar los objetos requeridos por los clientes.

Aparte de esto, incluye características extra, como por ejemplo el lenguaje de consulta (al estilo del SQL), que permite a los clientes obtener información sobre los componentes instalados en el sistema. Estas consultas se pueden hacer basadas en los interfaces implementados por los objetos, por ejemplo.

1.4. Bonobo, la joya de la corona

Como estrella de todos los desarrollos basados en CORBA en GNOME, tenemos a Bonobo, el sistema de componentes desarrollado por el equipo de programadores de GNOME.

Bonobo, como sistema de componentes, permite, de forma sencilla, lo siguiente:

- Implementación de objetos CORBA de una forma mucho más sencilla que mediante el uso directo del código generado por `orbit-idl`. Esto se consigue mediante el objeto BonoboObject.
- Implementación y uso de controles, que son el equivalente en GNOME de los Beans de Java o los ActiveX de Windows. Es decir, son controles de pantalla visuales que se exportan entre procesos, de forma que pueden ser reutilizados por otras aplicaciones de una forma sencilla. Esto se consigue mediante los objetos BonoboControl y BonoboWidget.
- Documentos compuestos, que son documentos cuyo contenido puede ser editado por varias aplicaciones a la vez, de forma que puedan insertarse los subdocumentos gestionados por una aplicación en un documento maestro gestionado por otra aplicación. Este sistema es la base de GNOME Office.
- Envío de datos entre procesos de forma sencilla, mediante el uso de BonoboStream's y BonoboListener's.
- Sistema de activación de objetos de alto nivel, conocido como 'monikers', y que permite una forma mucho más sencilla y potente de activar objetos que mediante el uso directo de bonobo-activation.

1.5. Referencias

- GNOME - http://www.gnome.org [http://www.gnome.org]
- Información para desarrolladores - http://developer.gnome.org [http://developer.gnome.org]
- OMG (Object Management Group) - http://www.omg.org [http://www.omg.org]
- ORBit - http://www.labs.redhat.com/orbit [http://www.labs.redhat.com/orbit]
- Recursos sobre ORBit - http://orbit-resource.sourceforge.net [http://orbit-resource.sourceforge.net]