

Fisterra Barnacle: Persistencia de Objetos en GObject

José María Casanova Crespo
jmcasanova@igalia.com

Alejandro García Castro
acastro@igalia.com

12 de abril de 2004

©2004, Igalia S.L. ¹

Resumen

Fisterra Barnacle² es una propuesta de persistencia en objetos basada en GObject, la ausencia de una capa **middleware** orientada a objetos de negocio en GNOME creó la necesidad dentro del equipo de Igalia de construir una capa de abstracción que ocultara las necesidades de persistencia de los objetos y el acceso a bases de datos.

El resultado de este trabajo llevó a una propuesta denominada **Fisterra Barnacle**, que implementa las operaciones básicas sobre objetos persistentes sin la intervención del programador para su implementación.

De esta forma las operaciones de almacenamiento, recuperación y transmisión a través de capas de transporte se realizaría automáticamente en base a la descripción de la clase del objeto persistente.

1. Introducción

La programación orientada objetos es el paradigma más aceptado hoy en día para el desarrollo de grandes aplicaciones. Las estructuras sobre las que se apoya, clases y objetos, necesitan ser almacenadas de forma persistente. Los sistemas de gestión de bases de datos relacionales, como herramientas para soportar datos persistentes, tienen carencias para el almacenamiento de objetos al no acomodarse estos a su estructura rígida de tablas y tuplas.

Las solución de persistencia más recurrente consisten en implementar *ad hoc* las transformaciones (*mapping*) de la estructura objetual en equivalentes persistentes que encajen con el modelo relacional.

En la actualidad los trabajos existentes proponen distintas recomendaciones para realizar estas transformaciones *ad hoc* entre clases y tablas relacionales. Esto es un problema ya que en aplicaciones de gestión las operaciones de acceso a datos forman su núcleo principal y su implementación supone una parte importante del tiempo de desarrollo.

Existen implementaciones que facilitan la descripción de la transformación pero ésta tiene que ser detalladamente especificada por el desarrollador, otra aproximación es la utilización de una base de datos orientadas a objeto, pero las soluciones libres están bastante alejadas de ser utilizables en entornos de producción.

2. Objetivos

El objetivo de este proyecto **Fisterra Barnacle** es obtener una solución adecuada que permita el **almacenamiento persistente automático de objetos**, de forma que no sea necesaria la etapa de transformación (*mapping*) ya que sería realizada directamente a través de un soporte genérico de persistencia de objetos. Esta transformación automática debería tener en cuenta las asociaciones existentes entre los objetos así como las agregaciones y composiciones.

La parte de experimentación sobre las distintas propuestas procedentes de este trabajo se realiza sobre la implementación objetual de GNOME, GObject, que dota al lenguaje C de un entorno completo de objetos basado en Software Libre.

Este proyecto se plantea los siguientes tareas para alcanzar sus objetivos:

- Recopilación y clasificación de las distintas propuestas existentes para el almacenamiento persistente de objetos. Estudio de implementaciones *ad hoc* y necesidades finales de los desarrolladores.
- Análisis y diseño de los nuevos planteamientos, estas tareas se refieren al trabajo de creación de las hipótesis de trabajo.
- Implementación de pruebas, comprobación de hipótesis, se probarán las características de las

¹Copyright Igalia, S.L. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

²Percebe en Castellano

hipótesis más interesantes para la comparación de resultados sobre distintos modelos de transformaciones y distintos SGBD ³.

- Documentación y difusión, este trabajo se refiere a la creación de información acerca del proyecto y la difusión de resultados del mismo.

El resultado final de este trabajo sería propuesta completa de almacenamiento y recuperación de objetos, con una definición de interfaces genéricos para su futura integración dentro de un lenguaje de programación.

Esto nos permitirá proponer en firme la realización de la implementación de esta tecnología a la comunidad de desarrolladores de GNOME. Esta solución incrementaría el rendimiento del trabajo de implementación de aplicaciones en estos entornos, la base del trabajo de Igalia.

3. Estado Actual

El estado actual del proyecto **Fisterra Barnacle** dota a la librería **fisterra-base** del proyecto **Fisterra 2** de unas funcionalidades básicas para la gestión de la persistencia de los objetos. El prototipo actualmente en funcionamiento tiene las siguientes características:

- Primera aproximación a la descripción del modelado de clases.
- Sistema de generación de código fuente.
- *Fisterra Barnacle Data Object*: son objetos básicos con operaciones get y set sobre atributos con tipos básicos.
- Métodos generados automáticamente para el almacenamiento y recuperación de **Fisterra Barnacle** utilizando **libgda**.
- *Mapping* automático de transporte de **Fisterra Barnacle** a través de CORBA basado en Orbit.
- Generación del esquema relacional en base a la descripción de los **Fisterra Barnacle** para PostgreSQL.
- Sistema de identificación única de los objetos.
- Gestión de versiones de objetos para control de concurrencia.

El modelo de persistencia actual está basado en código generado a partir de la definición del modelo de objetos definido mediante un fichero XML. Este refleja los nombres de las clases, de los atributos y las relaciones existentes entre las clases.

Una vez procesamos este fichero XML con el generador de código, obtenemos una estructura de compilación con los *Fisterra Barnacle Data Object*, los **Fisterra Barnacle** DataAccess Objects, un esquema de base de datos descrito en SQL y un *mapping* CORBA para los *Fisterra Barnacle Data Object*.

De esta forma el desarrollador ya tiene implementado los objetos base del negocio y solo tendrá que implementar contra ellos las operaciones complejas y cuando quiera que sus operaciones sean persistentes simplemente debe realizar la llamada al API de persistencia para que el objeto que se haga persistente. En este momento, se actualizará su número de versión si ya era persistente o se le asignará un nuevo código si nunca había sido almacenado.

Los objetos son únicos a través de la identificación de los objetos utilizando el código **barnacle**. Esta clave está formada por tres campos: identificador del *host* donde se instanció por primera vez el objeto, identificador de la clase del objeto y código de secuencia del objeto. Estas tres partes de la clave implementadas mediante un único entero de 64 bits, nos aporta las siguientes características:

- Dos objetos creados en distintas máquinas no tendrán nunca el mismo identificador, aunque coincida su secuencia de clave.
- El código **barnacle** identifica la clase a la que pertenece un objeto. De esta forma que podremos conocer la clase de un objeto que se tiene que recuperar en base su clave.

Todas las clases generadas heredan de **Fisterra Barnacle**, que implementa operaciones sobre el control de la clave. El control sobre los distintos atributos se basa en la utilización de **properties** estructuradas en una tabla *hash* con atributos en la clase **Fisterra Barnacle**. Esto nos permite tener incluir en la lógica de los *Fisterra Barnacle Data Object* el concepto de un atributo sin valor (NULL), muy útil en múltiples situaciones.

Basándonos en la misma característica de conocimiento de la clase de un *Fisterra Barnacle Data Object*, es posible generar un *mapping* CORBA de los objetos de forma que un *Fisterra Barnacle Data Object* se pueda convertir en un tipo de dato genérico **Fisterra Barnacle** CORBA, este se transmita y posteriormente se recupere de forma idéntica a como fue transmitido. Eliminando la tediosa tarea de hacer el *mapping* entre los tipos de datos.

³Sistema Gestor de Base de Datos

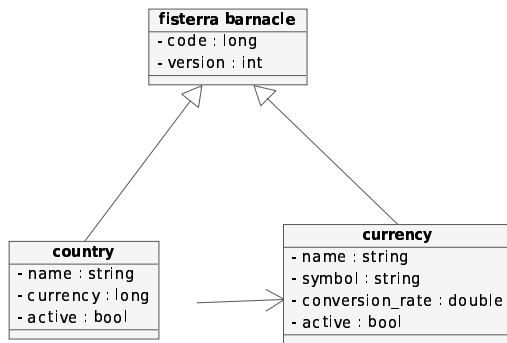


Figura 1: Ejemplo de Diagrama de Clases

4. Un ejemplo práctico

Para mostrar la utilidad de un sistema de persistencia la mejor forma es partir de un ejemplo, la figura 1 describe un diagrama de clases de un modelo objetual sencillo.

La transcripción al modelo en XML `prueba.xml` basándose en un DTD definido a tal efecto consistiría en la siguiente descripción:

```

<model>
  <class name="currency">
    <attribute name="name" type="varchar_n"/>
    <attribute name="symbol" type="varchar_n"/>
    <attribute name="conversion_rate" type="double"/>
    <attribute name="active" type="boolean" query="yes"/>
  </class>
  <class name="country">
    <attribute name="name" type="varchar_n"/>
    <attribute name="currency" type="uint64" references="currency"/>
    <attribute name="active" type="boolean" query="yes"/>
  </class>
</model>

```

Una vez procesamos el fichero con el generador de código obtenemos el código generado para el esquema relacional de la base de datos, los *Fisterra Barnacle Data Object*, el *mapping CORBA* y una implementación de acceso a los datos. Esto nos ha ahorrado teclear alrededor de 4000 líneas de código, que serían necesarias para implementar las operaciones básicas de acceso a datos. Y la inclusión de nuevas clases realmente se convierte en una tarea muy sencilla.

```

./database/pruebadb.sql
./src/common/dataobjects/f_do_currency.h
./src/common/dataobjects/f_do_country.h
./src/common/dataobjects/f_do_currency.c
./src/common/dataobjects/f_do_currency.c
./src/common/dataobjects/Makefile.am
./src/common/dataobjects/f_do.h
./src/common/com/f_com_corba_model.idl
./src/common/com/f_com_mapping.h
./src/common/com/f_com_mapping.c
./src/server/daos/f_dao_currency.h
./src/server/daos/f_dao_country.h
./src/server/daos/f_dao_currency.c
./src/server/daos/f_dao_country.c
./src/server/daos/f_dao.h
./src/server/daos/f_dao.c

```

Una vez tenemos el código generado el siguiente punto es utilizarlo, para esto sirve el siguiente ejemplo que crea un objeto de tipo moneda y lo inserta en la base de dato. Es de reseñar que no es necesario escribir ni una línea SQL ya que la capa de persistencia enmascara este funcionamiento apoyándose en `libgda`.

```

static void test_f_do_currency(void){
  FDOCurrency * currency1 = f_do_currency_new();
  f_do_currency_set_name(currency1,"Euro");
  f_do_currency_set_symbol(currency1,"Eur");
  f_do_currency_set_conversion_rate(currency1,1.0);
  f_do_currency_set_active(currency1,TRUE);
  f_dao_currency_insert(currency1);
}

```

Actualmente el código generado depende de `fisterra-base`, permite las operaciones de consulta sobre los atributos definidos mediante operaciones de igualdad, comparadores numéricos y similitud de cadenas.

5. Conclusiones y trabajo futuro

Como se ha mostrado en este trabajo, el trabajo con objetos persistente utilizando la tecnología *Fisterra Barnacle* nos permite reducir tiempos en implementación en una tarea tediosa y mecánica como es la persistencia de objetos.

La implementación actual es utilizable en entornos de producción con limitaciones en cuanto a rendimiento y versatilidad en las asociaciones y navegación de objetos. La implementación de una caché de objetos y gestión de consultas programadas deberían subsanar estos problemas.

Para alcanzar una solución adecuada se deberían cumplir las siguientes propiedades:

Arquitectura flexible , debe permitir fuentes de datos diferentes, es importante que la arquitectura no haga ninguna exigencia a la base de datos fuera del standard SQL.

Rendimiento adecuado en función de las necesidades. Será necesaria una evaluación exhaustiva de las diferencias con un acceso directo a la base de datos.

Control de concurrencia , recuperación y alta disponibilidad.

Mayor integración con GNOME , sistema de objetos GObject, este es un requisito de acuerdo con a la apuesta de *Igalia* por una tecnología de Software Libre. Además esta tecnología debería soportar la adaptación a múltiples lenguajes de programación orientada a objetos, como puede ser Java, C++ o Python.

Referencias

- [1] García Castro A., Dapena Paz J. *GNOME for business appliances: case study and architecture proposal*. GUADEC IV Dublin, 2003.
- [2] García Castro A., Dapena Paz J., Casanova Crespo J.M., Sánchez Penas J.J. *Fisterra 2: Software libre para la gestión empresarial* OSIC Malaga, 2004
- [3] García Castro A., Dapena Paz J., Casanova Crespo J.M., Sánchez Penas J.J. *Hacia Fisterra 2.0: Aplicaciones de Empresa para PYMES* Hispalinux Mostoles, 2003