Паттерн Proxy для Files

Кушнир П.М.
Кузьмицкий И.А.
(К.И.А: 26.11.2012)

1. Задача

В рамках функционирования информационной системы предприятия (ИСП) для обмена данными между клиентами ИСП требуется создание общих файловых ресурсов в локальной сети как можно более простым и удобным способом. Общие файловые ресурсы должны быть защищены от нежелательного воздействия, как то: несанкционированный доступ или вирусная атака.

2. Предпосылки возникновения задачи

Обычно создание общих ресурсов делается средствами операционной системы путём создания сетевой папки на файл-сервере. Но этот способ имеет очевидный недостаток в виде возможности вирусного заражения файлов в общей папке и моментального инфицирования всех участников сети, работающих с этой папкой. Кроме этого, папку придётся защищать от любопытных пользователей. Таким образом, для поддержки общего доступа требуются внешние по отношению к ИСП механизмы, требующие участия квалифицированного администратора для настройки.

Как правило, ИСП имеет в комплекте центральный сервер баз данных, доступный клиентам ИСП. Доступ как к ИСП, так и к самому серверу ограничивается встроенными системами разделения доступа. Сам же сервер БД способен хранить бинарные ресурсы (с некоторыми ограничениями по размеру), что наводит на мысль использовать базу данных как хранилище файлов.

В случае размещения файлов в БД, автоматически решаются проблемы безопасности, вирус не имеет доступа к общему хранилищу и вероятность эпидемии значительно снижается.

Размещать файлы в БД возможно с помощью специальных клиентских средств. Поскольку доступ к ИСП производится через клиентское программное обеспечение на базе BlackBox Component Builder (Блэкбокс), то постановка задачи сводится к разработке компонента, обеспечивающего т.н. *файловый доступ* к базе данных.

3. Решение и возможные применения

Каркас Блэкбокс имеет в своём составе файловый компонент *Files* [1]. Это абстрактный интерфейс, содержащий в себе всё необходимое для работы с файлами. Интерфейс *Files* реализуется отдельным компонентом *HostFiles*, в котором содержатся платформенно-зависимые механизмы. Прикладные обращения к интерфейсу *Files* транслируются в вызовы *HostFiles*.

Как правило, прикладные системы работают исключительно с интерфейсом *Files*. Таким образом, становится возможно обеспечить доступ к файлам, размещённым в БД, не затрагивая прикладной логики и не добавляя никаких специальных прикладных средств для создания общих файловых ресурсов. Для этого необходимо сохранить уже работающие файловые механизмы, лишь добавив к ним специальные возможности для доступа к центральной БД.

В сборнике шаблонов проектирования GoF [2] упоминается структурный паттерн Заместитель (Proxy). Этот шаблон описывает структуру, которая контролирует доступ к другому объекту. Для наших целей требуется именно такой паттерн, ведь удобство Заместителя в том, что остальные компоненты вообще не видят подмены, потому что продолжают использовать интерфейс *Files*.

Итак, мы реализуем шаблон Заместитель, который встраивается в абстрактный интерфейс *Files* и обрабатывает обращения к некоторым каталогам файловой системы. Остальные запросы передаются в *HostFiles*, и файловая система работает как обычно. Другими словами, мы получаем “портал” в базу данных. Всё, что записывается в специальный каталог, автоматически падает в особую таблицу из подключенной БД. А если обращаешься к такому каталогу, он выдаёт список файлов из базы данных или хэндлер файла.

Листинг 1. Интерфейс модуля-прокси

```
DEFINITION DbHostFiles;

	IMPORT DbCore, ypkCryptMd5, Files;

	TYPE
		Proxy = POINTER TO ABSTRACT RECORD
			(this: Proxy) List (iri: IRI): Files.FileInfo, NEW, ABSTRACT;
			(this: Proxy) Read (iri: IRI): Files.File, NEW, ABSTRACT;
			(this: Proxy) Write (f: Files.File; iri: IRI), NEW, ABSTRACT
		END;

		IRI = ARRAY 256 OF CHAR;


	VAR
		ir, ur: Rec;
```

```
        packedDir-, stdDir-: Files.Directory;

    PROCEDURE ConnectTo (database: DbCore.Database);
    PROCEDURE RestoreFilesDir;
    PROCEDURE SetFilesDir;
    PROCEDURE Share (IN s: ARRAY OF CHAR);

END DbHostFiles.
```

Рассмотрим интерфейсные процедуры.

Процедура *ConnectTo* подключает модуль к текущему соединению с базой данных.

Тип данных *DbCore.Database* описан и реализован в подсистеме *Db*, рассмотрение которой выходит за рамки статьи. Стоит только сказать, что *DbCore.Database* является абстрактным типом, позволяющим отправлять в базу данных SQL-запросы, а реализации этого типа учитывают особенности используемого сервера БД.

Процедура *SetFilesDir* устанавливает экземпляр Заместителя в интерфейс *Files*, после чего все прикладные обращения к *Files* будут сначала обработаны Заместителем, который после необходимых действий вернёт управление в *Files*. Чтобы убрать Заместителя из *Files,* применяется процедура *RestoreFilesDir*.

Чтобы использовать каталог файловой системы как “портал” в нашу базу данных, надо сообщить имя каталога Заместителю с помощью процедуры *Share*. Этот каталог должен существовать на диске.

Интерфейс Заместителя описан абстрактной структурой Proxy. Реализация этой структуры должна учитывать особенности используемого сервера БД. Внутри модуля исполнена стандартная реализация, для сервера БД MySQL 5.1.xx.

4. Перспективы развития

Описанный в статье механизм имеет один технический недостаток: для функционирования портала необходимо создать каталог в файловой системе. Полностью виртуальный каталог был бы гораздо удобнее, но для такой виртуализации придётся дополнять модуль Заместителя новым локатором, который подменит собой стандартный *Files.Locator*.

Отдельно требуется отметить неполную совместимость описанного решения и используемых в Блэкбокс системных диалогов выбора файлов для открытия/сохранения. Так как файлы внутри портала по факту отсутствуют на локальном диске, системный диалог Windows не в силах отобразить эти файлы, в то время как все компоненты Блэкбокс “видят” портал, как например, самый простейший диалог из компонента *CpcFileBrowser* справляется с этой задачей, так как опирается на механизмы Блэкбокс.

В свою очередь, попытка заменить механизм реализации выбора файлов на уровне реализации *HostDialog* не увенчалась успехом в силу несовместимости синхронных методов хука *Dialog.GetHook* и отсутствия модальности оконной подсистемы Блэкбокс. Поэтому вопрос пользовательского взаимодействия с файловым порталом остаётся открытым. Для программного взаимодействия подобных препятствий нет.

5. Выводы

Возможности языка программирования Component Pascal вкупе с продуманными механизмами каркаса BlackBox Component Builder позволяют решать прикладные задачи простыми, но иногда неожиданными и даже неочевидными на первый взгляд способами.

Список использованной литературы

1. Документация Блэкбокс.
2. Design Patterns: Elements of Reusable Object-Oriented Software