**Использование открытых интерфейсов среды программирования Delphi**

Одной и наиболее сильных сторон среды программирования Delphi является ее открытая архитектура, благодаря которой Delphi допускает своего рода метапрограммирование, позволяя “программировать среду программирования”. Такой подход переводит Delphi на качественно новый уровень систем разработки приложений и позволяет встраивать в этот продукт дополнительные инструментальные средства, поддерживающие практически все этапы создания прикладных систем. Столь широкий спектр возможностей открывается благодаря реализованной в Delphi концепции так называемых открытых интерфейсов, являющихся связующим звеном между IDE (Integrated Development Environment) и внешними инструментами.

Данная статья посвящена открытым интерфейсам Delphi и представляет собой обзор представляемых ими возможностей. В Delphi определены шесть открытых интерфейсов: Tool Interface, Design Interface, Expert Interface, File Interface, Edit Interface и Version Control Interface. Вряд ли в рамках данной статьи нам удалось бы детально осветить и проиллюстрировать возможности каждого из них. Более основательно разобраться в рассматриваемых вопросах вам помогут исходные тексты Delphi, благо разработчики снабдили их развернутыми комментариями. Объявления классов, представляющих открытые интерфейсы, содержатся в соответствующих модулях в каталоге ...\Delphi\Source\ToolsAPI. Design Interface (модуль DsgnIntf.pas) предоставляет средства для создания редакторов свойств и редакторов компонентов.

Редакторы свойств и компонентов – это тема, достойная отдельного разговора, поэтому напомним лишь, что редактор свойства контролирует поведение Инспектора Объектов при попытке изменить значение соответствующего свойства, а редактор компонента активизируется при двойном нажатии левой кнопки мыши на изображении помещенного на форму компонента. Version Control Interface (модуль VCSIntf.pas) предназначен для создания систем контроля версий. Начиная с версии 2.0, Delphi поддерживает интегрированную систему контроля версий Intersolv PVCS, поэтому в большинстве случаев в разработке собственной системы нет необходимости. По этой причине рассмотрение Version Control Interface мы также опустим. File Interface (модуль FileIntf.pas) позволяет переопределить рабочую файловую систему IDE, что дает возможность выбора собственного способа хранения файлов (в Memo-полях на сервере БД, например). Edit Interface (модуль EditIntf.pas) предоставляет доступ к буферу исходных текстов, что позволяет проводить анализ кода и выполнять его генерацию, определять и изменять позицию курсора в окне редактора кода, а также управлять синтаксическим выделением исходного текста.

Специальные классы предоставляют интерфейсы к помещенным на форму компонентам (определение типа компонента, получение ссылок на родительский и дочерние компоненты, доступ к свойствам, передача фокуса, удаление и т.д.), к самой форме и к ресурсному файлу проекта. Также Edit Interface позволяет идентифицировать так называемые модульные нотификаторы, определяющие реакцию на такие события, как изменение исходного текста модуля, модификация формы, переименование компонента, сохранение, переименование или удаление модуля, изменение ресурсного файла проекта и т. д. Tool Interface (модуль ToolIntf.pas) предоставляет разработчикам средства для получения общей информации о состоянии IDE и выполнения таких действий, как открытие, сохранение и закрытие проектов и отдельных файлов, создание модуля, получение информации о текущем проекте (число модулей и форм, их имена и т. д.), регистрация файловой системы, организация интерфейсов к отдельным модулям и т.д. В дополнение к модульным нотификаторам Tool Interface определяет add-in нотификаторы, уведомляющие о таких событиях, как открытие/закрытие файлов и проектов, загрузка и сохранение desktop-файла проекта, добавление/исключение модулей проекта, инсталляция/деинсталляция пакетов, компиляция проекта, причем в отличие от модульных нотификаторов add-in нотификаторы позволяют отменить выполнение некоторых событий.

Кроме того, Tool Interface предоставляет средства доступа к главному меню IDE Delphi, позволяя встраивать в него дополнительные пункты. Expert Interface (модуль ExptIntf.pas) представляет собой основу для создания экспертов — программных модулей, встраиваемых в IDE c целью расширения ее функциональности. В качестве примера эксперта можно привести входящий в Delphi Database Form Wizard, выполняющий генерацию формы для просмотра и изменения содержимого таблицы БД. Определив класс эксперта, необходимо позаботиться о том, чтобы Delphi “узнала” о нашем эксперте. Для этого его нужно зарегистрировать посредством вызова процедуры RegisterLibraryExpert, передав ей в качестве параметра экземпляр класса эксперта. В качестве иллюстрации создадим простой эксперт в стиле esStandard, который при выборе соответствующего ему пункта меню Delphi выводит сообщение о том, что он запущен. Как видно из вышеприведенной таблицы, стиль esStandard обязывает перекрыть шесть методов:

Для того чтобы эксперт был “приведен в действие”, необходимо выбрать пункт меню Component/Install Component ... , выбрать в диалоге Browse модуль, содержащий эксперт (в нашем случае exmpl\_01.pas), нажать ОК, и после компиляции пакета dclusr30.dpk в главном меню Delphi в разделе Help должен появиться пункт Simple Expert 1, при выборе которого появляется информационное сообщение “Standard Expert started!”. Почему Delphi помещает пункт меню эксперта в раздел Help, остается загадкой. Если вам не нравится то, что пункт меню появляется там, где угодно Delphi, а не там, где хотите вы, возможен следующий вариант: создать эксперт в стиле add-in, что исключает автоматическое создание пункта меню, а пункт меню добавить “вручную”, используя средства Tool Interface. Это позволит задать местоположение нового пункта в главном меню произвольным образом.

Для добавления пункта меню используется класс TIToolServices — основа Tool Interface — и классы TIMainMenuIntf, TIMenuItemIntf, реализующие интерфейсы к главному меню IDE и его пунктам. Экземпляр ToolServices класса TIToolServices создается самой IDE при ее инициализации. Обратите внимание на то, что ответственность за освобождение интерфейсов к главному меню Delphi и его пунктам целиком ложится на разработчика. Попутно немного усложним функциональную нагрузку эксперта: при активизации своего пункта меню он будет выдавать справку об имени проекта, открытого в данный момент в среде: В этом примере центральное место занимает функция AddIDEMenuItem, осуществляющая добавление пункта меню в главное меню IDE Delphi. В качестве параметров ей передаются текст нового пункта меню, его идентификатор, идентификатор пункта, перед которым вставляется новый пункт, символьное представление клавиши, которая вместе с клавишей Ctrl может использоваться для быстрого доступа к новому пункту, и обработчик события, соответствующего выбору нового пункта. Мы добавили новый пункт меню в раздел View перед пунктом Watches.

Теперь познакомимся с нотификаторами. Определим add-in нотификатор, отслеживающий моменты закрытия/открытия проектов и корректирующий соответствующим образом поле, хранящее имя активного проекта (реализацию методов, не претерпевших изменений по сравнению с тпредыдущим примером, для краткости опустим): Для реализации нотификатора мы определили класс TAddInNotifier, являющийся потомком TIAddInNotifier, и перекрыли метод FileNotification. IDE будет вызывать этот метод каждый раз, когда происходит событие, на которое способен среагировать add-in нотификатор (каждое такое событие обозначается соответствующей константой типа TFileNotification). Поле Expert в классе TAddInNotifier служит для обратной связи с экспертом (метод TAddInNotifier.FileNotification). В деструкторе эксперта регистрация нотификатора снимается, и нотификатор уничтожается. А теперь проиллюстрируем использование модульных нотификаторов. Создадим add-in эксперт, выдающий сообщения о каждом акте сохранения проектного файла (реализацию уже знакомых нам методов для краткости не приводим): В данном примере add-in эксперт отслеживает события, соответствующие открытию/закрытию проектов.

При каждом открытии проекта регистрируется модульный нотификатор, соответствующий файлу проекта. В плане реализации модульные нотификаторы схожи с add-in нотификаторами: мы определяем класс TModuleNotifier, являющийся потомком TIModuleNotifier и перекрываем его методы Notify и ComponentRenamed. IDE вызывает метод Notify при возникновении определенных событий, имеющих отношение к данному модулю; внутри этого метода и определяется реакция на то или иное событие. Метод ComponentRenamed вызывается при изменении имени компонента, лежащего на форме модуля. Обратите внимание на то, что мы не используем этот метод, но обязаны его перекрыть, иначе при изменении имени компонента будет происходить вызов абстрактного метода базового класса, что приводит к непредсказуемым последствиям.

Регистрация модульного нотификатора является несколько более сложным процессом по сравнению с регистрацией add-in нотификатора: сначала мы получаем интерфейс модуля (TIModuleInterface), а затем с помощью интерфейса модуля регистрируем нотификатор. При закрытии проекта регистрация модульного нотификатора снимается (снова с использованием TIModuleInterface), и нотификатор уничтожается. В заключение покажем, как можно определять позицию курсора в окне редактора кода. Создадим эксперт, который при выборе соответствующего пункта меню выдавал бы сообщение, содержащее имя активного файла и позицию курсора в нем (приведена реализация только существенных для данного примера методов): Для определения позиции курсора мы должны получить следующую последовательность интерфейсов: модульный интерфейс (TIModuleInterface); интерфейс редактора кода (TIEditorInterface); интерфейс представления модуля в окне редактора (TIEditView).

Если при выборе пункта меню эксперта активным является файл с исходным текстом (\*.pas), то выдается сообщение, содержащее имя активного файла и текущую позицию курсора в нем. Если активным является не pas-файл, то сообщение не выдается. Для получения имени активного файла используется метод GetCurrentFile класса TIToolServices. На этом завершим рассмотрение способов использования открытых интерфейсов. CD-ROM содержит исходные тексты всех приведенных примеров. На CD-ROM также содержится более сложный и развернутый пример, содержащий add-in эксперт, позволяющий пользователю делать закладки в исходном тексте Delphi-модулей. Краткое руководство по установке и использованию Эксперта Закладок содержится в файле bkmrks97.htm. Итак, в данной статье в общих чертах рассмотрены открытые интерфейсы и приведены примеры их использования. Еще раз повторим: благодаря наличию исходных текстов открытых интерфейсов вы без труда сможете разобраться в интересующих вас деталях. Надеемся, что многообразие возможностей, предоставляемых открытыми интерфейсами, породит у вас не одну смелую и полезную идею.