Технология OLE

Связывание и внедрение объектов

OLE — это механизм, дающий возможность вставить в приложение документ, подготовленный в другом приложении.

- Когда вы вставляете табли¬цу Excel в документ Word, вы пользуетесь механизмом OLE.
- Приложение, в которое можно вставить данные из другого приложения, называется **клиентом OLE,**
- а приложение-поставщик данных — **сервером OLE.**

Механизм OLE может действовать двумя способами:

- **Внедрение (embedding).**

Внедренный объект становится **частью** того документа, в который он вставляется.

- **Связывание (linking).**

Связанный объект в приложении представляет собой не сам документ, а только **ссылку** на него.

Преимуществом связывания является то, что к документу могут иметь доступ несколько приложений одновременно.

Если одно приложение изменит документ, изменения будут учтены также во всех других приложениях, с которыми связан файл документа.

Компонент OleContainer (на вкладке System)

- Основным компонентом для работы с OLE является **OleContainer**
- Этот компонент предоставляет приложению возможность связывать и внедрять объекты, подготовленные сервером OLE.
- Контейнер OLE позволяет вставить данные из любого доступного OLE- сервера:
  - текстовый документ Word или WordPad,
  - таблицу Excel,
  - точечный рисунок Paint,
  - звук WAV и т. п.

Обычный способ работы с компонентом OleContainer

- в ответ на требование пользователя открыть объект (например, нажатием кнопки) приложение вызывает метод **TOleContainer. InsertObjectDialog.**
- Этот метод открывает диалоговое окно, содержащее список типов встраиваемых объектов, поддерживаемых системой в данный момент.
- После того, как тип объекта выбран, приложение может вызвать дальнейшие **методы** компонентаOleContainer:
    - CreateObject
    - CreateObjectFromFile
    - CreateObjectFromInfo (для внедрения объекта)
    - либо CreateLinkToFile (для связывания объекта)

КОМПОНЕНТЫ ДЛЯ КОНКРЕТНЫХ OLE-СЕРВЕРОВ

- **вкладка Servers** содержит компоненты, предназначенные для встраивания документов, подготовленных конкретными серверами OLE — приложениями Microsoft Office:
  - **MS Word,**
  - **MS Excel,**
  - **MS PowerPoint,**
  - **MS Outlook,**
  - **MS Access.**
- Эти компоненты облегчают разработчику задачу управления офисными приложениями из своей программы по сравнению с написанием программного кода.

ПРИМЕР ПРИЛОЖЕНИЯ-КЛИЕНТА OLE

Напишем приложение, умеющее отображать точечные рисунки, документы Word, таблицы Excel и объекты прочих типов.

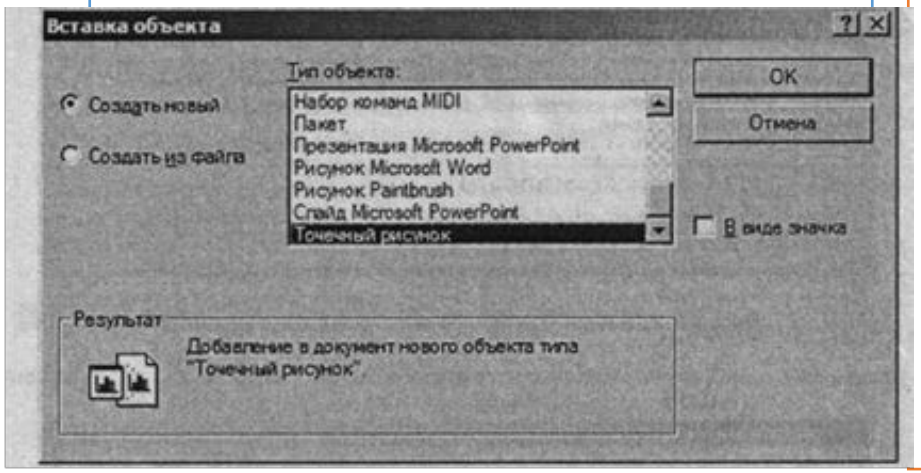
Компоненты:

•Panel (на вкладке Standard) – для размещения кнопок OleContainer и 2 x Button ;

События компонентов:

- Forml: OnCreate
- Buttonl, Button2: OnClick

Установите свойство Panell. Align равным alBottom и поместите кнопки на панель.

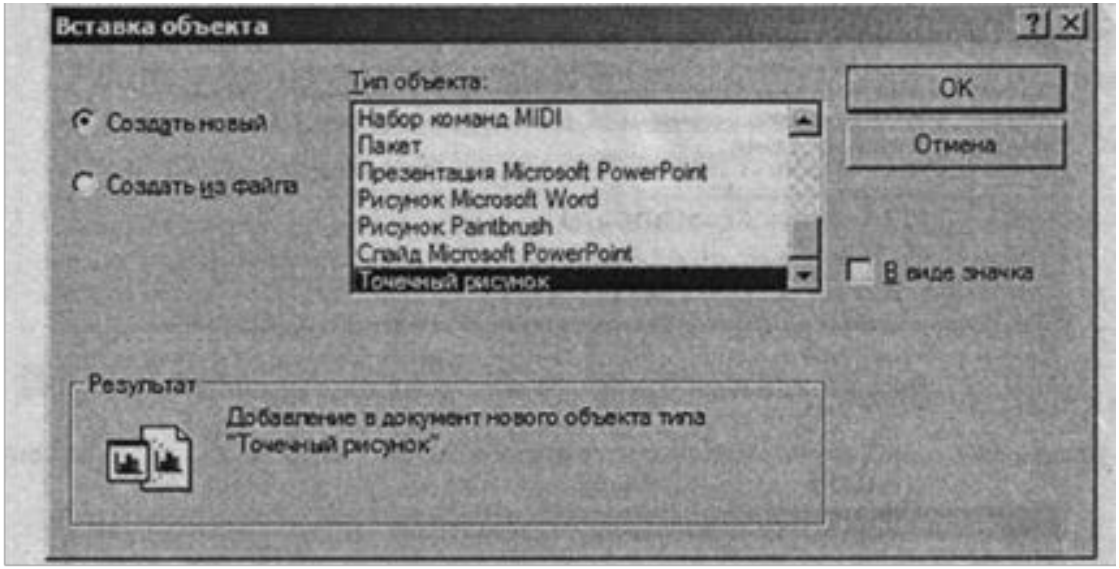



```
procedure TForm1.FormCreate(Sender: TObject);

begin
OleContainerl.Align := alClient;
OleContainerl.Ctl3D := false; // устанавливаем белый фон AutoScroll := false;
Caption := 'Учимся работать с OLE';
Panell.Caption := ' ';
Buttonl. Caption := '&Вставить объект...';
Button2.Caption := '&Готово';
end;

procedure TForm1.ButtonlClick(Sender: TObject);

begin // отображаем диалог выбора встраиваемого объекта
OleContainerl.InsertObjectDialog;
end;
```



В этом диалоговом окне можно выбрать либо создание нового объекта (на форме будет отображен пустой документ соответствующего формата), либо загрузку объекта из существующего файла.

Если вы попытаетесь открыть файл, не обслуживаемый ни одним из доступных серверов OLE, то будет сгенерировано **исключение EOleSysError,** стандартная обработка которого состоит в выдаче сообщения об ошибке.

Когда вы устанавливаете переключатель в положение Создать из файла, становится доступен флажок Связь. При установленном флажке встраиваемый объект будет связан (linked), при снятом, соответственно, — внедрен (embedded).

Всегда доступен флажок В виде значка. Его установка приводит к тому, что вместо объекта на форме будет показана иконка документов соответствующего типа.

КУДА ДЕЛОСЬ МЕНЮ?

- Когда вы открыли встроенный в ваше приложение документ Word для редактирования, вы могли заметить, что знакомый Word выглядит непривычно: а именно, не хватает строки меню

- Избавиться от этой неприятности очень просто: поместите на форму рядом с компонентом OleContainer компонент MainMenu на вкладке Standard).
- Если теперь снова собрать и запустить приложение, внедрить в него документ Word и открыть его двойным щелчком, то вы сразу же заметите изменение.

РЕДАКТИРОВАНИЕ ВСТРОЕННОГО ДОКУМЕНТА В ОТДЕЛЬНОМ ОКНЕ

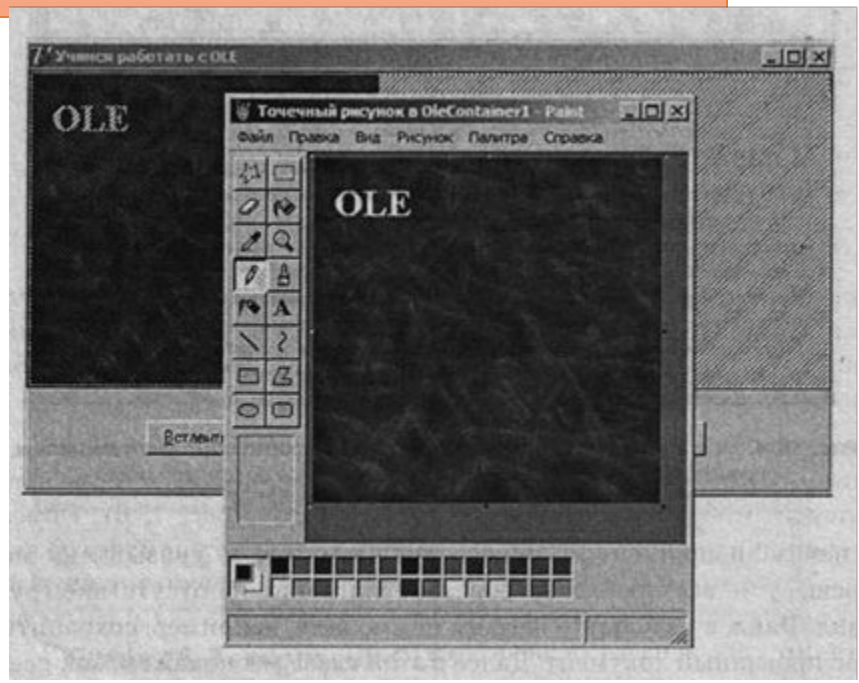
- Можно запускать приложение-сервер OLE не в окне нашего приложения, а в отдельном.
- Для установки подобного поведения служит свойство TOleContainer. AllowInPlace.
- Его значение True (по умолчанию) требует открывать документ для редактирования «на месте», то есть в текущем окне,
- а значение False требует открывать **новое окно**.

РЕДАКТИРОВАНИЕ ВСТРОЕННОГО ДОКУМЕНТА В ОТДЕЛЬНОМ ОКНЕ

- Измените метод TForml. FormCreate вашего приложения следующим образом:
- procedure TForml.FormCreate(Sender: TObject);
- begin

- OleContainerl.Align := alClient;
- OleContainerl.AllowInPlace := false;

- Теперь, если вы встроите в приложение точечный рисунок и откроете его для редактирования, то получите результат, изображенный на рис.
- Все изменения, внесенные в документ в окне редактирования, немедленно отображаются в главном окне приложения



КЛЮЧЕВЫЕ СЛОВА

- — это команды, которые можно послать этому объекту.
- Если в контейнер OLE за гружен какой-либо документ, то свойство **TOleContainer . ObjectVerbs** содержит список его ключевых свойств, а вызов метода TOleContainer. DoVerb заставляет объект OLE выполнить указанное действие.
- Свойство Obj ectVerbs имеет тип TStrings (список строк), и строки в этом списке не обработаны.
- То есть, если некоторая команда имеет клавишу быстрого доступа, то соответствующая ей строка будет содержать символ &.

Пример программы

Компоненты:

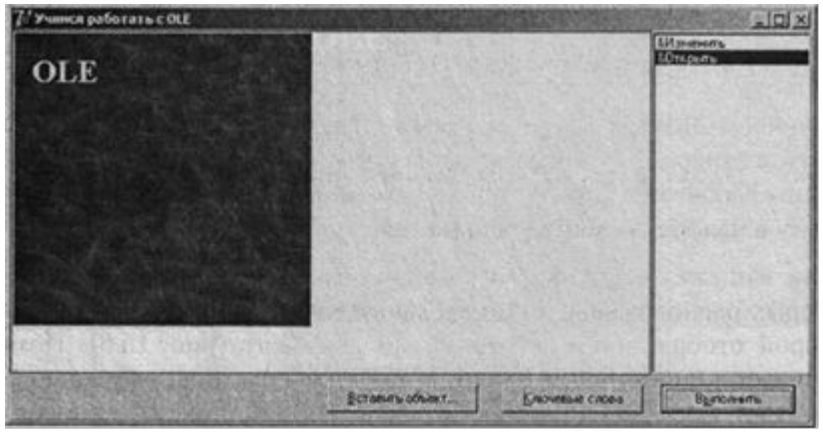
- 2 x Panel
- OleContainer
- ListBox
- 3x Button

События компонентов:

- Forml: OnCreate
- Buttonl, Button2, Button3: OnClick

1. В контейнер OLE мы загрузим объект OLE по нажатии кнопки «Вставить объект»,
2. в список ListBox выведем список его ключевых слов по нажатии кнопки «Ключевые слова»,
3. после чего можно будет выбрать из списка команду и нажать кнопку «Выполнить».

Панели, нужны для того, чтобы отделить области окна, в которых расположены элементы управления приложением, от области, в которой отображается встраиваемый документ



- Поместите все кнопки на панель Panell и привяжите ее к нижнему краю формы (Panell .Align = alBottom).
- На панель Pane12 поместите список и привяжите ее к правому краю формы (Panel2 . Align = alRight),
- после чего растяните список на всю панель (ListBoxl. Align = alClient).

```
procedure TForml.FormCreate(Sender:
TObject);

begin

OleContainerl.Align := alClient;
OleContainerl.Ctl3D := false;

AutoScroll := false;

Caption := 'Учимся работать с OLE';

 Buttonl.Caption := '&Вставить
объект...';

Button2.Caption ;= 'Ключевые
слова';

Button3.Caption := 'В&ыполнить';
end;

procedure
TForml.ButtonlClick(Sender: TObject);
begin
// отображаем диалог выбора
встраиваемого объекта
OleContainerl.InsertObjectDialog;
end;
```

Листинг программы

```
procedure TForml.Button2Click(Sender:
TObject);
begin
// копируем список ключевых слов
загруженного объекта в список ListBox

ListBoxl.Items.Assign(OleContainerl.ObjectV
erbs);
end;

procedure TForml.Button3Click(Sender:
TObject);
begin
// посылаем серверу OLE требование
выполнить выбранную команду ,
OleContainerl.DoVerb(ListBoxl.Itemlndex);
end;
```

ЧТЕНИЕ И СОХРАНЕНИЕ ДОКУМЕНТА OLE

- Следующий пример демонстрирует назначение методов LoadFromFile, CreateObjectFromFile, SaveToFile и SaveAsDocument и различие между ними.
- Компоненты
- Panel
- OleContainer
- 4 x Button
- События компонентов:
- Buttonl, Button2, Button3, Button4: OnClick
- Панель, как обычно, нужна для размещения четырех кнопок.
- Привяжите панель к нижнему краю формы и поместите кнопки на нее.
- В Инспекторе объектов озаглавьте кнопки именами четырех вышеперечисленных методов (свойство Caption).
- Значения свойств остальных объектов тоже установите с помощью Инспектора объектов.
- Можете проверить, как влияют на внешний вид приложения значения свойств OleContainerl. Align и OleContainerl.Ctl3D.

```
procedure TForml.ButtonlClick(Sender:
TObject);
begin
OleContainerl.CreateObjectFromFile(Expan
dFileName('test.doc'), false);
end;

procedure TForml.Button2Click(Sender:
TObject);
begin
OleContainerl.LoadFromFile('test.doc');
end ;

procedure TForml.Button3Click(Sender:
TObject);
 begin
OleContainerl.SaveToFile('test.doc');
end;

procedure TForml.Button4Click(Sender:
TObject); begin
OleContainerl.SaveAsDocument('test.doc');
end;
```

Функция ExpandFileName необходима потому, что метод CreateObjectFromFile требует в качестве аргумента полного пути к файлу.

Данные в одном и том же файле TEST.DOC могут храниться в двух форматах: как обычный документ Word и как объект OLE.

Если в файле хранится документ Word, то загружать его нужно методом CreateObj ectFromFi le,
а метод LoadFromFile выдаст сообщение об ошибке;
и наоборот, если в файле хранится объект OLE, то читать его можно только методом

Метод SaveToFile сохранит текущее содержимое контейнера OLE как объект OLE, а метод SaveAsDocument — как документ Word.

Почему же приложение, написанное нами в п. 15.2, не делало никакого различия между объектами OLE и обычными файлами?

Дело в том, что там мы не вызывали методов чтения непосредственно, а загружали данные в контейнер с помощью диалога InsertObjectDialog, который автоматически преобразует документ в объект

WORD, EXCEL И PAINT «В ОДНОМ ФЛАКОНЕ»?

Сейчас мы напишем приложение, обладающее всей функциональностью MS Word, MS Excel, графического редактора Paint и других приложений- серверов OLE.

Каждое из этих приложений умеет работать лишь с файлами своего типа (так, Excel не предназначен для обработки точечных рисунков), а наше небольшое приложение окажется способно заменить их все!.

Разумеется, речь идет не о действительной замене, а лишь о том, что наша программа будет в нужный момент передавать управление одному из установленных на вашем компьютере серверов OLE. Таким образом, ее можно рассматривать как «универсальную оболочку» для них.

- Строка меню нашего приложения будет содержать единственную группу команд Файл, позволяющую открывать документы разных типов.
- Открытый документ будет отображен на форме, и его можно будет активировать (двойным щелчком или выбором соответствующей команды меню), то есть запустить для его обработки подходящий сервер OLE.

- Поместите на форму компоненты OleContainer, MainMenu, OpenDialog иSaveDialog.
- Разработке и использованию меню была посвящена глава 3. Запустите Ре¬дактор меню и добавьте в компонент MainMenu пункты так, как показано на рис. 15.8. Поскольку заголовки (свойство Caption) пунктов мы предла¬гаем писать кириллицей, их автоматически сформированные имена (свой¬ство Name) будут состоять из символа N и порядкового номера. Чтобы не запутаться, замените значения Name вручную — например, так, как это сде¬лали мы (см. далее программный код).

- procedure TForml.FormCreate(Sender: TObject); begin
- Caption := 'Универсальное приложение'; AutoSize := false;
- OleContainerl.Align : = alClient;

```
// устанавливаем белый фон OleContainerl.Ctl3D :=
false;
II подгоняем размер объекта к размеру контейнера,
сохраняя пропорции OleContainerl.SizeMode := smScale;
end;

procedure TForml.InsertClick(Sender: TObject); begin
// открываем диалог внедрения/связывания
объектов OleContainerl.InsertObjectDialog;

end;
procedure TForml.OpenDocClick(Sender: TObject); begin
II запускаем QpenDialog и создаем объект OLE из
выбранного документа if OpenDialogl.Execute then
OleContainerl.CreateObjectFromFile(OpenDialogl.FileName,
false);
end;
```

```
procedure TForml.OpenOLEClick(Sender: TObject); begin
II загружаем файл, выбранный в OpenDialog, непосредственно //
как объект OLE if OpenDialogl.Execute then
OleContainerl.LoadFromFile(OpenDialogl.FileName);
end;
procedure TForml.SaveDocClick(Sender: TObject); begin
II сохраняем объект OLE в формате обычного документа if
(OleContainerl.State <> osEmpty) and SaveDialogl.Execute then
OleContainerl.SaveAsDocument(SaveDialogl.FileName);
end;
procedure TForml.SaveOLEClick(Sender: TObject); begin
II сохраняем объект OLE без преобразования в документ II
(обычные приложения не смогут его открыть)
if (OleContainerl.State о osEmpty) and SaveDialogl.Execute then
OleContainerl.SaveToFile(SaveDialogl.FileName);
end;
procedure TForml.CloseClick(Sender: TObject); begin
II уничтожаем загруженный в контейнер объект OLE
OleContainerl.DestroyObject;
end;
```

```
procedure TForml.Activatedick(Sender: TObject);

begin
// активируем загруженный в контейнер объект OLE, то есть //
запускаем соответствующий OLE-сервер if OleContainerl.State о
osEmpty then OleContainerl.DoVerb(ovShow);
end;
procedure TForml.DeactivateClick(Sender: TObject); begin
// деактивируем загруженный в контейнер объект OLE, то есть
// останавливаем соответствующий OLE-сервер if
OleContainerl.State о osEmpty then OleContainerl.Close;
end;
procedure TForml.CopyToClipboardClick(Sender: TObject); begin
// копируем загруженный объект OLE в буфер обмена if
OleContainerl.State <> osEmpty then OleContainerl.Copy;
end;
procedure TForml.PasteFromClipboardClick(Sender: TObject); begin
// если содержимое буфера допускает вставку его как
объекта OLE, // то вставим его в контейнер if
OleContainerl.CanPaste then OleContainerl.Paste;
end;
```