Технология Drag and Drop. Часть 2. Теоретические основы

Цели:

- Закрепить навыки работы с механизмом *Drag&Drop*
- Приобрести практические навыки использования механизма *Drag&Drop* для переноса информации между визуальными компонентами.

Технология *Drag&Drop* позволяет производить буксировку экранных объектов, а так же эффективно производить перенос информации между компонентами.

Перенос информации осуществляется таким же образом, что и буксировка компоненты с тем отличием, что изменению подвергаются не положение компонент (свойства ***Top*** и ***Left***), а их наполнение.

Перемещение информации так же осуществляется в четыре этапа:

- **Этап 1. Начало переноса.**

  В случае “ручного” управления началом переноса (что более целесообразно при переносе информации), свойство **DragMode** компонента задается в ***dmManual***. В этом случае необходимо обработать событие **OnMouseDown** компонента и запустить механизм переноса вручную методом **BeginDrag**, если компонент готов к переносу информации. Готовностью к переносу информации в данном случае считается, например, наличие выделенного фрагмента текста, либо элемента списка и т.п.

- **Этап 2. Перенос над компонентами.**

  Во время переноса при перемещении курсора мыши над компонентами они генерируют событие **OnDragOver**. Если компонент готов принять информацию из источника ***Source***, он выставляет параметр ***Accept*** в ***true***, если компонент не может принять информацию из указанного источника – то в ***false***.

- **Этап 3. Оставление информации на целевом компоненте.**

  При отпускании кнопки мыши над целевым компонентом он генерируют событие **OnDragDrop**. В обработчике именно этого события должно быть произведено сохранение переносимой информации в текущем компоненте.

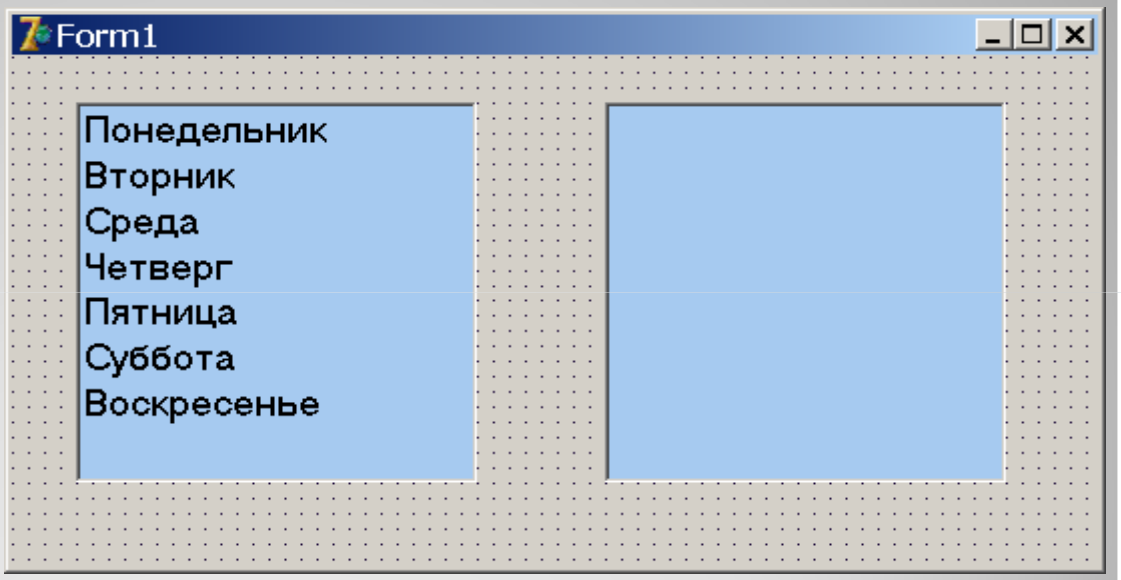
- **Этап 4. Завершение переноса**.

  Компонент - источник буксировки в момент отпускания генерирует событие **OnEndDrag**, параметр ***Target*** которого соответствует другому, целевому компоненту. В обработчике этого события можно задать действия, например, по удалению информации из компонента-источника.

  Стоит отметить, что событие **OnDragDrop** вызывается только тогда, когда компонент над которым произошло отпускание кнопки мыши принимает данные (***Accept=true*** в **OnDragOver**). Событие **OnEndDrag** вызывается всегда, даже в случае непринятия переносимых данных.

Задание
Копирование данных для списков строк

Создадим первое приложение, осуществляющее перенос информации между двумя списками (компонент ***TListBox***). Для этого создадим новое приложение. Разместим на форме два компонента ***TListBox***. Заполним свойство ***Items*** одного из списков (***ListBox1***). Полученный макет формы должен выглядеть примерно следующим образом.



Зададим события таким образом, чтобы было возможно осуществить перенос элементов из левого списка (***ListBox1***) в правый (***ListBox2***). Запуск переноса будем осуществлять вручную, поэтому свойство ***DragMode*** компонента ***ListBox1*** установим в ***dmManual***.

Определим код обработчиков событий:

- Запуск механизма переноса информации при нажатии левой кнопки мыши на компоненте ***ListBox1***. Перенос может осуществляться только в том случае, если в списке выбран элемент, подлежащий переносу (свойство ***ItemIndex*** указывает номер выбранного элемента списка).

```
procedure TForm1.ListBox1MouseDown(Sender: TObject; Button:
   TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
if (Button=mbLeft) and (ListBox1.ItemIndex<>-1)  then
ListBox1.BeginDrag(true);
end;
```

- Отметим факт начала переноса изменением цвета компонента ***ListBox1.*** Для этого зададим обработчик события ***StartDrag.***

```
procedure TForm1.ListBox1StartDrag(Sender: TObject;
  var DragObject: TDragObject);
begin
  ListBox1.Color:=clGreen;
end;
```

- Зададим обработчик события ***DragOver*** компонента ***ListBox2*** таким образом, чтобы он мог принимать информацию только из компонента ***ListBox1***.

```
procedure TForm1.ListBox2DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  Accept:=Source=ListBox1;
end;
```

- Зададим обработчик события ***DragDrop*** компонента ***ListBox2***. Именно в этом обработчике будет осуществляться физический перенос данных – добавление элемента к ***ListBox2*** и его удаление из ***ListBox1***.

```
procedure TForm1.ListBox2DragDrop(Sender, Source: TObject; X, Y: Integer);
Begin
 with ListBox1 do
  begin
   ListBox2.Items.Add(Items[ItemIndex]);
   Items.Delete(ItemIndex);
  end;
end;
```

- И, наконец, зададим процедуру завершения переноса, о чем будет сигнализировать изменение цвета компонента ***ListBox1*** на стандартный.

```
procedure TForm1.ListBox1EndDrag(Sender, Target: TObject; X, Y: Integer);
begin
 ListBox1.Color:=clWindow;
end;
```

Задайте обработчики событий таким образом, чтобы было возможно осуществить обратный перенос данных из ***ListBox2*** в ***ListBox1***.

Добавьте на форму компоненты ***TEdit*** и ***TMemo*** и задайте события так, чтобы в переносить информацию в компонент ***TMemo*** было возможно только из компонентов ***ListBox1*** и ***ListBox2***, а из ***Edit1*** – нельзя.