Как отобразить форму в приложении, которая содержится в **DLL**. Это довольно простой процесс.

Создание DLL

Сначала в этом проекте нужно создать **DLL**, которая будет содержать форму. Чтобы сделать это, следуйте за этими шагами:

1. Начните новый проект, используя *File -> New DLL*, у Вас будет шаблон **DLL**, в котором Вы должны создать форму, для этого выполните *File -> New Form*

2. Затем нужно добавить форму в проект. Но сначала сохраним форму под каким-либо именем (например *DllForm*) и добавим ее в проект, используя *File -> Add To Project...*, в диалоговом окне выберите сохраненную форму (*DllForm.pas*)

Написание кода в DLL

В этом примере будут два метода (одна процедура и одна функция) для отображения формы. Для отображения будем использовать методы **Show** и **ShowModal**.

Первая процедура, которая только покажет форму:

```
procedure ShowDllForm; stdcall;  begin    frmDllForm := TfrmDllForm.Create(nil);
frmDllForm.Show;  end;
```

Все, что мы делаем в этой процедуре - просто создаем форму, передавая **nil** как параметр, так как мы не знаем, кто владелец формы. Затем отображаем форму, используя стандартный метод **Show**. Директива **stdcall** используется, чтобы заявить, какие параметры будут передаваться процедуре.

Показ модальной формы слегка отличен, так как мы должны возвратить модальный результат. Код, чтобы сделать это:

```
function ShowDllFormModal: integer; stdcall;  begin    frmDllForm:= TfrmDllForm.Create(nil);
  Result:= frmDllForm.ShowModal;  end;
```

Единственное различие между этим кодом и предыдущим то, что мы используем функцию вместо процедуры, чтобы мы могли возвратить модальный результат формы и вызываем метод **ShowModal** вместо **Show**.

Одна вещь, которая присутствует в этиъ примерах - мы создаем форму, но нигде не уничтожаем ее, затрачивая при этом ресурсы. Самый легкий способ уничтожить форму состоит в том, чтобы сделать это в событии *OnClose* и устанавливая ее **TCloseAction**в *caFree*
(то есть, память, связанная с формой освобождается, когда форма будет закрыта). Сделайте это, вставив следующий код в событие
OnClose
формы:

```
procedure TfrmDllForm.FormClose(Sender: TObject;             var Action: TCloseAction);
begin    Action := caFree;  end;
```

Наконец, мы должны экспортировать функции в **DLL**, чтобы они могли вызываться нашим приложением. Это можно сделать, добавив следующий код в файл
DPR
:

```
Exports      ShowDllForm,      ShowDllFormModal;
```

Все, закончили писать **DLL** и продолжим писать приложение, которое будет вызывать эту **DLL**.

Создание приложения

Для создания приложения нужно выполнить следующие шаги:

1. Начните новый проект, используя *File -> New Application*, затем добавьте две кнопки на главную форму: одну для показа обычной формы в
DLL
, а другую, чтобы показать ее модально.

2. Измените надписи на кнопках, чтобы они отображали суть, что они будут делать и измените размеры в случае необходимости.

Теперь мы имеем простое приложение, которое не делает ничего. Нам нужно добавить код для кнопок, чтобы они выполняли требуемые функции.

Написание кода для Приложения

Для начала нужно написать объявления процедуры и функции, которыми мы будем вызывать **DLL**. Это делается перед разделом **implementation**:

```
procedure ShowForm;stdcall; external 'Project1dll.dll' name 'ShowDllForm'; function ShowFormModal:integer;stdcall; external 'Project1dll.dll' name 'ShowDllFormModal';
```

Снова используем **stdcall**, поскольку так была объявлена процедура в **DLL**.

Затем мы должны вызвать эти процедуры (функции), это делается в событии *OnClick* кнопок, которые мы создали ранее:

```
procedure TForm1.Button1Click(Sender: TObject); begin ShowFormModal; end;
procedure TForm1.Button2Click(Sender: TObject); begin ShowForm; end;
```

Все, что мы здесь делаем - вызываем процедуры в **DLL** в зависимости от того, какая кнопка нажата.

Заключение

Это простая демонстрация, как можно использовать формы в **DLL**. Это полезно, когда Вы хотите использовать одни формы между разных приложений. Также, когда **DLL** используется динамически (то есть связывается во время выполнения), это мощный путь для добавления новых особенностей в приложение.