МУСАЕВ А.Т., ДЖЕНТАЕВА К.Б.
Кыргызский Национальный Университет им.Ж.Баласагына
УДК 681.142.2.

EMBARCADERO RAD STUDIO XE2 LIVEBINDINGS ДЛЯ ОБЪЕКТОВ

LiveBinding *— это новый механизм связывания различных типов данных. Это статья, посвященной работе с* ***LiveBindings*** *были рассмотрены простенькие примеры того как и где может использоваться связывание любых данных с визуальными компонентами на форме. Собственно всё, что от нас требовалось — правильно составить выражение для обеспечения связи и «виртуозно» им воспользоваться.*

LiveBinding – *бул жаңы ар кандай түрдөгү байланыштын так механизми. Бул макала жумушка карай* ***LiveBindings****ге арналып жөнөкөй мисал менен каралып байланыш көрүлгөн бардык даректер визуалдык компоненттерде колдоно билүү. Өзубүздөн талап кылынган бардык- сүйлөмдүү тура түзүү байланышты камсыздоо менен «виртуоздук» колдонуу.*

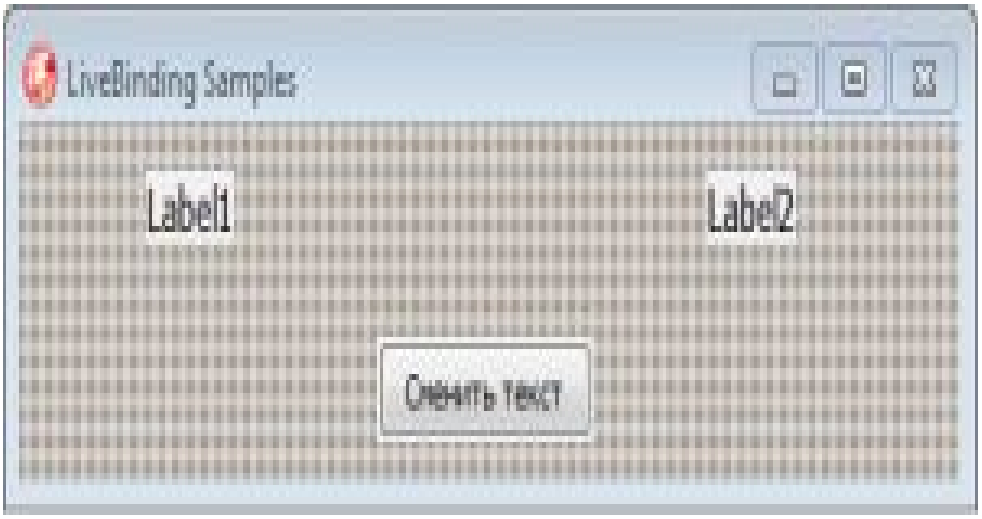
LiveBinding - *this is a new mechanism of binding of different types of data.* This article is dedicated to working with LiveBindings were considered a simple example of how and where you can use any data binding with the visual components on the form. In fact, all that was required of us - to draw up an expression for the communication and "virtuoso" to use.

LiveBinding — это новый механизм связывания различных типов данных. С помощью LiveBinding мы можем связывать различные свойства компонентов, поля баз данных со свойствами компонентов, свойтсво компонента типа Integer, со свойством другого компонента типа String без явного преобразования типов данных и т.д. и т.п.

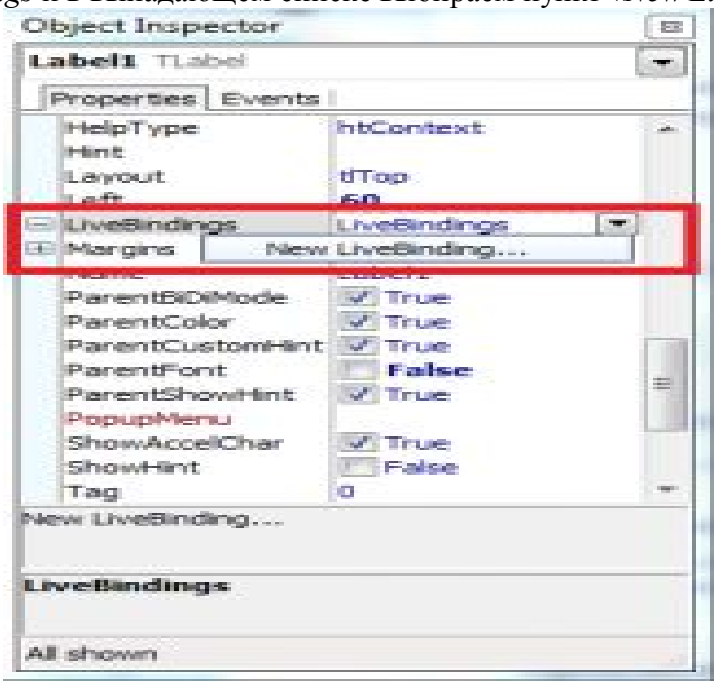
Кроме того, LiveBinding позволяет создавать новые приложения с минимальными затратами времени на написание исходного кода.

Но это все пока только слова. Давайте рассмотрим работу механизма LiveBinding на примерах, ну а вывод о том использовать или не использовать LiveBinding в своих приложениях — останется за Вами ;). С выходом Delphi XE2 у компонентов как в VCL так и в FireMonkey появилось ещё одно свойство — **LiveBindings.** Собственно, именно через это свойство мы и будем связывать данные. И начнем с простенького примерчика связывания двух метод — **Label**. К примеру, нам необходимо, чтобы при изменении текста в первой метке автоматически изменялся текст и во второй. Понятно, что такое реализовать и в **Delphi 1** элементарно, но это самый простой пример, который пришел в голову по части работы с **LiveBinding**.

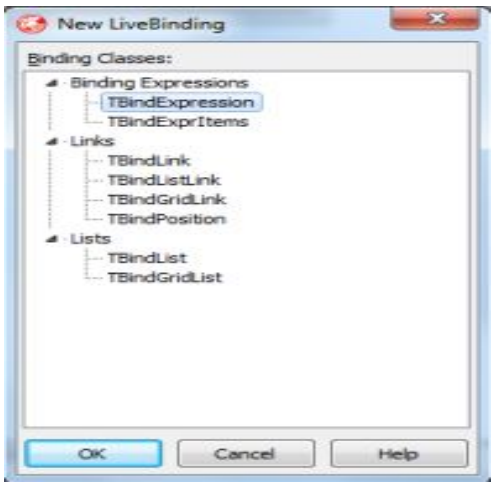
Итак, запускаем Delphi XE2, создаем новый проект «VCL Application» и бросаем на главную форму две метки TLabel и одну кнопку TButton клик по которой будет менять текст в одной из меток. Вид главной формы будет примерно такой:



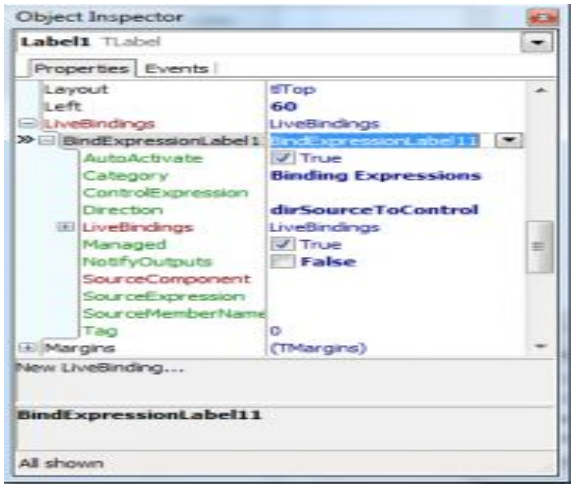
Теперь выбираем первую метку (*Label1*), в Object Inspector ищем свойство *LiveBindings* и в выпадающем списке выбираем пункт «*New LiveBinding…*«:



Это действие приведет к тому, что на форме появится новый компонент — ***TBindingsList***, в котором будут храниться все сведения о связывании данных, а также откроется окно выбора типа связывания — нам необходимо выбрать значение ***TBingExpression*** :



Жмем «Ok» и снова переходим к Object Inspector. Теперь в свойстве LiveBindings появился новый элемент — BindExpressionLabel1 свойства которого необходимо изменить:



Устанавливаем следующие свойства:

- **AutoActivate = true** — указывает на то, что выражение для связывания данных активируется автоматически во время выполнения программы.
- **ControlExpression = 'Caption'** — свойство компонента, которое будет использоваться в выражении для связывания данных. Менять будем Caption метки.
- **SourceComponent = 'Label2'** — определяет компонент, который будет использоваться в качестве источника данных для текущего компонента. Т.е. мы указываем, что данные для свойства Caption метки Label1 будут браться из компонента Label2.
- **SourceExpression = 'Caption'** - выражение для компонента-исходника данных, которое будет использоваться в выражении связывания данных.
- **Direction = dirSourceToControl** — направление связывания. В данном случае данные из источника с помощью связывающего выражения будут передаваться в компонент.

Теперь напишем пару строчек кода для кнопки:

```
procedure TForm12.Button1Click(Sender: TObject);
begin
Label2.Caption:='Test';//изменяем текст у источника
BindingsList1.Notify(Label2,'');//сообщаем об изменение
end;
```

Во второй строке мы уведомляем компонент **TBindingsList** о том, что **какое-то** свойство у Label2 было изменено и необходимо произвести изменения. Теперь все готово для запуска программы. Жмем F9 и в запущенном приложении нажимаем кнопку. Вы увидите, что при изменении Cation у второй метке автоматически изменяется и текст в первой — вот такой простой пример **LiveBinding** в действии.Теперь ещё пару слов об этом примере. Что касается метода **Notify** у **TBindingsList**, то мы могли бы сразу казать списку какое именно свойство у компонента **Label2** было изменено и написать так:

```
procedure TForm12.Button1Click(Sender: TObject);
begin
Label2.Caption:='Test';
BindingsList1.Notify(Label2,'Caption');
end;
```

В этом случае была бы проведена только одна проверка свойства, указанного во втором параметре метода.

Теперь более «сложный» пример — **связывание свойств различных типов**. На самом деле все настройки компонентов останутся практически идентичными. Итак,

бросаем на форму ещё один компонент — **TTrackBar**. Пусть теперь свойство **Position: Integer** у **TrackBar** отражается всегда в метке Label1.

Снова выбираем **Label1** и в Object Inspector добавляем новое выражение для связывания данных, на этот раз указав такие свойства нового выражения:

- **AutoActivate = true** - указывает на то, что выражение для связывания данных активируется автоматически во время выполнения программы.
- **ControlExpression = 'Caption'** - свойство компонента, которое будет использоваться в выражении для связывания данных. Менять будем Caption метки.
- **SourceComponent = 'TrackBar1'** - определяет компонент, который будет использоваться в качестве источника данных для текущего компонента. Т.е. мы указываем, что данные для свойства Caption метки Label1 будут браться из компонента Label2.
- **SourceExpression = 'Position'** - выражение для компонента-исходника данных, которое будет использоваться в выражении связывания данных.
- **Direction = dirSourceToControl** - направление связывания. В данном случае данные из источника с помощью связывающего выражения будут передаваться в компонент. У TrackBar пишем обработчик свойства OnChange:

```
procedure TForm12.TrackBar1Change(Sender: TObject);
begin
BindingsList1.Notify(TrackBar1,'Position');
end;
```

Всё. Можете запустить приложение и убедиться, что не смотря на то, что свойства **Caption** метки и **Position** у TrackBar относятся к разным типам данных, никаких исключений в работающей программе не возникает. Все преобразования проводятся для нас неявно механизмом **LiveBinding**. Как управлять преобразованиями в **LiveBinding** мы ещё посмотрим, однако факт на лицо — связывание различных типов данных сработало.

Подобным образом Вы можете производить любое количество связываний данных любых компонентов. Однако приведенный выше примеры — это довольно простые вещи, которые можно сделать без особого труда и без **LiveBinding**. Вполне возможно, что, используя Live Binding Вам придется связывать не только Integer и String, но и более сложные типы данных. Как связать свойства класса с компонентом на форме мы рассмотрим чуть ниже, а пока рассмотрим по-ближе компонент **TBindingsList**.

Литература:

1. "Основы программирования в Delphi XE Автор: Никита Культин Изд-во: БХВ-Петербург 2011
2. Delphi XE2 Автор: Дмитрий Осипов Издательство: БХВ-Петербург 2012г
3. Большой самоучитель Delphi XE2 Автор: Рубанцев В 2012г
4. Учимся программировать на Delphi 2007: О. В. Чеснокова — Санкт-Петербург, НТ Пресс, 2008 г.- 368 с.
5. Программирование на Delphi Win32: С. А. Любавин — Санкт-Петербург, НТ Пресс, 2008 г.- 576 с