Lesson 2: Turning up the Style and Data!

David Intersimone "David I"
Vice President of Developer Relations and Chief Evangelist
davidi@embarcadero.com

Mobile App Development

- Lesson 1 – Hello World! My First Multi-Device App
- **Lesson 2 – Turning up the Style and Data!**
- Lesson 3 – Accessing Local Storage
- Lesson 4 – Building Multi-tier, Multi-device Apps
- Lesson 5 – Connecting Mobile and Desktop using Tethering
- Lesson 6 – Accessing REST and BaaS Cloud Services

Replay links and lesson slides will appear on my blog
http://blogs.embarcadero.com/davidi/

Lesson 2 Agenda

- Styles
- TListBox and TListView
- LiveBindings
- PrototypeBindSource
- Samples and Snippets
- Continue development of the mobile business app
- Review, Homework and Next Time
- Q&A

Styles

- FireMonkey controls are arrangements of a tree composed of subcontrols, primitive shapes and brushes, decorated with effects and defined as styles.
- Styles are stored in a *"style book" – **TStyleBook** component. Apps have a default "style book" built in. Set a form's StyleBook property to use a different style.*
- Individual elements of a style are internally called *resources*; because that term has several other meanings, the term **style-resource** is used for clarity.
- Styles provide a great deal of customization without subclassing.
- The **StyleName** property is the name by which a style or style subcomponent is known to others and can be found.
- A control's **StyleLookup** property is set to the name of the desired style-resource to adopt that style for the specific control. When StyleLookUp is empty, the default style is used.
- The **Bitmap Style Designer** enables you to Create, Edit and Test FireMonkey styles

Style Resolutions and Platforms

- The various target platforms (Windows, Mac OS X, iOS, and Android) can support different resolutions at run time:
  - The Mac OS X and iOS platforms support the **Retina** display (2880×1800 or 5.2 megapixels), which doubles the standard resolution. So Mac and iOS support two different resolutions: 1x, 2x.
  - The Android platform supports four different resolutions: 1x, 1.5x, 2x, 3x.
  - Windows supports only standard resolution.
- iOS and Android style files are found in the
  - C:\Users\Public\Documents\Embarcadero\Studio\14.0\Styles\iOS
  - C:\Users\Public\Documents\Embarcadero\Studio\14.0\Styles\Android

Changing a Control's Style in a Mobile App

- Choose File > New > FireMonkey Mobile Application and choose any mobile template.
- Place a control on your form (for example, FMX.StdCtrls.TButton):
- In the Form Designer, select the TButton on your form.
- In the Object Inspector, click the Down Arrow in the StyleLookup property.
- In the StyleLookUp popup menu, you can see the different designs for the button, as shown in the illustration.
- Apply a style by selecting the design you want. For example, you might select the Info button
- Also, the style of a control can be set by changing the style of the entire form.

Working with Styles at runtime

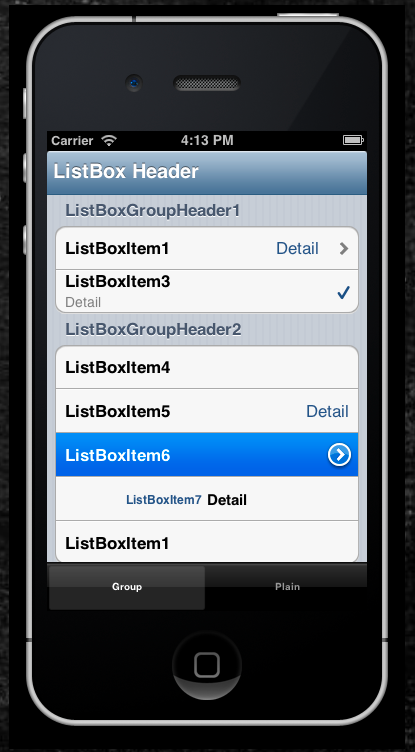
- You can load a Style at runtime from a file
  - TStyleManager.SetStyleFromFile(<style filename>);
  - TStyleManager::SetStyleFromFile(<style filename>);
  - Do not place multiple lines calling SetStyleFromFile in a project, because you can have only one active style in the style manager.
  - You can call SetStyleFromFile either in the project source code (before calling Application.Initialize) or in the initialization section of one of the form units:
    - If you call SetStyleFromFile in a form, the style is reapplied.
    - If you call SetStyleFromFile before the form is created, the custom style fully replaces the platform style.
- You can load a Style at runtime from a resource in your project
  - Add your custom styles to the project resources: Select Project > Resources and Images.
    - Load your custom Android style and set the identifer to Android<StyleName>.
    - Load your custom iOS style and set the identifier to iOS<StyleName>.
  - Style := TStyleManager.LoadFromResource(HInstance, 'AndroidDark', RT_RCDATA);
  - TStyleManager.SetStyle(Style);
  - style = TStyleManager::LoadFromResource((unsigned int)HInstance, L"AndroidDark", RT_RCDATA);
  - TStyleManager::SetStyle(style);
- http://docwiki.appmethod.com/appmethod/1.14/topics/en/Working_with_Native_and_Custom_FireMonkey_Styles

Looking at sample Controls with Styles

- Mobile Samples\User Interface\Controls
  - ToolBars
  - ToolButtons
  - Tabs
  - ListBoxes
  - Controls
  - Editors

TListBox

- TListBox displays a set of items in a scrollable list.
- TListBox can contain
  - TListBoxItem
  - TListBoxHeader
  - TSearchBox
  - TListBoxGroupHeader, TListBoxGroupFooter
- Some TListBox properties
  - MultiSelect property - set the list to accept single-item or multi-item selection
  - Each item's ItemData can have an Accessory, Bitmap, Detail, Text
  - Set different backgrounds for consecutive list items by using the AlternatingRowBackground.
- Note: TListBox performance can be slow on mobile. Use TListView if you want to develop more complex applications, especially apps with large databases.



TListView

- TListView displays a collection of items in a list that is optimized for LiveBindings and for fast and smooth scrolling.
- The items in the list view can have one or more of the following appearance features:
  - A caption or detail text (for example, using the **Item.Text** bindable member of **TListView**)
  - An associated image (for example, using the **Item.Bitmap** bindable member of **TListView**)
  - An accessory icon (for example, using the **ItemEditAppearance** property in the Object Inspector)
  - A graphic or a text button attached (for example, using the **Item.ButtonText** bindable member of **TListView**)
- Enabling the Swipe-to-Delete Feature on TListView Items
  - When the Swipe-to-Delete feature is enabled, the end user can swipe an item in a list view, and a **Delete** button temporarily appears on the item.
  - The user can then click the **Delete** button to delete the item from the list view, or release the swipe to retain the item in the list view. This feature works on mobile (that is, iOS and Android), and, when touch input is enabled, on desktop apps as well (Mac OS X and Windows).
  - To set the swipe-to-delete feature on TListView items, set the **CanSwipeDelete** property to **True**.

LiveBindings and the LiveBindings Designer

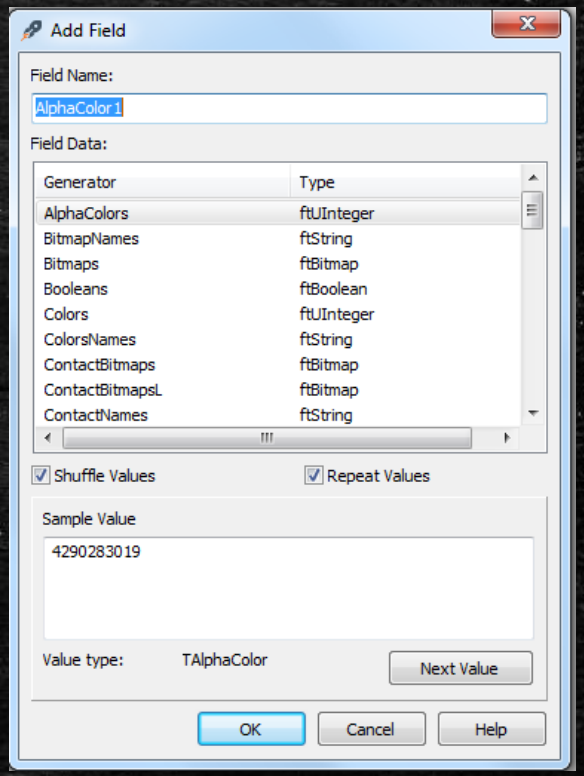
- LiveBindings is a data-binding feature supported by the FireMonkey framework
- The **primary way** to create bindings is using the LiveBindings Designer. The Designer can only create QuickBindings components.
- There is a **second way** to create such bindings, using the LiveBindings Wizard. It also only creates QuickBinding components.
- The LiveBindings Designer uses QuickBindings to create this type of bindings (which is also reflected in the wizard):
  - Link a control such as TEdit to a field in a data source
  - Link a control such as a TGrid to a data source
  - Link a control such as TEdit to a component property (such a TLabel.Text)
  - Link a component property to a field in a data source
- You can save LiveBindings Diagram as an Image

LiveBindings Wizard

- Tools | Options | LiveBindings – to display the LiveBindings Wizard in the context menu
- The context-sensitive LiveBindings Wizard is invokable through the right-click menu on a form or on any control on that form. Depending on your selection of binding tasks in the first wizard page, one or more of the wizard pages described below are accessible for you in the LiveBindings Wizard.
  - Binding Task Page
  - Data Source Page
  - Field Page
  - Options Page
  - Component Property Page
  - Control Page

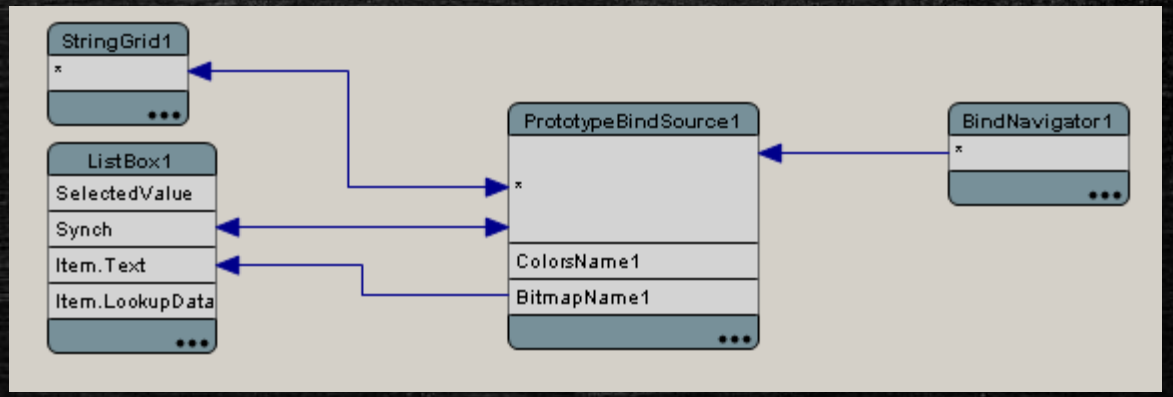
PrototypeBindSource

- When designing applications that make use of the LiveBindings framework, you can use a**TPrototypeBindSource** component readily available in the Tool Palette to generate sample data for your bindings.
- Properties
  - AutoActivate
  - AutoEdit
  - AutoPost
- Right-Mouse click context menu
  - Add fields and their types
  - Add Navigator



Synchronizing Data through LiveBindings

- Synchronize data using the **Synch** and ***** properties of certain components within the LiveBindings Designer.
- On a main form, drop these components:
  - TPrototypeBindSource -- will provide the sample data.
  - TBindNavigator -- will provide navigation functionality for the sample data.
  - TListBox -- will display some information (for instance names of alpha colors).
  - TStringGrid -- will display all information about the sample data.
    - In the Object Inspector set the Options.goEditing property to True. This allows for editing of the items directly into the string grid.



ListBox, ListView & PrototypeBindSource Samples

- Folders
  - C:\Users\Public\Documents\Embarcadero\Studio\14.0\Samples\CPP\Mobile Samples\User Interface
  - C:\Users\Public\Documents\Embarcadero\Studio\14.0\Samples\Object Pascal\Mobile Samples\User Interface
- ListBox
  - Settings Project – SettingsDemo
- ListView & PrototypeBindSource
  - ListViewCheckListProject
  - SimpleListView

Next Steps for our Business Mobile App

- Add a PrototypeBindSource for Customer data
- Add ListView to Customer tab item
- Set "CanSwipeDelete" property
- Bind PrototypeBindSource data to the ListView
- Add a ListBox to the Settings tab item
  - Set StyleLookup = transparentlistboxstyle
  - Set GroupingKind = Grouped
- Add a TListBoxGroupHeader – "Push Notifications"
- Add ListBoxItem – "Order Shipped" with a TSwitch
- Add ListBoxItem – "Parts Backordered" with a TSwitch

Lesson 2 Review

- Customize your app using styles
- Use TListBox and TListView for grouping data in your UI
- Use LiveBindings to connect data to your UI controls
- Use PrototypeBindSource to view the UI/Data until the database is ready
- Continued work on the business mobile app

Resources

- Styles - Docwiki
  - http://docwiki.appmethod.com/appmethod/1.14/topics/en/Customizing_FireMonkey_Applications_with_Styles
  - http://docwiki.appmethod.com/appmethod/1.14/topics/en/Applying_FireMonkey_Styles
  - http://docwiki.appmethod.com/appmethod/1.14/topics/en/FireMonkey_Style_Designer
  - http://docwiki.appmethod.com/appmethod/1.14/topics/en/Working_with_Native_and_Custom_FireMonkey_Styles
- ListView and ListBox – Docwiki
  - http://docwiki.appmethod.com/appmethod/1.14/libraries/en/FMX.ListView.TListView
  - http://docwiki.appmethod.com/appmethod/1.14/libraries/en/FMX.ListView.TListViewItem
  - http://docwiki.appmethod.com/appmethod/1.14/libraries/en/FMX.ListBox.TListBox
  - http://docwiki.appmethod.com/appmethod/1.14/topics/en/Mobile_Tutorial:_Using_LiveBindings_to_Populate_a_ListView_(iOS_and_Android)
- LiveBindings and PrototypeBindSource
  - http://docwiki.appmethod.com/appmethod/1.14/topics/en/LiveBindings_Designer
  - http://docwiki.appmethod.com/appmethod/1.14/topics/en/LiveBindings_Wizard
  - http://docwiki.appmethod.com/appmethod/1.14/topics/en/Creating_LiveBindings
- Blogs
  - http://blogs.embarcadero.com/
  - Jim McKeeth - http://delphi.org/
  - Sarina Dupont - http://blogs.embarcadero.com/sarinadupont/

Note: http://docwiki.appmethod.com/appmethod/1.14/topics/en/… = http://docwiki.embarcadero.com/RADStudio/XE6/en/…

Homework & Next Time

- Create your own apps using Styles, TListBox, TListView, LiveBindings and the PrototypeBindSource
- Take a look at more mobile samples and snippets
- Explore the Docwiki articles and tutorials listed on the Resources page
- Continue work on the business mobile app
- Lesson 3 – Accessing Local Storage
  - Local Storage
  - IniFiles
  - FireDAC database access components
  - FDMemTable – in memory dataset
  - Mobile local SQL databases SQLite, IBLite, IBToGo

Note: http://docwiki.appmethod.com/appmethod/1.14/topics/en/... = http://docwiki.embarcadero.com/RADStudio/XE6/en/...

Q&A

Thank You ☺

davidi@embarcadero.com