Containers and Collection Controls

Brian Noyes
CTO, Solliance (www.solliance.net)
brian.noyes@solliance.net, @briannoyes

pluralsight
hardcore developer training

Outline

Understanding Data-driven Composition

ComboBoxes and ListBoxes

TabControls

DataGrids

Collection Views

Data-Driven Visual Composition

- **ContentControl and ItemsControl dynamically generate child elements at runtime**
  - Based on bound data objects
  - Can be driven by DataTemplates
- **ContentPresenter generated by ContentControl**
  - Renders out the Content of the control
  - Also used by many other controls – Window, UserControl, TabControl, DataGridCell, etc
  - DataContext set on this element as the root container for the content
- **ItemsPresenter generated by ItemsControl**
  - Renders out a collection of Item container controls
    - Type specific to the control – i.e. ComboBoxItem, ListBoxItem, TabItem, etc.
  - Each item rendered generally uses a ContentPresenter somewhere within its child elements to render the dynamic content of a single item
  - DataContext set on each Item container control to current data object in collection

ComboBoxes and ListBoxes

- **Both derive from ItemsControl**
- **Data binding properties:**
  - ItemsSource – provide the collection for the list
  - DisplayMemberPath – property name on each object to obtain string for rendering in list
  - SelectedValuePath – property name on each object to obtain Id-like value to identify the "value" of each item
  - SelectedValue – value of currently selected item's SelectedValuePath property
  - SelectedItem – reference to the selected data bound object (i.e. Customer)
  - ItemTemplate – provide the DataTemplate to use for rendering each item
- **ListBox:**
  - SelectedItems – supports multi-select
    - Not easy to databind – read-only
    - Can write a custom behavior to monitor SelectionChanged event, read SelectedItems, push through a property binding on the behavior

TabControls

- **Great for dynamic view containment scenarios**
- **Derives from ItemsControl**
- **Each TabItem contains two ContentControls**
  - Header – what shows up in the tab
  - Content – client area of tab
- **TabControl DataTemplates**
  - ItemTemplate applies to the Header of each TabItem
  - ContentTemplate applies to the Content of each TabItem

DataGrids

- **ItemsControl derived**
- **Editable**
- **Supports dynamic column generation, static column definitions, or template columns**
- **Templating:**
  - Cell level: defined through template column
    - Separate templates available for normal vs edit mode
  - Row details:
    - Can provide a DataTemplate that gets rendered out as a detail view under each row
- **DataGridColumns are not part of the visual tree**
  - RelativeSource and ElementName bindings won't work in the Binding property of a column
    - Use template columns instead

Working With CollectionViewSource

- **Wraps a collection**
- **Exposes an ICollectionView**
  - Maintains current object in collection
- **Can use to morph the rendered data collection:**
  - Sort
  - Filter
  - Group
- **Supports Master-Details binding nicely**
  - Chain CollectionViewSources
    - Set Source of one CVS to another with a property path

Summary

ContentControl / ItemsControl play pivotal role in UI composition

ComboBoxes / ListBoxes have multiple data binding focused properties

TabControls can be data bound and DataTemplate rendered

DataGrids are just another ItemsControl at a top level

DataGrid column bindings result in cell-level container bindings

DataGrid CellTemplates give you control of cell-level visual structure

DataGrid row details templates let you provide drill-down UI for a selected row

CollectionViewSource gives you filtering, sorting, grouping & master-details capabilites