


LVL08
**Black Belt Silverlight Business
Data Validation**

Brian Noyes
Chief Architect, IDesign Inc (www.idesign.net)
brian.noyes@idesign.net, @briannoyes



About Brian

Chief Architect
IDesign Inc. (www.idesign.net)

Microsoft Regional Director
(www.theregion.com)

Microsoft MVP
Silverlight

E-mail: brian.noyes@idesign.net
Twitter: @briannoyes
Blog: <http://briannoyes.net>

Publishing

Developers Guide to Microsoft Prism 4, O'Reilly & Assoc., March 2011

Developing Applications with Windows Workflow Foundation, LiveLessons training DVD, June 2007

Smart Client Deployment with ClickOnce, Addison Wesley, January 2007

Data Binding in Windows Forms 2.0, Addison Wesley, January 2006

MSDN Magazine, MSDN Online, CoDe Magazine, The Server Side .NET, asp.netPRO, Visual Studio Magazine

Speaking

Microsoft TechEd US, Europe, Malaysia, DevConnections, DevTeach, others



Agenda

- Silverlight data binding review
- Validation with exceptions
- Validation with IDataErrorInfo
- Validation with INotifyDataErrorInfo
- Custom Validation Display
- Data Annotations
- Building your own rules engine
- Leveraging WCF RIA Services



SL Data Binding

- **DataContext**
 - The data that flows into your element bindings
- **Binding**
 - Gets data into your XAML element properties from bound object properties



Binding Participants

Target ↓ Source → {DataContext, Element, RelativeSource, Source}

```
<TextBox Text="{Binding Path=Name}"/>
```

- **Source**

- The object / property that the binding “points” to through its source/path
- Value is retrieved on initial binding
- Value is retrieved when PropertyChanged fires
- Value is set depending on binding Mode and UpdateSourceTrigger



Binding Participants

- **Target**

- The bound element property that the Binding is being set on
- `<TextBox Text={Binding ...} />`
- Must be a DependencyProperty on a DependencyObject



Binding Workflow

- XAML gets parsed
- Binding gets reference to source object through DataContext, Source, RelativeSource, or ElementName
- Binding calls get{} block called on source property
- Binding sets target property with value retrieved, passing through Converter if set, and TypeConverter if needed



Binding workflow

- **Target property changes**
 - User interaction or programmatic
 - Immediately if UpdateSourceTrigger=PropertyChanged or on focus change by default, source property is set{} to new value
 - Passing through Converter if set and TypeConverter if needed
 - Binding invokes validation
- **Source property changes**
 - Raises PropertyChanged event
 - Binding calls get{} block
 - Binding invokes validation



Binding Validation

- **By default, no validation**
- **If `ValidatesOnExceptions`**
 - Treats any exception thrown when setting source property as validation error
- **If `ValidatesOnDataErrors`**
 - Calls the `IDataErrorInfo` API to get errors *only for the property that was just set*
- **If `ValidatesOnNotifyDataErrors`**
 - Calls the `INotifyDataErrorInfo` API to check immediately for errors and subscribe for changes in errors for the property the binding is tied to



Validating with Exceptions

- **Source property set{} block throws exception**
- **Swallowed by the binding by default**
- **`ValidatesOnExceptions=true`**
 - Input control will display a validation error
 - Error message = `Exception.Message`



Validating with IDataErrorInfo

- `ValidatesOnDataErrors=true`
- Source object implements `IDataErrorInfo`
- After source property is set
 - Binding calls indexer and `Error` properties on `IDataErrorInfo`
 - If non-null or empty string returned, treated as validation error
 - Returned string from indexer is used for property validation error
 - Returned string from `Error` property is used for object level validation indication (i.e. row in `DataGrid`)
- Implementation of indexer invokes property level validation logic



Validating with INotifyDataErrorInfo

- `ValidatesOnNotifyDataErrors = true`
- Source object implements `INotifyDataErrorInfo`
- Binding subscribes to `ErrorsChanged`
- After source property set, calls `GetErrors` for property
- If `ErrorsChanged` fires, calls `GetErrors` again
- Strings returned are used for error indications
- Designed for asynchronous calls to the back end to obtain validation errors



Validation Display

- Each Silverlight control has an error template built in to its default control template
- Can be customized through styling or visual states
- ValidationSummary control displays all validation errors within a common parent element



Using DataAnnotations

- `System.ComponentModel.DataAnnotations`
- Declarative attributes for:
 - [Required]
 - [Range]
 - [RegEx]
 - [StringLength]
 - [CustomValidation]
- Some code needs to go looking for them
 - DataGrid / DataForm
 - WCF RIA Services built in
 - Base class infrastructure



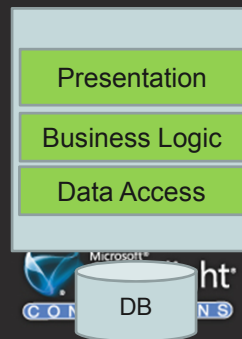
Building Your Own Rules Engine

- Base class infrastructure for your model and view model objects
- Common implementation of IDataErrorInfo / INotifyDataErrorInfo
- Way to plug in rules for evaluation by bound property modifications
- Indicate property dependencies
 - Raise PropertyChanged notifications for coupled properties



WCF RIA Services Overview

- Simplifies building N-tier Line of Business (LOB) applications
 - Highly dependent on push pull of data
 - Can use for non-CRUD operations as well
- Architecture and tools for building the glue code between the client and the back end
 - Streamlined pipeline for data and operations between client and server
 - Clients: Silverlight & ASP.NET
 - Future releases: ASP.NET client side JavaScript (WCF JQuery)



RIA Services Validation

- Data Annotation Attributes
- Server Update method logic
- Async [Invoke] method execution



Summary

- Rich binding infrastructure in Silverlight
- Many ways to indicate validation errors
- INotifyDataErrorInfo the most flexible
- Consider WCF RIA Services for pre-built infrastructure
- Or write your own (DRY)



Resources

- Data Validation with Silverlight 3 and the DataForm, John Papa, MSDN Magazine, Oct 2009, <http://msdn.microsoft.com/en-us/magazine/ee335695.aspx>
- Enforcing Complex Business Data Rules with WPF, Brian Noyes, MSDN Magazine, June 2010, <http://msdn.microsoft.com/en-us/magazine/ff714593.aspx>
- WCF RIA Services Validation, Brian Noyes, The SilverlightShow, <http://www.silverlightshow.net/items/WCF-RIA-Services-Part-6-Validating-Data.aspx>

E-mail: brian.noyes@idesign.net
Twitter: @briannoyes
Blog: <http://briannoyes.net>



Your Feedback is Important

Please fill out a session evaluation form
drop it off at the conference registration
desk.

Thank you!

