Best Practices: Building a Real-World Microsoft Silverlight Line-Of-Business Application

Microsoft® tech·ed Africa | 2010

OCTOBER 17-20, 2010 | DURBAN, SA

Web and User Experience

WUX 4 07

Level 400

Rudi Grobler

Barone, Budge & Dominick

Blog

http://www.rudigrobler.net

Twitter

@rudigrobler

DTL323 - Using the MVVM Design Pattern with the Microsoft Visual Studio 2010 XAML Designer
Mon, 18 Oct 2010 (10:45 - 11:45) | Breakout Session | Sessions Room C3 | Level: 300 - Advanced

WTB312 - Powering Rich Internet Applications: Windows Server AppFabric, Web Services, and Microsoft Silverlight
Tue, 19 Oct 2010 (14:30 - 15:30) | White Board | Session Room D2 | Level: 300 - Advanced

WUX310 - Securing Microsoft Silverlight
Tue, 19 Oct 2010 (17:15 - 18:15) | Breakout Session | Session Room A3 | Level: 300 - Advanced

WUX407 - Best Practices: Building a Real-World Microsoft Silverlight Line-of-Business Application
Wed, 20 Oct 2010 (08:30 - 09:30) | Breakout Session | Session Room D4 | Level: 400 - Expert

Software development in the “Real” world

Agenda

- Best Practices:
  - Lesson #1: Pick a Pattern and Stick to It
  - Lesson #2: Data Binding and Nested Controls
  - Lesson #3: Notify Users of Successes (and failures)
  - Lesson #4: Get an Agent - A Service Agent
  - Lesson #5: Extend Existing Controls

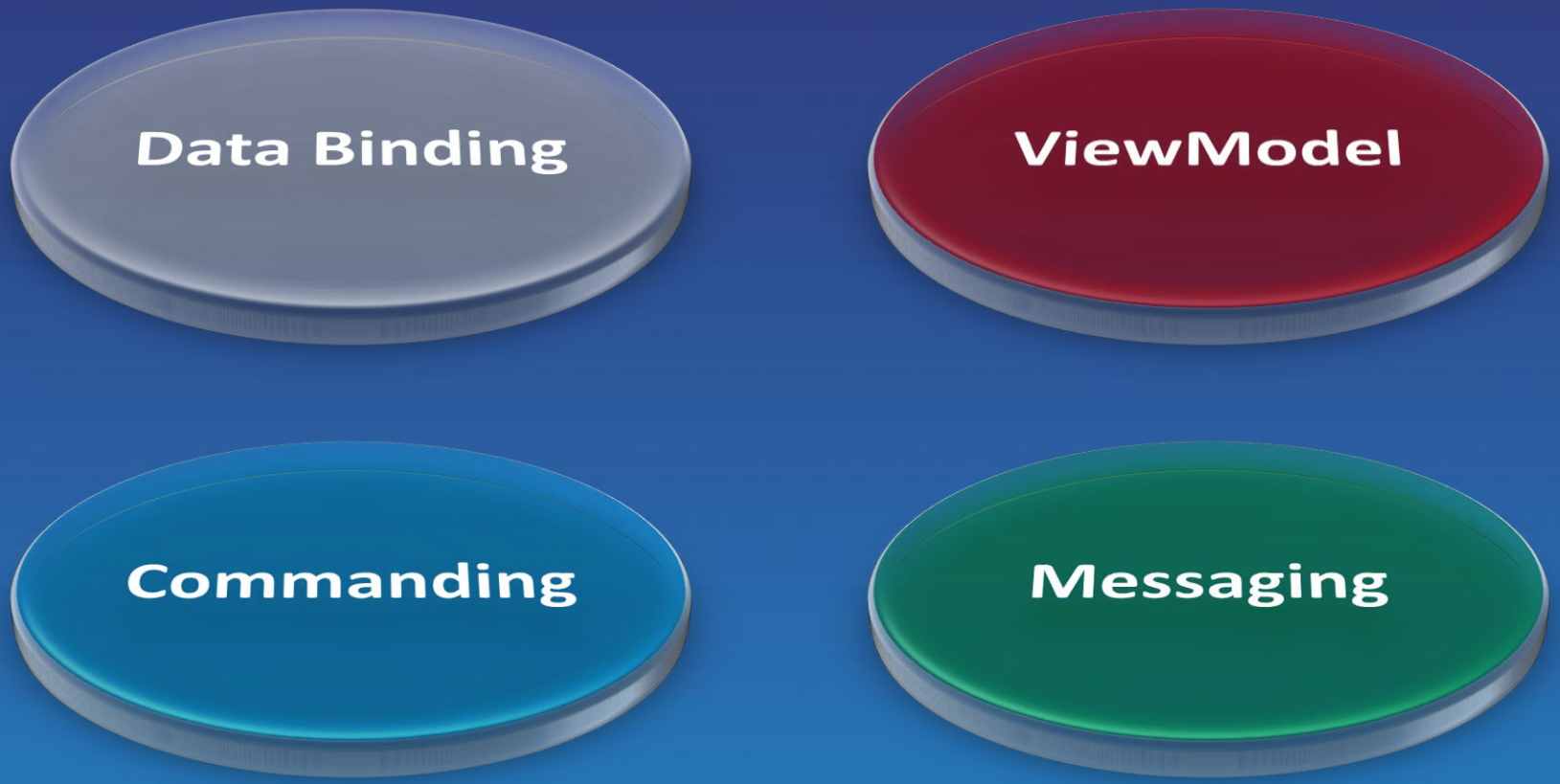
Lesson #1: Pick a Pattern and Stick to It

- Silverlight Line-of-Business (LOB) applications can grow out of control without proper planning:
  - Typically has many screens with supporting code
  - Validation code
  - Code to call the application's services
  - Service code
  - Model code
  - Data access code

The MVVM Pattern

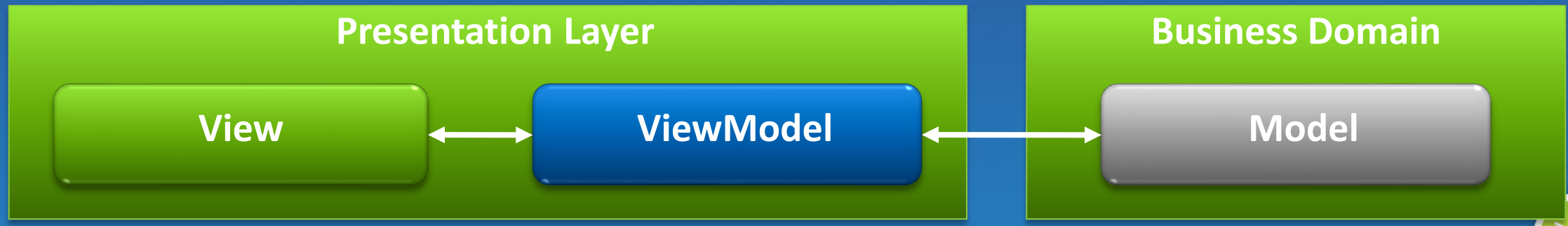
- The **Model-View-ViewModel** (MVVM) works very well for Silverlight LOB applications:
  - Provides consistency across code
  - Allows for better code re-use
  - Allows for Separation of Concerns (SoC)
  - Supports Unit Testing
  - Used as a type of controller for your view
  - Silverlight 4 provides ICommand support
- Code-behind IS NOT evil but minimize the code you add there

MVVM Pattern Enablers



MVVM Pattern Players

- The MVVM pattern has 3 key players:
  - **Model**: Business domain (business rules, data access, model classes)
  - **View**: User interface (Silverlight screens)
  - **ViewModel**: Middle-man between View and Model – similar to a controller



MVVM

DEMO

What every MVVM framework needs?

- ViewModel base class
- ICommand implementation
- Event bus

CALIBURN
EX CALCE LIBERATUS

MEFed MVVM

Calcium

MVVM Foundation

MVVM Light Toolkit

Microsoft®
patterns & practices
proven practices for predictable results

nRoute
framework

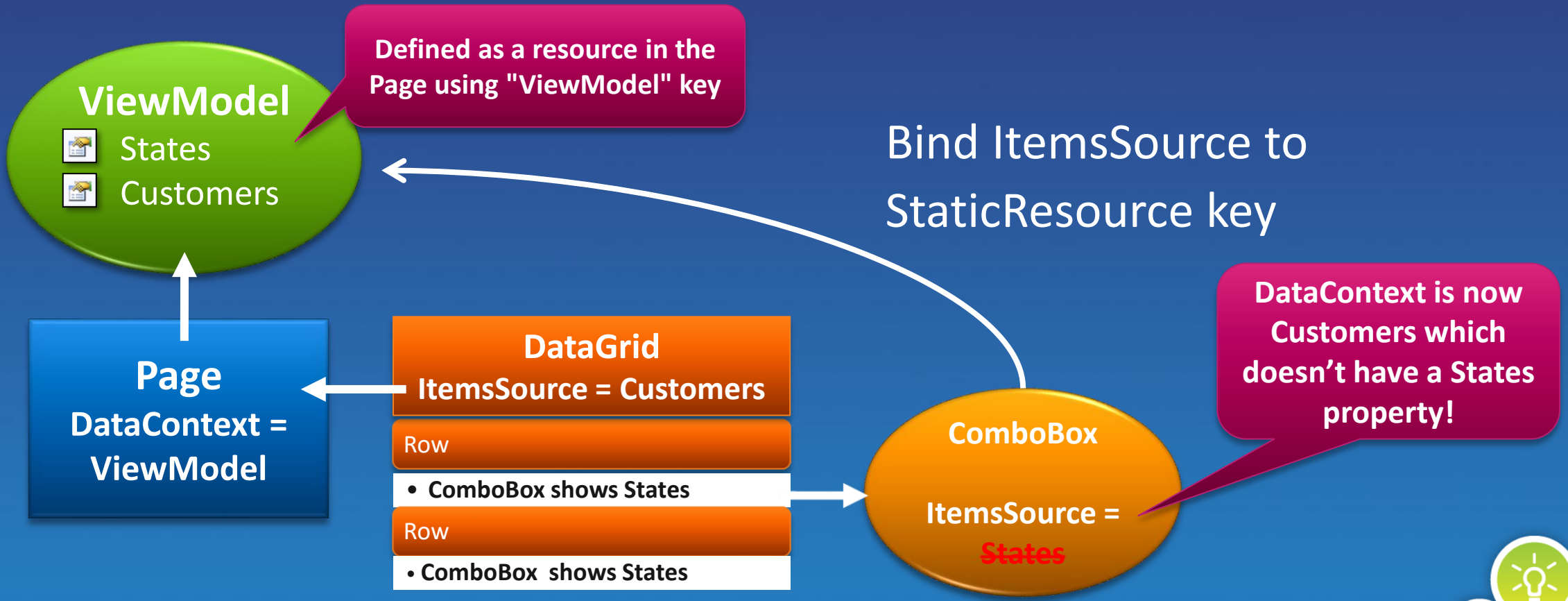
Cinch

Lesson #2: Data Binding in Nested Controls

- Binding data in nested control scenarios can be challenging
- Data required by controls may be out of scope due to where it's defined. For example:
  - Items controls nested in a DataGrid or ListBox
  - Items controls inside of a nested User Control
- How do items controls get their data when the target collection is out of scope?

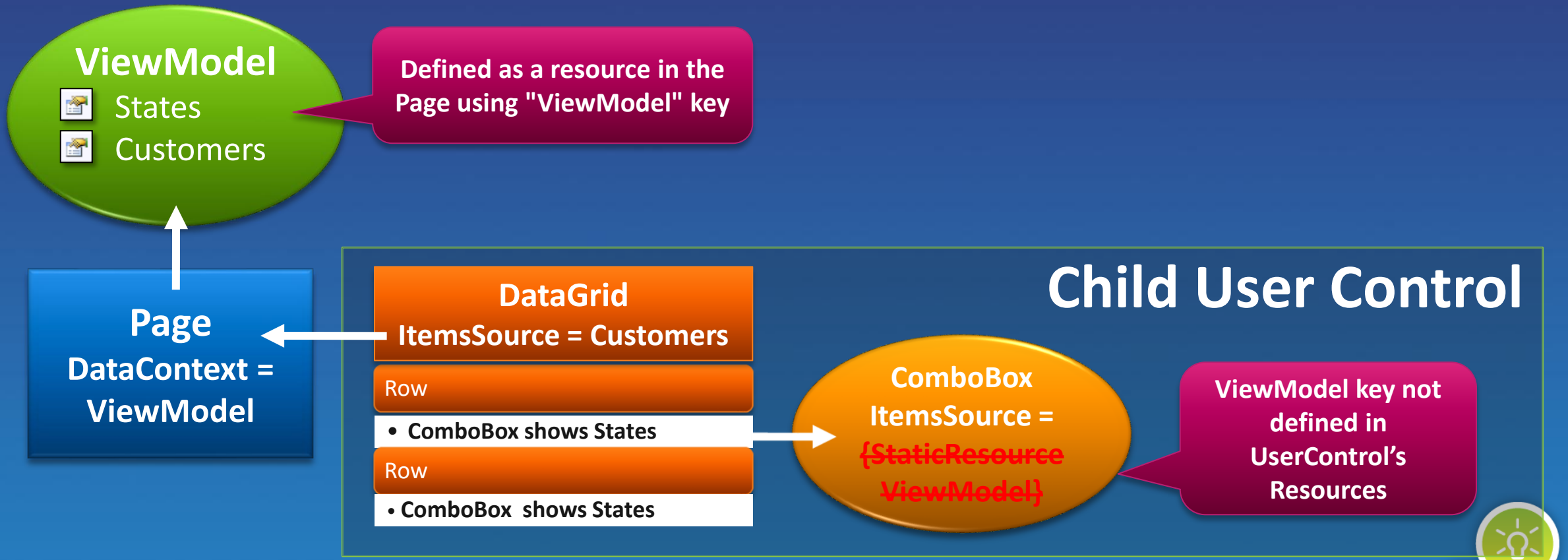
Data Binding in Nested Controls

Nested controls can access data stored in a StaticResource



Data Binding in Nested User Controls

Accessing the DataContext within child User Controls



Using a DataContextProxy

- DataContextProxy class simplifies data binding in nested User Controls:
  - Exposes the parent's DataContext object (that's defined as a StaticResource) to child User Controls and their children
  - Provides a DataSource Dependency Property
- Modeled after the ScriptManagerProxy from ASP.NET AJAX

DataContextProxy

DEMO

Lesson #3: Notify Users of Successes (and Failures)

- Silverlight provides a rich animation engine and media capabilities – Use them!
- Animations can be used to provide a visual indication of success or failure without getting in the way
- Minimize the number of times a user has to click on a MessageBox or ChildWindow
- Take advantage of media to play subtle sounds as operations succeed or fail

Using Animations and Media

- Animations can be used to show unobtrusive messages:

Success!

```
    <DoubleAnimationUsingKeyFrames Storyboard.TargetName="StatusCanvasTranslateTransform"
      Storyboard.TargetProperty="Y">
      <EasingDoubleKeyFrame KeyTime="00:00:00" Value="-66"/>
      <EasingDoubleKeyFrame KeyTime="00:00:00.6" Value="0" />
      <EasingDoubleKeyFrame KeyTime="00:00:03" Value="0"/>
      <EasingDoubleKeyFrame KeyTime="00:00:03.6" Value="-66"/>
    </DoubleAnimationUsingKeyFrames>
</Storyboard>
```

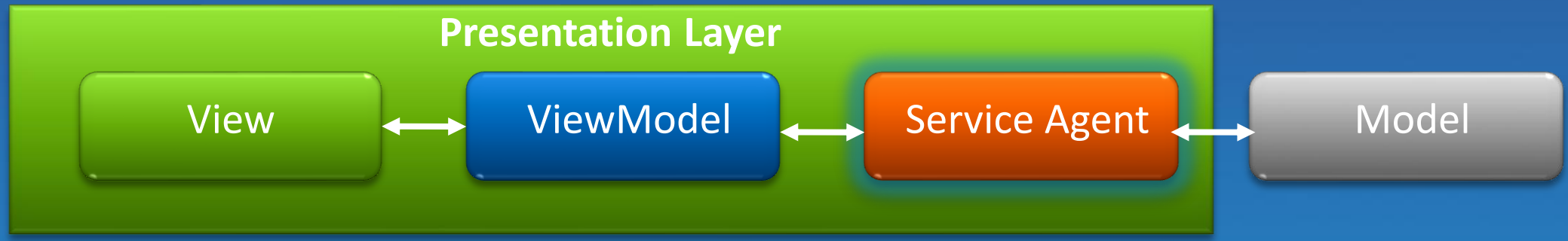
- MediaElement provides a simple way to play sounds

Failures & Success

DEMO

Lesson #4: Get an Agent – A Service Agent

- Make service calls from a Service Agent:
  - Allows for better code re-use
  - Allow ViewModel classes to stay focused and clean
  - Can minimize amount of wrapper code
  - Service Agent can implement an interface to allow for testing



Typical WCF Service Calls

```
public void GetJobs(EventHandler<GetJobsCompletedEventArgs> callback)
{
    IJobPlanService proxy = GetProxy();
    proxy.GetJobsCompleted += callback;
    proxy.GetJobsAsync();
}

public void GetEmployees(EventHandler<GetEmployeesCompletedEventArgs>
callback)
{
    IJobPlanService proxy = GetProxy();
    proxy.GetEmployeesCompleted += callback;
    proxy.GetEmployeesAsync();
}
```

Minimize WCF Service Call Code

```
//Method in ViewModel Class
public void GetAreasByJobID()
{
    MyServiceAgent.CallService<GetAreasCompletedEventArgs>
    (
          (s,e) => CurrentTimeSheetView.Areas = e.Result,
            TimeSheetJob.JobID
    );
}
```

The CallService<T> Method

```
Type _Type = typeof(JobPlanServiceClient);

public void CallService<T>(EventHandler<T> callback,
 params object[] parameters) where T : EventArgs {

   var proxy = new JobPlanServiceClient(_Binding,
  _EndPointAddress);
   string action = typeof(T).Name
  .Replace(CompletedEventargs, String.Empty);

   _Type.GetEvent(action + Completed)
  .AddEventHandler(proxy, callback);
   _Type.InvokeMember(action + Async,
   BindingFlags.InvokeMethod, null,
   proxy,parameters);
}
```

Service Agent

DEMO

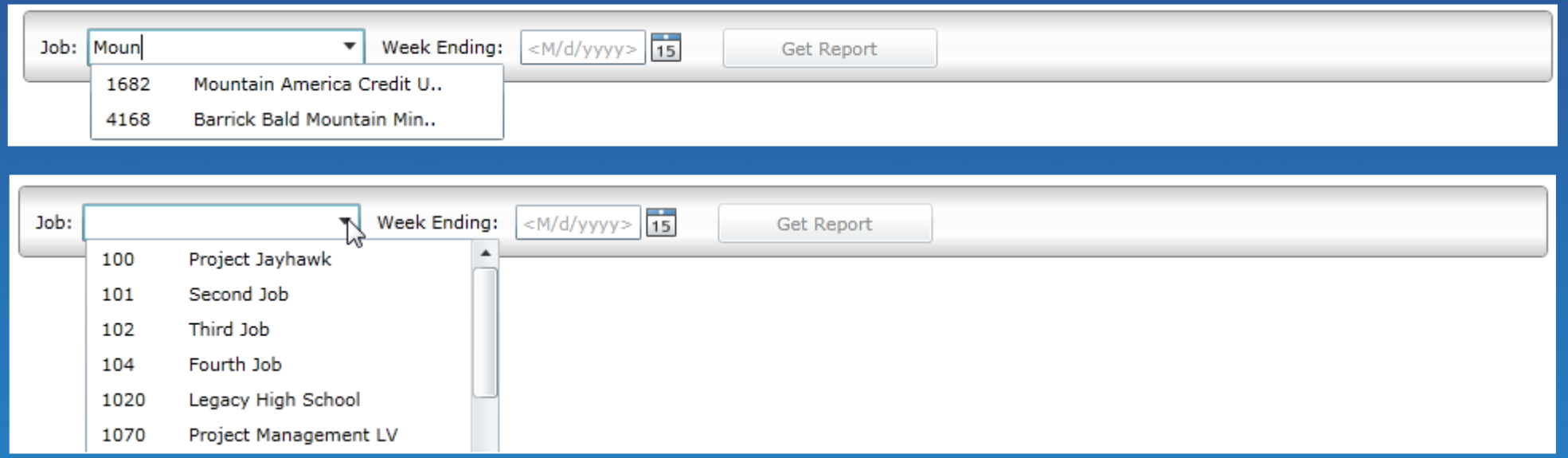
Lesson #5: Extend Existing Controls

- Extend built-in Silverlight controls and Toolkit controls rather than reinventing the wheel

- Examples of extended controls:
  - TextBox → FilteredTextBox
  - ComboBox extended to deal with primary keys better
  - AutoCompleteBox → AutoCompleteComboBox

Extending AutoCompleteBox

- Customer required filtering combined with ComboBox-like functionality
- AutoCompleteBox was extended to create an AutoCompleteComboBox for filtering and/or direct selection:



AutoCompleteComboBox

DEMO

Summary

- Best Practices:
  - Lesson #1: Pick a Pattern and Stick to It
  - Lesson #2: Data Binding and Nested Controls
  - Lesson #3: Notify Users of Successes (and failures)
  - Lesson #4: Get an Agent - A Service Agent
  - Lesson #5: Extend Existing Controls

Questions?

Blog

http://www.rudigrobler.net

Twitter

@rudigrobler

And don't forget to poken me

BARONE, BUDGE & DOMINICK
Business·Technology·Solutions

Resources

Microsoft tech·ed Online

Sessions On-Demand & Community

www.microsoft.com/teched

Microsoft | Learning

Microsoft Certification & Training Resources

www.microsoft.com/learning

Microsoft TechNet

Resources for IT Professionals

http://microsoft.com/technet

msdn

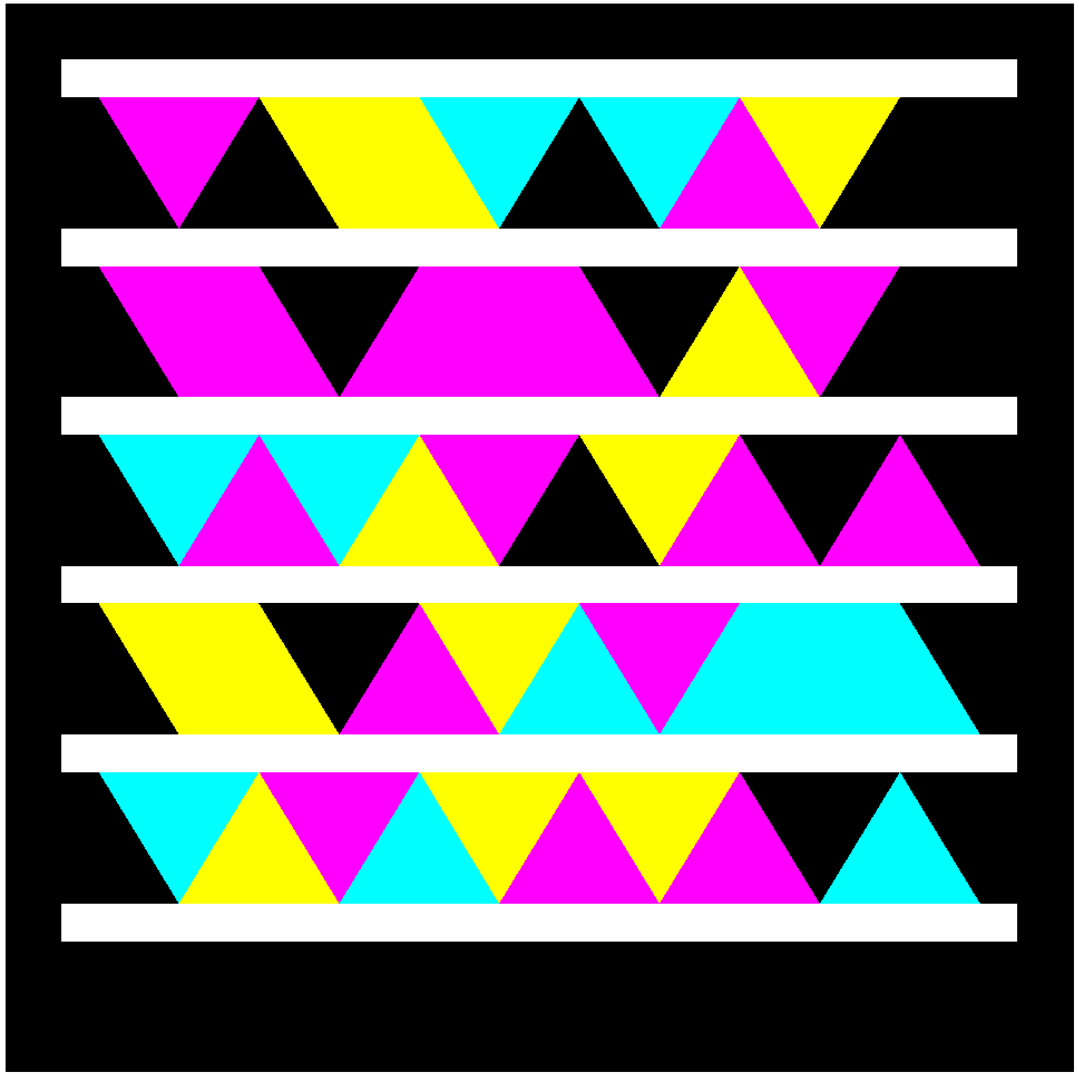
Resources for Developers

http://microsoft.com/msdn

Need more Information?

SMS [Your Name] and the word "Web" to 41491

Complete an evaluation via CommNet and Tag to win amazing prizes!



Microsoft®