

Porting the ToDo Listing Screen – Part 2

Derik Whittaker

Twitter: @derikwhittaker



Agenda

- **Handling Real Time Two way text binding**
 - Knockout valueUpdate vs XAML Behaviors
- **Filtering Client Side Data**
 - Knockout View Model vs XAML View Model
 - Knockout Click Binding vs ICommand
- **Advanced UI Styling**
 - Knockout Custom Bindings vs. Style Converter

Agenda

- **Handling Visible State**

- Knockout Visible vs Visibility Converter

- **Deleting Client Side Data**

- Ajax Posting to Web API end point

- **Computed Totals**

- Knockout Computed vs View Model Properties

Agenda

- Two Way, Real Time, Text Binding
- Client Side Filtering
- Advanced UI Styling
- Visibility State of UI elements
- Deleting Data
- Computed Totals

Two Way, real time Text Binding

localhost:8888/ToDo x


localhost:8888/ToDoHtml/

HTML for XAML Developer Template Home

Filter ToDo's Filter

● Call the cable company	03/31/2013	03/30/2013	Normal	Honey Do	Edit	Delete
● Make Vet Appointment for dog	04/01/2013	03/31/2013	Normal	Honey Do	Edit	Delete
● Order items off Amazon	04/02/2013		Normal	Personal	Edit	Delete
● Get the car washed	04/02/2013		Normal	Personal	Edit	Delete
● Date Night	04/04/2013	04/03/2013	High	Quality	Edit	Delete
● Fix the issues w/ the door	04/04/2013	04/03/2013	Normal	Honey Do	Edit	Delete
● Clean the grill	04/04/2013	04/03/2013	Normal	Personal	Edit	Delete
● Fix issues w/ the computer	03/27/2013	03/26/2013	High	Personal	Edit	Delete
● Pickup the dry cleaning	03/29/2013	03/28/2013	Normal	Honey Do	Edit	Delete
● Pick paint color for the walls	03/26/2013	03/27/2013	Normal	Honey Do		

Overdue 2 Active 7 Total 10



Two Way Text Binding

Silverlight Text Binding

```
<assets:WatermarkedTextBox Width="150" Margin="5" Watermark="Filter ToDo's"  
    Text="{Binding FilterText, Mode=TwoWay}" >  
</assets:WatermarkedTextBox>
```

Setup Binding to our
backing property



Have to explicitly set the
Mode to TwoWay



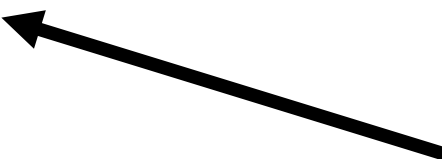
Knockout/Html Text Binding

```
<input type="text" class="no-margin" placeholder="Filter ToDo's"  
    data-bind="value: FilterText" />
```

Setup binding via
data-bind



No need to explicitly set
binding mode

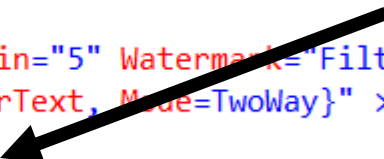


Real Time Binding Update

Silverlight Immediate Update

Need to implement a custom behavior


```
<assets:WatermarkedTextBox Width="150" Margin="5" Watermark="Filter ToDo's"
    Text="{Binding FilterText, Mode=TwoWay}" >
    <i:Interaction.Behaviors>
        <b:TextBoxUpdateOnChangeBehavior />
    </i:Interaction.Behaviors>
</assets:WatermarkedTextBox>
```



Knockout Immediate Update

Wire to the valueUpdate browser event

```
<input type="text" class="no-margin" placeholder="Filter ToDo's"
    data-bind="value: FilterText, valueUpdate: 'afterKeyDown'" />
```



Possible Events:

- | | | |
|--------------|---|--|
| keyup | ← | User releases a key |
| keypress | ← | Like keyup, but will repeat if key is held |
| afterKeyDown | ← | As user types – Best Choice for real-time |

Agenda

- Two Way, Real Time, Text Binding
- **Client Side Filtering**
- Advanced UI Styling
- Visibility State of UI elements
- Deleting Data
- Computed Totals

Filtering Client Side Data

Silverlight View Model

```
private void FilterToDo()  
{  
    var foundItems = _rawToDoItemsList.Where(x => x.Task.ToLower()  
                                              .Contains(FilterText.ToLower())).ToList();  
  
    _toDoItems.Clear();  
  
    foreach (var foundItem in foundItems)  
    {  
        _toDoItems.Add(foundItem);  
    }  
}
```

Use Linq to filter our data



Set the results to the
bound collection



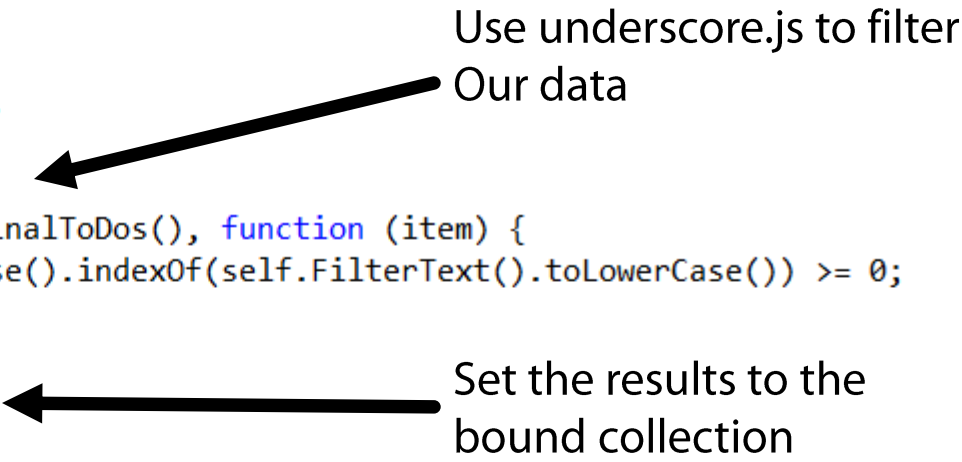
Filtering Client Side Data

Knockout View Model

```
filterList() {  
    var self = this;  
  
    if (self.FilterText().length == 0) {  
        self.Todos(self.OriginalTodos());  
    }  
    else {  
        var results = _.filter(self.OriginalTodos(), function (item) {  
            return item.Task().toLowerCase().indexOf(self.FilterText().toLowerCase()) >= 0;  
        });  
  
        self.Todos(results);  
    }  
}
```

Use underscore.js to filter
Our data

Set the results to the
bound collection



Agenda

- Two Way, Real Time, Text Binding
- Client Side Filtering
- **Advanced UI Styling**
- Visibility State of UI elements
- Deleting Data
- Computed Totals

Changing Styles w/ Custom Bindings

Changing Styles via Computed Observables

```
this.StatusStyle = ko.computed(() => {  
    return "circle status-" + this.Status().toLowerCase() + "-color"  
});
```

← Computed Observable

Changing Styles via Custom Binding

```
ko.bindingHandlers.strikeThroughCompleted = {  
    init: (element, valueAccessor) => {  
        var value = ko.utils.unwrapObservable(valueAccessor());  
        var addClass = value.toLowerCase() == 'completed'  
        $(element).toggleClass("striketthrough", addClass);  
    }  
};
```

← Add a new Binding to Knockout

← Change our underlying style via some logic

`data-bind="striketthroughCompleted: Status().Description":`

Agenda

- Two Way, Real Time, Text Binding
- Client Side Filtering
- Advanced UI Styling
- **Visibility State of UI elements**
- Deleting Data
- Computed Totals

Visibility State of UI Elements

Boolean to Visibility Converter

```
public class BooleanToVisibilityConverter : IValueConverter
{
    public Visibility TrueValue { get; set; }
    public Visibility FalseValue { get; set; }


    public BooleanToVisibilityConverter()...

    public object Convert(object value, Type targetType, object parameter,
        System.Globalization.CultureInfo culture)
    {
        if (value == null)
        {
            return Visibility.Collapsed;
        }

        return ((bool)value) ? TrueValue : FalseValue;
    }

    public object ConvertBack(object value, Type targetType, object parameter,
        System.Globalization.CultureInfo culture)...
```

Typical Boolean to
visibility converter



Visibility State of UI Elements

Visibility binding in Knockout

```
<input type="button" class="btn" value="Edit" style="width: 65px;"  
  data-bind="visible: Status().Description() != 'Completed'" />
```



Use the Visible binding



Statement being evaluated to
determine visibility.

This evaluates via the truthy/falsy
concepts in JavaScript

Agenda

- Two Way, Real Time, Text Binding
- Client Side Filtering
- Advanced UI Styling
- Visibility State of UI elements
- **Deleting Data**
- Computed Totals

Deleting Data

Making the call in Silverlight

```
public void DeleteToDo( int idToDelete, Action<bool> callbackAction)
{
    var url = string.Format("http://localhost:8888/ToDoServices/api/ToDo/Delete/{0}", idToDelete);
    var client = new RestClient(url);
    var request = new RestRequest(Method.DELETE);

    client.ExecuteAsync(request, (response, handle) =>
    {
        if (response.StatusCode == HttpStatusCode.OK ||
            response.StatusCode == HttpStatusCode.NoContent)
        {
            DispatcherHelper.CheckBeginInvokeOnUI(() => callbackAction.Invoke(true));
        }
        else
        {
            callbackAction.Invoke(false);
        }
    });
}
```

Service to perform the delete

Client to make the service call


Evaluate the response and take some action

Deleting Data


Making the call in Typescript

```
deleteToDo(id: number) {  
  var self = this;  
  var url = "http://localhost:8888/ToDoServices/api/ToDo/Delete/" + id;  
  
  $.ajax({  
    url: url,  
    type: 'DELETE',  
    success: (data) => {  
      self.fetchToDoItems();  
    },  
    error: (XMLHttpRequest, textStatus, errorThrown) => {  
  
    }  
  });  
}
```

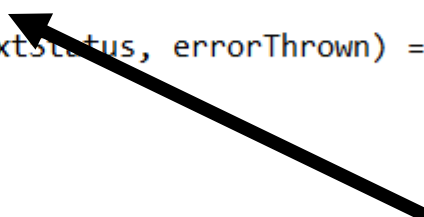
Service to perform the delete



Make the call via jQuery and Ajax



Evaluate the response and take some action



Agenda

- Two Way, Real Time, Text Binding
- Client Side Filtering
- Advanced UI Styling
- Visibility State of UI elements
- Deleting Data
- **Computed Totals**

Computed Totals

HTML for XAML Developer Template

Home

Filter

● Call the cable company	03/31/2013	03/30/2013	Normal	Honey Do	Edit	Delete
● Make Vet Appointment for dog	04/01/2013	03/31/2013	Normal	Honey Do	Edit	Delete
● Order items off Amazon	04/02/2013		Normal	Personal	Edit	Delete
● Get the car washed	04/02/2013		Normal	Personal	Edit	Delete
● Date Night	04/04/2013	04/03/2013	High	Quality	Edit	Delete
● Fix the issues w/ the door	04/04/2013	04/03/2013	Normal	Honey Do	Edit	Delete
● Clean the grill	04/04/2013	04/03/2013	Normal	Personal	Edit	Delete
● Fix issues w/ the computer	03/27/2013	03/26/2013	High	Personal	Edit	Delete
● Pickup the dry cleaning	03/29/2013	03/28/2013	Normal	Honey Do	Edit	Delete
● Pick paint color for the walls	03/26/2013	03/27/2013	Normal	Honey Do		




Overdue **2** Active **7** Total **10**

Computed Totals

Computing totals in Silverlight View Model


```
RaisePropertyChanged(() => ActiveCount);  
RaisePropertyChanged(() => OverdueCount);  
RaisePropertyChanged(() => TotalCount);
```

Raise Notifications to the UI to rebind the properties



```
public int ActiveCount  
{  
    get  
    {  
        return ToDoItems.Count(x => x.Status.Id == (int)State.Active);  
    }  
}
```

Backing field to calculate the summary values




Computed Totals

Computing totals in Typescript View Model


```
public OverdueCount: KnockoutComputed;  
public ActiveCount: KnockoutComputed;  
public TotalCount: KnockoutComputed;
```

Declare our Knockout
computed fields




```
this.ActiveCount = ko.computed(() => {  
    var count = _.filter(this.ToDos(), (item) => {  
        return item.Status() == "Active"  
    }).length;  
    return count;  
});
```

Computed field using
underscore JS to
summarize the value



Will recalculate when the
ToDo's collection is
modified



Summery

- Learned how to setup Two Way, Real Time, Text Binding
- Learned how to do Client Side Filtering
- Learned another way to apply styles to your UI elements
- Learned how to toggle visibility state of UI elements
- Learned how to make http posts to Deleting Data
- Learned how to create Computed Totals