

DataBinding - a look into the generated code

Before pre-processing

Product.as

```
package valueObject
{
    [Bindable]
    public class Product
    {
        public var productName:String;
    }
}
```

DataBindingSample.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
layout="absolute">

    <mx:Script>
        <![CDATA[
            import valueObject.Product;

            [Bindable]
            private var product : Product;
        ]]>
    </mx:Script>

    <mx:Label id="productLabel" text="{product.productName}"/>
</mx:Application>
```

After pre-processing

_Product-binding-generated.as

_DataBindingSample-generated.as

_DataBindingSampleWatcherSetupUtil.as

_Product-binding-generated.as

```
class BindableProperty implements flash.events.IEventDispatcher
{
    [Bindable(event="propertyChange")]
    public function get productName():String
    {
        return this._1491817446productName;
    }

    public function set productName(value:String):void
    {
        var oldValue:Object = this._1491817446productName;
        if (oldValue !== value)
        {
            this._1491817446productName = value;
            this.dispatchEvent(mx.events.PropertyChangeEvent.createUpdateEvent(
                this, "productName", oldValue, value));
        }
    }

    //      IEventDispatcher implementation
    //
    private var _bindingEventDispatcher:flash.events.EventDispatcher =
        new flash.events.EventDispatcher(flash.events.IEventDispatcher(this));

    //...
}
```

DataBindingSample-generated.as

```
override public function initialize():void
{
    mx_internal::setDocumentDescriptor(_documentDescriptor_);
    var bindings:Array = _DataBindingSample_bindingsSetup();
    var watchers:Array = [];

    var target:DataBindingSample = this;

    if (_watcherSetupUtil == null)
    {
        var watcherSetupUtilClass:Object = getDefinitionByName("_DataBindingSamplewatcherSetupUtil");
        watcherSetupUtilClass["init"](null);
    }

    _watcherSetupUtil.setup(this,
        function(propertyName:String):* { return target[propertyName]; },
        bindings,
        watchers);

    for (var i:uint = 0; i < bindings.length; i++)
    {
        Binding(bindings[i]).execute();
    }

    mx_internal::_bindings = mx_internal::_bindings.concat(bindings);
    mx_internal::_watchers = mx_internal::_watchers.concat(watchers);

    super.initialize();
}
```

DataBindingSample-generated.as

```
private function _DataBindingSample_bindingsSetup():Array
{
    var result:Array = [];
    var binding:Binding;

    binding = new mx.binding.Binding(this,
        function():String
        {
            var result:* = (product.productName);
            var stringResult:String = (result == undefined ? null : String(result));
            return stringResult;
        },
        function(_sourceFunctionReturnValue:String):void
        {
            productLabel.text = _sourceFunctionReturnValue;
        },
        "productLabel.text");
    result[0] = binding;
    return result;
}
```

_DataBindingSample-generated.as

Constructing a mx.binding.Binding

- instances of this class accept a document, *srcFunc*, *destFunc* and a *destString* as parameters
- *srcFunc* is the function that returns the value to use in this binding
- *destFunc* is the function that will take that value and assign it to the destination
- *destString* is the destination represented as a String

```
private function _DataBindingSample_binding_1() {
    var result:Array = [];
    var binding:Binding;

    binding = new mx.binding.Binding(this,
        function @:String {
            result = (product.productName);
        },
        function (turnvalue:String):void {
            product.label.text = turnvalue;
        },
        "productLabel");
    result.push(binding);
    return result;
}
```

DataBindingSample-generated.as

```
override public function initialize():void
{
    mx_internal::setDocumentDescriptor(_documentDescriptor_);

    var bindings:Array = _DataBindingSample_bindingsSetup();
    var watchers:Array = [];

    var target:DataBindingSample = this;

    if (_watcherSetupUtil == null)
    {
        var watcherSetupUtilClass:Object = getDefinitionByName("_DataBindingSamplewatcherSetupUtil");
        watcherSetupUtilClass["init"](null);
    }

    _watcherSetupUtil.setup(this,
        function(propertyName:String):* { return target[propertyName]; },
        bindings,
        watchers);

    for (var i:uint = 0; i < bindings.length; i++)
    {
        Binding(bindings[i]).execute();
    }

    mx_internal::_bindings = mx_internal::_bindings.concat(bindings);
    mx_internal::_watchers = mx_internal::_watchers.concat(watchers);

    super.initialize();
}
```

_DataBindingSampleWatcherSetupUtil.as

```
public function setup(target:Object,  
                    propertyGetter:Function,  
                    bindings:Array,  
                    watchers:Array):void  
{  
    watchers[0] = new mx.binding.PropertyWatcher(  
        "product",  
        {propertyChange: true},  
        [[bindings[0]],  
        propertyGetter  
    );  
  
    watchers[1] = new mx.binding.PropertyWatcher(  
        "productName",  
        {propertyChange: true},  
        [bindings[0]],  
        null  
    );  
  
    watchers[0].updateParent(target);  
    watchers[0].addChild(watchers[1]);  
}
```

Property watchers are responsible for noticing a change and notify-ing the *Binding-objects* that they should execute

DataBindingSampleWatcherSetupUtil.as

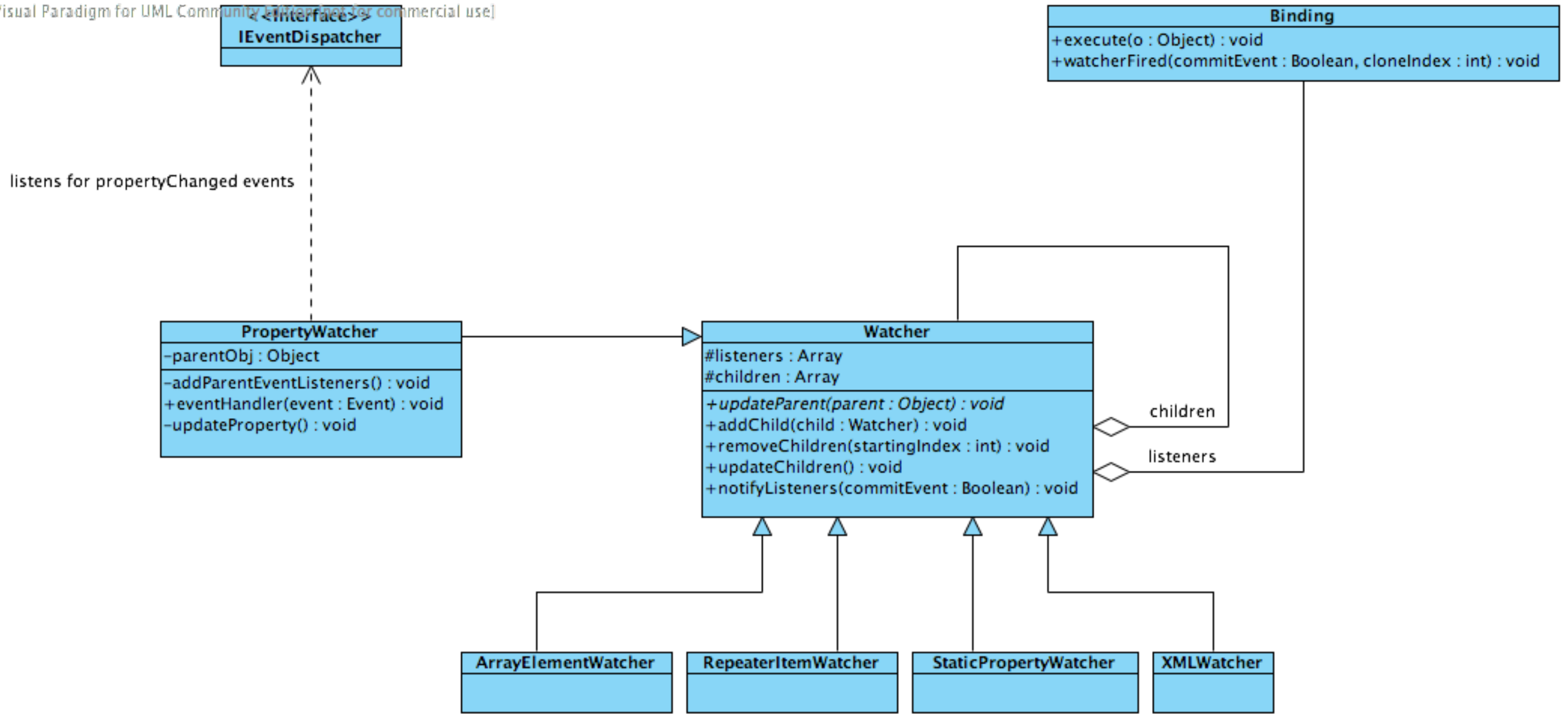
```
public function setup(target:Object,  
                    propertyGetter:Function,  
                    bindings:Array,  
                    watchers:Array):void  
{  
    watchers[0] = new mx.binding.PropertyWatcher(  
        "product",  
        {propertyChange: true},  
        [[bindings[0]],  
        propertyGetter  
    );  
  
    watchers[1] = new mx.binding.PropertyWatcher(  
        "productName",  
        {propertyChange: true},  
        [bindings[0]],  
        null  
    );  
  
    watchers[0].updateParent(target);  
    watchers[0].addChild(watchers[1]);  
}
```

To configure a

PropertyWatcher you pass:

- *propertyName*
- an object that indicates what type of events are dispatched on property change
- array of listeners - any *Binding* instances created for the property
- *propertyGetter* - an anonymous function that returns the value of the property value

Connecting the dots...



Summary

- the expansion of *[Bindable]*
- bindings setup and the *mx.binding.Binding* class
- setup all *PropertyWatch-ers*
- UML diagram of the binding mechanism