Business Rules: RowBuilder Attribute

Latest update of Data Aquarium Framework introduces a new and much improved method of extending framework with custom business logic. You can take full advantage of declarative user interface of data controllers custom form templates, while having a significant control over data processing.

ABOUT BUSINESS RULES

ASP.NET development model makes it very easy for developers to mix user interface code and business logic code. It is just too tempting to write a few lines of a business rule that will be executed in response to a user interface control event. This leads to expensive maintenance.

We have put a significant effort into preventing such code blending from happening in applications built with Data Aquarium Framework. The first step was to introduce custom action handlers and data filters.

The new release builds on top of these two features and uses attribute-based approach to business rule development.

We will start review of business rules with *RowBuilder* attribute.

SAMPLE WEB APPLICATION

Generate a sample Data Aquarium application from *Northwind* database. Leave *MyCompany* as a default namespace. Make sure to request generation of business logic layer. We will use generated business objects to enhance our business rules whenever we need to access data. Note that if you generate your application with Membership support enabled then you will have to sign into your application in order to see any data presentation.

Open generated project in *Visual Studio 2008* or *Visual Web Developer 2008* and add a new class *Class1* to *~/App_Code* folder of your application. Our new class will inherit its base functionality from *BusinessRules* class of *MyCompany.Data* namespace.

C#:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using MyCompany.Data;

public class Class1: BusinessRules
{
    public Class1()
    {
    }
}
```

VB:

```
Imports Microsoft.VisualBasic
Imports MyCompany.Data

Public Class Class1
    Inherits BusinessRules

End Class
```

Open file *~/Controllers/Employees.xml* and add attribute handler to *dataController* element.

```
<dataController name="Employees" conflictDetection="overwriteChanges"
    label="Employees" xmlns="urn:schemas-codeontime-com:data-aquarium"
    handler="Class1">
```

This will hook your business rules to *Employees* data controller. Your business rules will be universally applied whenever a reference to a data controller is made. You business rules will be invoked when your application is utilizing *DataViewExtender* or *ControllerDataSource*, when you export data or create new employees in data lookups.

The same set of business rules can be applied to multiple data controllers.

BUILDING NEW ROWS

The first rule will be assigning employee ID of company's CEO to *ReportsTo* field of any new employee record.

Add the following method to *Class1*.

C#:

```
[RowBuilder("Employees", "createForm1", RowKind.New)]
protected void PrepareNewEmployeeRow()
{
    using (SqlText findCEO = new SqlText(
        "select EmployeeID, LastName from Employees where ReportsTo is
null"))
    {
        if (findCEO.Read())
        {
            UpdateFieldValue("ReportsTo", findCEO["EmployeeID"]);
            UpdateFieldValue("ReportsToLastName", findCEO["LastName"]);
        }
    }
}
```

VB:

```
<RowBuilder("Employees", "createForm1", RowKind.New)> _
Protected Sub PrepareNewEmployeeRow()
```

```
    Using findCEO As SqlText = New SqlText( _
        "select EmployeeID, LastName from Employees where ReportsTo is null")
        If findCEO.Read() Then
            UpdateFieldValue("ReportsTo", findCEO("EmployeeID"))
            UpdateFieldValue("ReportsToLastName", findCEO("LastName"))
        End If
    End Using
End Sub
```

The protection level of this method is only important if you are planning to create hierarchies of business rule classes.

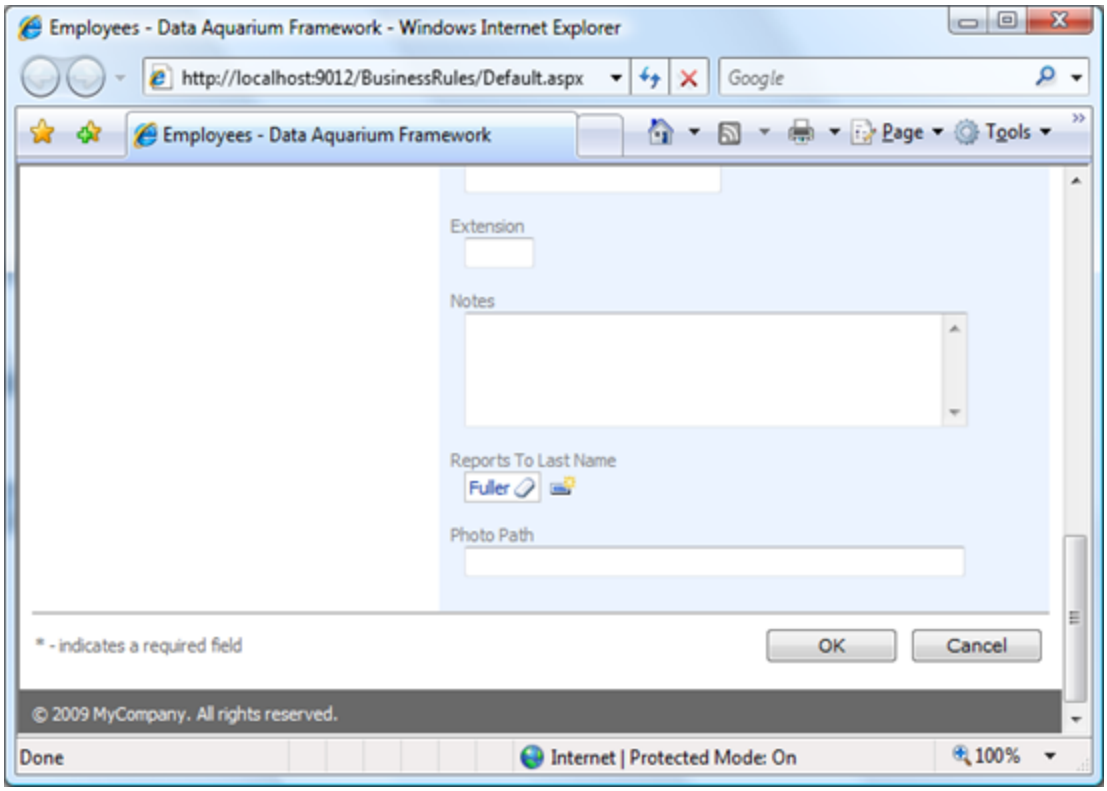
The name of the method plays no role at all. Name your business rules methods to reflect their purpose.

The method is automatically called whenever a new row is about to be returned to user interface components by *Employees* data controller. Typically this will happen when end user is creating a new record. There is also a restriction on the presentation view that will ensure method execution when *createForm1* is presenting data. You can apply multiple *RowBuilder* attributes to the same method if the same business logic is executed by other controllers and/or views.

Use method *UpdateFieldValue* to assign any default values to fields of a new row.

Class *SqlText* is a utility provided with Data Aquarium Framework to give you a simple and efficient way of querying a database.

Run the sample application and start creating a new employee record. Scroll to the bottom of the screen to see the result of our effort.



Next post will show how to manipulate fields of existing rows and demonstrate new look item style *CheckBoxList*.

The new lookup item style is featured in Membership user manager. Select any user and start editing a record. A menu of check boxes is presented to help you select user roles.

Code OnTime LLC

http://www.codeontime.com