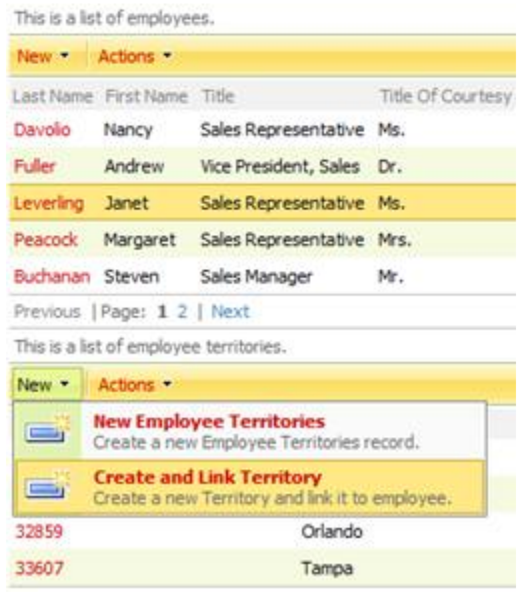
# ExternalFilter And Modal Views

New *ExternalFilter* property is available to business rules developers and allows implementing code that is aware of the external master-detail relationships without mixing business logic and user interface code and markup.



This is a list of employees.

New ▾ Actions ▾

| Last Name | First Name | Title | Title Of Courtesy |
|---|---|---|---|
| Davolio | Nancy | Sales Representative | Ms. |
| Fuller | Andrew | Vice President, Sales | Dr. |
| Leverling | Janet | Sales Representative | Ms. |
| Peacock | Margaret | Sales Representative | Mrs. |
| Buchanan | Steven | Sales Manager | Mr. |

Previous | Page: 1 2 | Next

This is a list of employee territories.

New ▾ Actions ▾

**New Employee Territories**
Create a new Employee Territories record.

**Create and Link Territory**
Create a new Territory and link it to employee.

| | |
|---|---|
| 32859 | Orlando |
| 33607 | Tampa |

## SAMPLE PROJECT

You can see this sample live at http://dev.codeontime.com/demo/externalfilter. The source code is available at http://dev.codeontime.com/demo/externalfilter/Source.zip.

Generate a Data Aquarium application from *Northwind* database with business objects enabled and add *~/EmployeeTerritories.aspx* page to the root of the web site created by Code OnTime Generator.

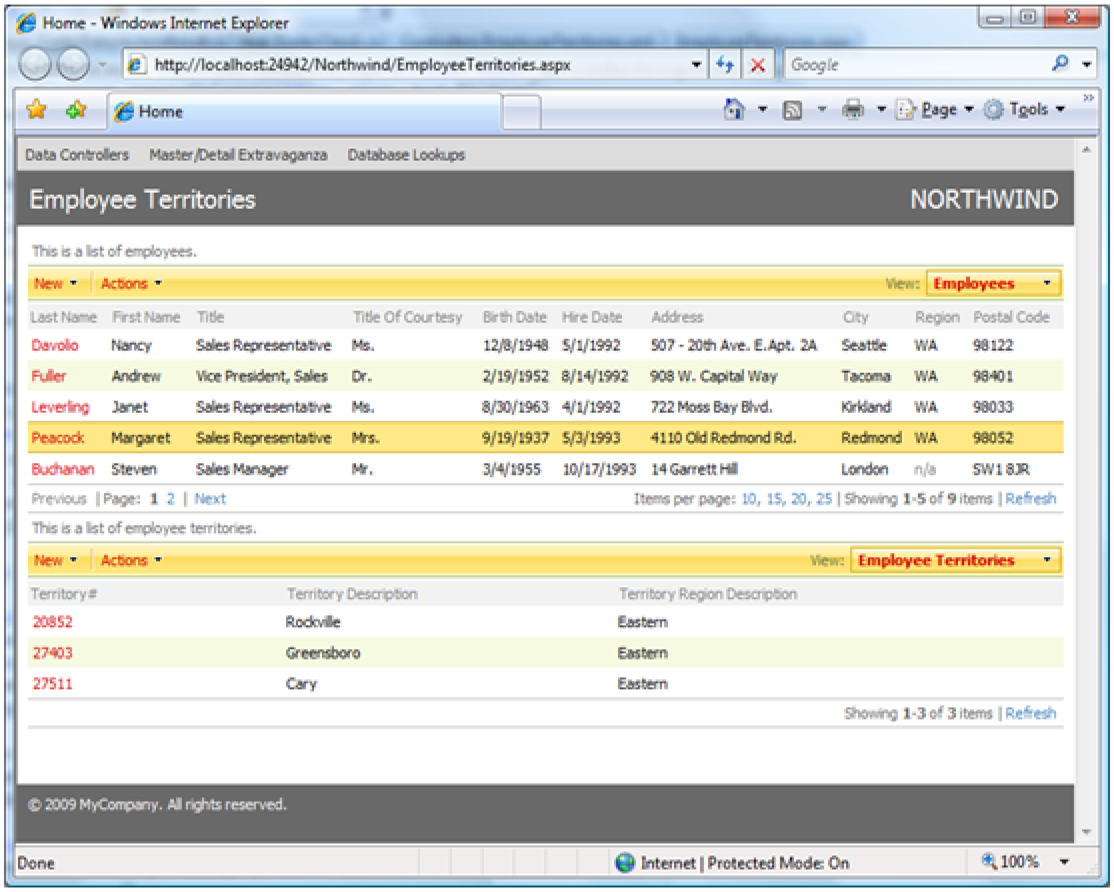
Open *~/Controllers/EmployeeTerritories.xml* and create its copy named *~/Controllers/MyEmployeeTerritories.xml*. Change view *grid1* to reveal the territory code and add territory description as a separate field.

```
<view id="grid1" type="Grid" commandId="command1" label="Employee
Territories">
    <headerText>This is a list of employee territories. </headerText>
    <dataFields>
        <dataField fieldName="EmployeeID"/>
        <dataField fieldName="TerritoryID" />
        <dataField fieldName="TerritoryTerritoryDescription"/>
        <dataField fieldName="TerritoryRegionRegionDescription"/>
    </dataFields>
</view>
```

Modify *EmployeeTerritories.aspx* as shown in this snippet.

```
<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.master"
    AutoEventWireup="true" CodeFile="EmployeeTerritories.aspx.cs"
    Inherits="EmployeeTerritories" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="Header1Placeholder"
runat="Server">
    Employee Territories
</asp:Content>
<asp:Content ID="Content3" ContentPlaceHolderID="Header2Placeholder"
runat="Server">
    Northwind
</asp:Content>
<asp:Content ID="Content4" ContentPlaceHolderID="BodyPlaceholder"
runat="Server">
    <div id="EmployeesDiv" runat="server">
    </div>
    <aquarium:DataViewExtender ID="EmployeesExtender" runat="server"
        Controller="Employees"
        PageSize="5" TargetControlID="EmployeesDiv" />
    <div id="EmployeeTerritoriesDiv" runat="server" />
    <aquarium:DataViewExtender ID="EmployeeTerritoriesExtender"
runat="server"
        Controller="MyEmployeeTerritories"
        TargetControlID="EmployeeTerritoriesDiv" FilterFields="EmployeeID"
        FilterSource="EmployeesExtender"
        PageSize="20" />
</asp:Content>
```

Open the page in the browser. The following user interface will be presented.


Home - Windows Internet Explorer
http://localhost:24942/Northwind/EmployeeTerritories.aspx
Google
Home
Page
Tools
Data Controllers
Master/Detail Extravaganza
Database Lookups

Employee Territories

NORTHWIND

This is a list of employees.

New ▾ | Actions ▾ | View: Employees

| Last Name | First Name | Title | Title Of Courtesy | Birth Date | Hire Date | Address | City | Region | Postal Code |
|---|---|---|---|---|---|---|---|---|---|
| Davolio | Nancy | Sales Representative | Ms. | 12/8/1948 | 5/1/1992 | 507 - 20th Ave. E.Apt. 2A | Seattle | WA | 98122 |
| Fuller | Andrew | Vice President, Sales | Dr. | 2/19/1952 | 8/14/1992 | 908 W. Capital Way | Tacoma | WA | 98401 |
| Leverling | Janet | Sales Representative | Ms. | 8/30/1963 | 4/1/1992 | 722 Moss Bay Blvd. | Kirkland | WA | 98033 |
| Peacock | Margaret | Sales Representative | Mrs. | 9/19/1937 | 5/3/1993 | 4110 Old Redmond Rd. | Redmond | WA | 98052 |
| Buchanan | Steven | Sales Manager | Mr. | 3/4/1955 | 10/17/1993 | 14 Garrett Hill | London | n/a | SW1 8JR |

Previous | Page: 1 2 | Next    Items per page: 10, 15, 20, 25 | Showing 1-5 of 9 items | Refresh

This is a list of employee territories.

New ▾ | Actions ▾ | View: Employee Territories

| Territory# | Territory Description | Territory Region Description |
|---|---|---|
| 20852 | Rockville | Eastern |
| 27403 | Greensboro | Eastern |
| 27511 | Cary | Eastern |

Showing 1-3 of 3 items | Refresh

© 2009 MyCompany. All rights reserved.

TASK

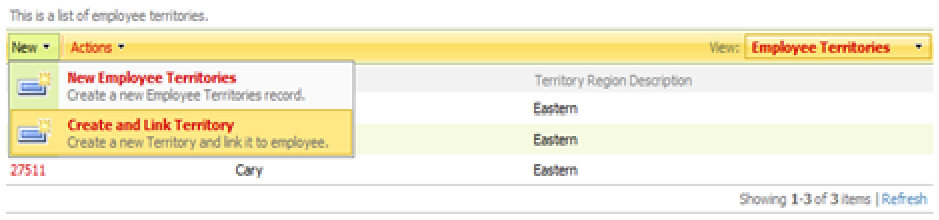
Users can quickly associate selected employee and territories by clicking *New* | *New Employee Territories* option on the action bar of *Employee Territories* view.

We will add a new option to *New* menu to allow simultaneous creation of a new territory and association of this territory with the selected employee. We will be using modal dialog and *ExternalFilter* property for this task.

## IMPLEMENTATION

Add the new action to *MyEmployeeTerritories.xml*.

```
<actionGroup scope="ActionBar" headerText="New">
    <action commandName="New" commandArgument="createForm1"
        headerText="New Employee Territories"
        description="Create a new Employee Territories record." />
    <action commandName="Custom" commandArgument="CreateAndLinkTerritory"
        headerText="Create and Link Territory"
        description="Create a new Territory and link it to employee."
        cssClass="NewLargeIcon"/>
</actionGroup>
```


This is a list of employee territories.

New ▾ Actions ▾ View: Employee Territories

New Employee Territories
Create a new Employee Territories record.

Create and Link Territory
Create a new Territory and link it to employee.

| | | Territory Region Description |
|---|---|---|
| | | Eastern |
| | | Eastern |
| 27511 | Cary | Eastern |

Showing 1-3 of 3 items | Refresh

Modify *command1* and add new *command2* as follows.

```
        <command id="command1" type="Text">
            <text>
                <![CDATA[
select
    "EmployeeTerritories"."EmployeeID" "EmployeeID"
    ,"Employee"."LastName" "EmployeeLastName"
    ,"EmployeeTerritories"."TerritoryID" "TerritoryID"
    ,"Territory"."TerritoryDescription" "TerritoryTerritoryDescription"
    ,"TerritoryRegion"."RegionDescription" "TerritoryRegionRegionDescription"
    ,null "TerritoryCode"
    ,null "TerritoryDescription"
    ,null "RegionID"
from "dbo"."EmployeeTerritories" "EmployeeTerritories"
    left join "dbo"."Employees" "Employee" on
"EmployeeTerritories"."EmployeeID" = "Employee"."EmployeeID"
    left join "dbo"."Territories" "Territory" on
"EmployeeTerritories"."TerritoryID" = "Territory"."TerritoryID"
    left join "dbo"."Region" "TerritoryRegion" on "Territory"."RegionID" =
"TerritoryRegion"."RegionID"
]]>
            </text>
        </command>
        <command id="command2" type="Text">
            <text>
                <![CDATA[
select
    null "EmployeeID"
    ,null "EmployeeLastName"
    ,null "TerritoryID"
    ,null "TerritoryTerritoryDescription"
    ,null "TerritoryRegionRegionDescription"
    ,null "TerritoryCode"
    ,null "TerritoryDescription"
    ,null "RegionID"

]]>
            </text>
        </command>
    </commands>
    <fields>
```

Notice that *command1* was enhanced with three additional fields *TerritoryCode, TerritoryDescription*, and *RegionID* that are being selected as empty values. Command

*command2* mirrors all fields of *command1* with one exception - all fields are returned as *NULL* values. The second command returns a single row of empty values.

Let's add definitions of the new fields to the *fields* element of the data controller.

```
<!-- new fields to support "createTerritory" view -->
<field name="TerritoryCode" type="String" allowNulls="false" label="Code"/>
<field name="TerritoryDescription" type="String" allowNulls="false"
label="Description"/>
<field name="RegionID" type="Int32" allowNulls="false" label="Region">
    <items style="ListBox" dataController="Region" />
</field>
```

Finally, we will add *createTerritory* view. This view will be displayed in response to a selection of *Create and Link Territory* action bar menu option and is relying on *command2* for its data.

```
<view id="createTerritory" type="Form" commandId="command2"
    label="Create New Territory">
    <headerText>Please fill this form and click OK button to create
        a new territory and link it to the employee. Click Cancel
        to return to the previous screen.</headerText>
    <categories>
        <category headerText="Employee Information">
            <description>New territory will be linked to this
employee.</description>
            <dataFields>
                <dataField fieldName="EmployeeLastName"/>
            </dataFields>
        </category>
        <category headerText="Territory">
            <description><![CDATA[
                Please enter the territory code, territory description, and
                select a territory region. <br/><br/>You can use zip code
                of the territory as code and city or county name as
                territory description.]]></description>
            <dataFields>
                <dataField fieldName="TerritoryCode" columns="5"/>
                <dataField fieldName="TerritoryDescription" />
                <dataField fieldName="RegionID" />
            </dataFields>
        </category>
    </categories>
</view>
```

Now we are ready to write some code. Data Aquarium Framework promotes business logic reusability. Custom code is implemented in independent classes that are hooked to data controllers via *@handler* attribute.

Create a new class *Class1* and link the class to data controller.

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

Enter the following code in the class definition.

C#

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

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

    [ControllerAction("MyEmployeeTerritories",
        "Custom", "CreateAndLinkTerritory")]
    protected void NewTerritory()
    {
        if (Arguments.ExternalFilter.Length > 0
            && Arguments.ExternalFilter[0].Value != null)
        {
            Employees emp = Employees.SelectSingle(
                Convert.ToInt32(Arguments.ExternalFilter[0].Value));
            Context.Session["Employee"] = emp;
            Context.Session["ControllerID"] = Arguments.ContextKey;
            Result.ShowModal("MyEmployeeTerritories", "createTerritory",
                "New", "createTerritory");
        }

        else
            Result.ShowAlert("Please select an employee.");
    }

    [RowBuilder("MyEmployeeTerritories",
        "createTerritory", RowKind.New)]
    protected void CreateTerritoryNewRow()
    {
        Employees emp = (Employees)Context.Session["Employee"];
        UpdateFieldValue("EmployeeLastName", emp.LastName);
    }
}
```

## VB

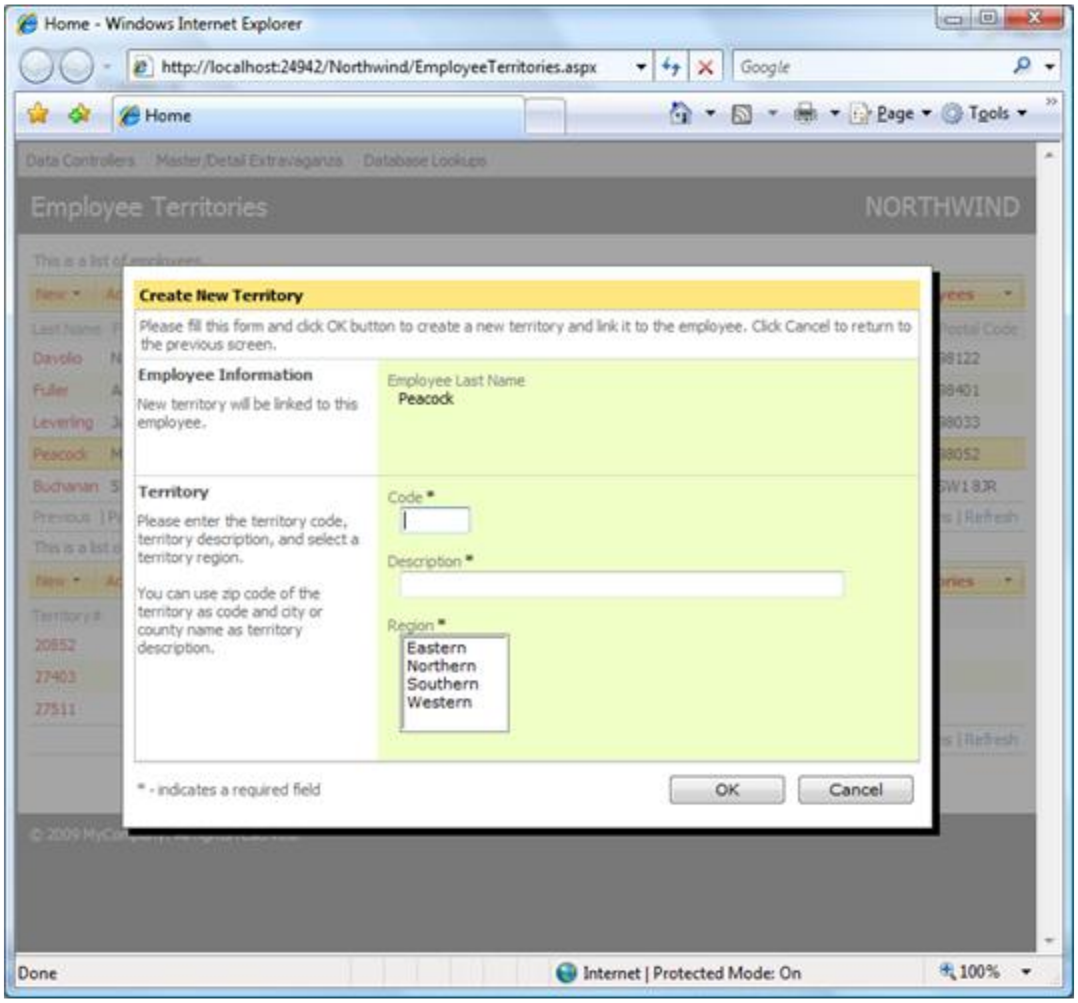
```
Imports Microsoft.VisualBasic
Imports MyCompany.Data
Imports MyCompany.Data.Objects

Public Class Class1
    Inherits BusinessRules

    <ControllerAction("MyEmployeeTerritories", _
        "Custom", "CreateAndLinkTerritory")> _
    Protected Sub NewTerritory()
        If (Arguments.ExternalFilter.Length > 0 _
                AndAlso Not Arguments.ExternalFilter(0).Value Is Nothing)
Then
            Dim emp As Employees = Employees.SelectSingle( _
                Convert.ToInt32(Arguments.ExternalFilter(0).Value))
            Context.Session("Employee") = emp
            Context.Session("ControllerID") = Arguments.ContextKey
            Result.ShowModal("MyEmployeeTerritories", "createTerritory", _
                             "New", "createTerritory")
        Else
            Result.ShowAlert("Please select an employee.")
        End If
    End Sub

    <RowBuilder("MyEmployeeTerritories", _
                "createTerritory", RowKind.New)> _
    Protected Sub CreateTerritoryNewRow()
        Dim emp As Employees = CType(Context.Session("Employee"), Employees)
        UpdateFieldValue("EmployeeLastName", emp.LastName)
    End Sub
End Class
```

Class *Class1* is inherited from *BusinessRules* base. Business rules are recognized by the framework. Business rules class members are treated as serving a specific purpose when adorned with certain attributes.

Attribute *ControllerAction* on method *NewTerritory* will instruct the framework to invoke this method whenever a custom command with argument *CreateAndLinkTerritory* is invoked in a client-side JavaScript component *Web.DataView*.

Business rules property *Arguments* gives access to all information about the incoming command request. Its own new property *ExternalFilter* allows inspecting values of a master-detail link when a *Web.DataView* component is linked as a child in such a relationship. At the top of the article we have configured *EmloyeeTerritoriesExtender* component to be filtered by *EmployeeID* field. The extender injects a JavaScript snippet into the page to create an instance of *Web.DataView* with the appropriate filter.

Method *NewTerritory* verifies if an employee is selected and then stores the selected employee information in the web request session and displays a view *createTerritory* in a modal mode. The startup command for the view is *New* with an argument that points to the view itself. This will cause the view to be displayed in the "new record" mode.


Home - Windows Internet Explorer
http://localhost:24942/Northwind/EmployeeTerritories.aspx
Google
Home
Page
Tools
Data Controllers
Master/Detail Extravaganza
Database Lookups
Employee Territories
NORTHWIND
Done
Internet | Protected Mode: On
100%

**Create New Territory**

Please fill this form and click OK button to create a new territory and link it to the employee. Click Cancel to return to the previous screen.

**Employee Information**

New territory will be linked to this employee.

Employee Last Name
Peacock

**Territory**

Please enter the territory code, territory description, and select a territory region.

You can use zip code of the territory as code and city or county name as territory description.

Code *

Description *

Region *
Eastern
Northern
Southern
Western

* - indicates a required field

OK Cancel

In the screen shot you can see this in action.

Note that employee name corresponds to the name of the employee selected in the list of employees. The name is provided by method *CreateTerritoryNewRow* adorned with an attribute *RowBuilder*. This method retrieves selected employee from the session and updates the row returned with employee's last name just before the row is returned to the client.

If you set a break point in method *CreateTerritoryNewRow* then the debugger will stop there twice. It happens for the first time when view is initialized for modal presentation. Then the startup command is executed to switch the view in the "new record" mode, which causes the second invocation. The framework does not distinguish views to be serving any specific purpose and assumes the "new record" mode only when the *New* command has been executed.

The following code will create a new territory and link it to the selected employee by create a new record in *EmployeeTerritories* table. This code will execute when user clicks the *OK* button.

C#

```
[ControllerAction("MyEmployeeTerritories", "createTerritory",
    "Insert", ActionPhase.Before)]
protected void NewTerritorySave(string territoryCode,
    string territoryDescription, int regionId)
{
    PreventDefault();
    Employees emp = (Employees)Context.Session["Employee"];
    Territories t = new Territories();
    t.TerritoryID = territoryCode;
    t.TerritoryDescription = territoryDescription;
    t.RegionID = regionId;
    if (t.Insert() == 1)
    {
        EmployeeTerritories et = new EmployeeTerritories();
        et.TerritoryID = t.TerritoryID;
        et.EmployeeID = emp.EmployeeID;
        et.Insert();
    }
    Result.HideModal();
    Result.ExecuteOnClient(
        String.Format("$find('{0}').goToPage(-1)",
        Context.Session["ControllerID"]));
}
```

VB

```
<ControllerAction("MyEmployeeTerritories", "createTerritory", _
                      "Insert", ActionPhase.Before)> _
```

```
Protected Sub NewTerritorySave(ByVal territoryCode As String, _
    ByVal territoryDescription As String, ByVal regionId As Integer)
    PreventDefault()
    Dim emp As Employees = CType(Context.Session("Employee"), Employees)
    Dim t As Territories = New Territories()
    t.TerritoryID = territoryCode
    t.TerritoryDescription = territoryDescription
    t.RegionID = regionId
    If t.Insert() = 1 Then
        Dim et As EmployeeTerritories = New EmployeeTerritories()
        et.TerritoryID = t.TerritoryID
        et.EmployeeID = emp.EmployeeID
        et.Insert()
    End If
    Result.HideModal()
    Result.ExecuteOnClient( _
        String.Format("$find('{0}').goToPage(-1)", _
        Context.Session("ControllerID")))

End Sub
```

Command *command2* is not based on any real database table. We are intercepting a request from *createTerritory* view to insert a new record via *NewTerritorySave* method. Any data fields available in the view can be listed as parameters of the method.

First, we prevent any default logic from executing when our method exits by calling *PreventDefault*.

Then we use the automatically generated business objects *Employees*, *Territories,* and *EmployeeTerritories* to update the database. Business objects are not equipped with database connections or commands, which allows storing them safely in ASP.NET session. Data manipulation commands are executed by business objects via Data Aquarium Framework. Business objects rely on the same data controller architecture to promote business logic reuse.

Last, we have to hide the modal view displayed at the moment in the user's browser. The request is executed in the context of the client side view *createTerritory*. Many methods available via business rules *Result* property are producing small snippets of JavaScript that are being accumulated while the request is processed on the server. Nothing happens until the method exists and the batch is returned to the browser.

Most likely you would want the employee territories refreshed and display a new territory that employee is responsible for. The last line of the method sends a JavaScript command to the browser to refresh the view that has requested displayed of *createTerritory* view in the first place. Client ID of the view was available in *NewTerritory* method via *Arguments.ContextKey* property.

## CONCLUSION

Simple and power business rules model of Data Aquarium Framework allows archiving functionality of a significant complexity with a very modest amount of code. Database interactions are greatly simplified by business objects automatically generated by Code OnTime Generator.

Business rules and objects are available to subscribers to premium projects only.

Code OnTime LLC

http://www.codeontime.com