New Data Entity Code

FX-Price-Predictor

This document describes how to add code to support a new data entity that will allow adding, deleting, sorting and filtering the data.

2010

Kevin Drummond

3/28/2010

Contents

[eventmanager.aspx 3](#_Toc257640402)

[js/eventmanager.js [CONTROL BINDING] 3](#_Toc257640403)

[js/eventmanager.js [FUNCTIONS] 3](#_Toc257640404)

[css/eventmanager.css 5](#_Toc257640405)

[templates/<entity>.xslt 5](#_Toc257640406)

[request.aspx.cs 7](#_Toc257640407)

# Overview

New data entities need a number of functions, controls, styles and other objects added to the FX-Price-Predictor in order to display and process properly. Each of the following sections is a separate file with sample code to make the entity work. The header of the section is the filename and in some cases also a note about what part of the file the code belongs in. You may copy the code into the appropriate file and replace the data entity tag with the name of your data entity. Note that you see upper case <Entity> and lowercase <entity>. This is purely for readability and cleanliness of code, so I recommend replacing the data entity names appropriately. For instance, if your entity name is fxevent, replace <Entity> with one naming scheme, such as “FXEvent” and replace <entity> with “fxevent”.

# // eventmanager.aspx

<div id="<entity>s" class="dataframe">

<h3><entity>s</h3>

<div id="<entity>s\_controls" class="fxcontrols-box">

<p><a id="get<Entity>s" class="fxcontrol-refresh fg-button ui-state-default ui-corner-all" href="#">Refresh</a></p>

<p><a id="del<Entity>s" class="fxcontrol-del fg-button ui-state-default ui-corner-all" href="#">Delete <Entity> (s)</a></p>

<p><a id="add<Entity>sTableRow" class="fxcontrol-add fg-button ui-state-default ui-corner-all" href="#">Add <Entity></a></p>

<p><a id="save<Entity>New" class="fxcontrol-save fg-button ui-state-default ui-corner-all" href="#">Save Event</a></p>

<p><a id="cancel<Entity>New" class="fxcontrol-cancel fg-button ui-state-default ui-corner-all" href="#">Cancel</a></p>

</div>

<div id="<entity>list">

</div>

</div>

# // js/eventmanager.js [CONTROL BINDING]

// Control to get events

$("#get<Entity>s").live("click", function() {

get<Entity>s();

return false;

});

// Control to delete events

$("#del<Entity>s").live("click", function() {

del<Entity>s\_Verify(this);

return false;

});

// Control to add event

$("#save<Entity>New").live("click", function() {

add<Entity> (this);

return false;

});

# // js/eventmanager.js [FUNCTIONS]

/\*\*

\* function: get<Entity>s()

\* Load <Entity> list html block from server

\*/

function get<Entity>s() {

$.post("request.aspx", { a: "get<Entity>s" }, function(response) {

$("#<entity>list").html("").append(response);

$o<Entity>sTable = $("#<entity>listTable").dataTable();

});

}

/\*\*

\* function: get<Entity>AddFormRow()

\* Load <Entity> list html form elements block from server

\*/

function get<Entity>sAddFormRow() {

$.post("request.aspx", { a: "get<Entity>sAddFormRow" }, function(response) {

$s<Entity>sAddRowForm = response.replace(/<\?xml(?:.|\s)\*?>/g, "").replace(/\"/g, "'");

});

}

/\*\*

\* function: del<Entity>s()

\* Delete selected <Entity>s from server

\*/

function del<Entity>s(v, m, f) {

var aEvtids = [];

if (v) { // If we clicked OK...

// Build the list of checked checkboxes into a string that will be passed to the web service.

$("#<entity>list input[type=checkbox]:checked").each(

function(index) {

aEvtids.push($(this).val());

}

);

// Send those <Entity>s to Hell.

$.post("request.aspx", { a: "del<Entity>s", "evtids": aEvtids.join() }, function(response) {

$.prompt(response);

get<Entity>s();

});

}

}

/\*\*

\* function: del<Entity>s\_Verify()

\* Verify deletions before sending request to server.

\*/

function del<Entity>s\_Verify() {

$.prompt("Do you really want to delete these <Entity>s? This action is permanent.", {

submit: del<Entity>s,

buttons: { Ok: true, Cancel: false }

});

}

/\*\*

\* function: add<Entity> ()

\* Add new <Entity> row to database

\*/

function add<Entity>(obj) {

var curr = $('#<Entity>listTable #curr').val();

$.prompt('Click OK to confirm adding forex <Entity> ' + <Entity>id + ' (' + <Entity>name + ').', { buttons: { Ok: true, Cancel: false },

callback: function(v, m, f) {

if (v) {

$.post("request.aspx", { "a": "add<Entity>", "curr": curr }, function(response) {

// Server response

$.prompt(response);

// Refresh table

get<Entity>s();

// Enable add control, hide save control

$("#save<Entity>New").hide('fast');

$("#cancel<Entity>New").hide('fast');

$("#add<Entity>sTableRow").show('slow');

});

}

}

});

}

/\*\*

\* function: fnClickAdd<Entity>Row()

\* Adds row to the <Entity>s HTML table.

\*/

function fnClickAdd<Entity>sRow() {

// Append it to the <Entity>s table.

//$o<Entity>sTable.fnAddData($a<Entity>RowElements);

$('#<Entity>listTable tbody tr:first').before($s<Entity>sAddRowForm);

$('#<Entity>listTable #curr').parent().html("").append($sCurrSelect);

// Bind date control to next\_date input field

$(function() {

$("#next\_date").datepicker();

});

// Focus on the first form element.

$('#<Entity>listTable #<Entity>id').focus();

// Enable save control, hide add control

$("#add<Entity>sTableRow").hide('fast');

$("#save<Entity>New").show('slow');

$("#cancel<Entity>New").show('slow');

}

/\*\*

\* function: fnClickDel<Entity>sRow()

\* Adds row to the <Entity>s HTML table.

\*/

function fnClickDel<Entity>sRow() {

// Remove first row in <Entity> table

$('#<Entity>listTable tbody tr:first').remove();

// Enable add control, hide save control

$("#save<Entity>New").hide('fast');

$("#cancel<Entity>New").hide('fast');

$("#add<Entity>sTableRow").show('slow');

}

# // css/eventmanager.css

#<entity>s {

width: <Insert\_Width\_Here>;

}

# // templates/<entity>.xslt

// Add the necessary XSLT code into its own file. For example, here is an XSLT file which can be used for the fxevent entity. The important elements or attributes you must have for proper styling or functionality are highlighted in yellow. Because this is a live example, incidents of “fxevent” should be replaced by your entity name.

<?xml version="1.0" encoding="utf-8"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

xmlns:msxsl="urn:schemas-microsoft-com:xslt" exclude-result-prefixes="msxsl">

<xsl:output method="html" indent="yes"/>

<xsl:template match="/">

<table cellpadding="0" cellspacing="0" border="0" class="display" id="fxeventlistTable">

<thead>

<tr>

<th>Currency</th>

<th>Event ID</th>

<th>Name</th>

<th>Next Date</th>

<th>Recurring</th>

<th>Previous</th>

<th>Importance</th>

<th>Watch</th>

<th>Delete</th>

</tr>

</thead>

<tbody>

<xsl:for-each select="ArrayOffxevent/fxevent">

<tr class="gradeA">

<td>

<xsl:value-of select="currency"/>

</td>

<td>

<xsl:value-of select="eventid"/>

</td>

<td class="fxeventtable-name">

<xsl:value-of select="name"/>

</td>

<td>

<xsl:value-of select="substring(next\_date,1,10)"/>

</td>

<td>

<xsl:value-of select="recurring"/>

</td>

<td>

<xsl:value-of select="previous"/>

</td>

<td>

<xsl:value-of select="importance"/>

</td>

<td>

<xsl:value-of select="watch"/>

</td>

<td>

<xsl:element name="input">

<xsl:attribute name="type">checkbox</xsl:attribute>

<xsl:attribute name="value">

<xsl:value-of select="eventid"/>

</xsl:attribute>

<xsl:attribute name="name">

<xsl:text>delete</xsl:text>

<xsl:value-of select="eventid"/>

</xsl:attribute>

<xsl:if test="@default='yes'">

<xsl:attribute name="checked">checked</xsl:attribute>

</xsl:if>

</xsl:element>

</td>

</tr>

</xsl:for-each>

</tbody>

</table>

</xsl:template>

</xsl:stylesheet>

# // request.aspx.cs

case "get<Entity>s":

IEnumerable<<entity>> <entity>s = from fxe in fxdc.<entity>s

select fxe;

Response.Write(FXTransforms.Linq2SQL\_XSLTransform("<entity>s", <entity>s, "templates/<entity>.xslt").ToString());

break;

case "get<Entity>sAddFormRow":

IEnumerable<<entity>> <LINQ\_OBJECT>form = from fxe in fxdc.<entity>s

select fxe;

Response.Write(FXTransforms.Linq2SQL\_XSLTransform("<entity>s", <entity>form, "templates/<entity>\_add.xslt").ToString());

break;

case "del<Entity>s":

String[] aEvtids = ((String)coll["evtids"]).Split();

var <entity>s\_deleted = from fxe in fxdc.<entity>s where aEvtids.Contains(fxe.eventid)

select fxe;

fxdc.<entity>s.DeleteAllOnSubmit(<entity>s\_deleted);

try

{

fxdc.SubmitChanges();

Response.Write("Successfully deleted records.");

}

catch (Exception ex)

{

Response.Write(ex);

// Provide for exceptions.

}

break;

case "update<Entity>":

break;

case "add<Entity>":

<entity> <entity>\_new = new <entity> ();

<entity>\_new.<field1> = coll["<field1>"];

…

fxdc.<entity>s.InsertOnSubmit(<entity>\_new);

fxdc.SubmitChanges();

Response.Write("Adding " + <entity>\_new.eventid + " (" + <entity>\_new.name + "). Err No: " + <entity>\_new.err\_no + ", Err Desc: " + <entity>\_new.err\_desc);

break;