

Data View Extender Events: selected and executed

[Data Aquarium Framework](#) includes a collection of JavaScript components that are rendering user interface in a web browser and are interacting with the server components of the framework. You can participate in the client-side life cycle by creating JavaScript event handlers that are responding to *selected* and *executed* events raised by *Web.DataView* component instances injected into the page by *DataViewExtender* server components.

Event *selected* is fired when a row is selected in the user interface. You can use *get_selectedKey()* method of *Web.DataView* component to find a primary key of a selected record.

Event *executed* is fired when a command has been executed. Event handler is provided with the result of execution on the server including auto-incremental primary key field values and error messages if any.

Let's explore these events by creating master-detail AJAX grid views that link *Products* to *Suppliers* in *Northwind* database.

Generate an [Aquarium Express](#) project with [Code OnTime Generator](#) from *Northwind* database and open this project in *Visual Studio 2008* or *Visual Web Developer 2008 Express Edition*.

Add page *Events.aspx* to the root of the web site as shown in the sample below.

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

<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="Server">

    <script type="text/javascript">
function pageLoad() {
```

```
var dataView = Web.DataView.find('SuppliersExtender');
if (dataView) {
    dataView.add_selected(onSupplierSelected);
    dataView.add_executed(onSupplierExecuted);
}
}

var justInserted = false;

function onSupplierSelected(sender) {
    if (justInserted)
        justInserted = false;
    else
        updateProductsVisibility(sender.get_lastCommandName() != 'New');
}

function onSupplierExecuted(sender, args) {
    switch (sender._lastArgs.CommandName) {
        case 'Insert':
            updateProductsVisibility(true);
            justInserted = true;
            break;
        case 'Delete':
            updateProductsVisibility(false);
            break;
    }
}

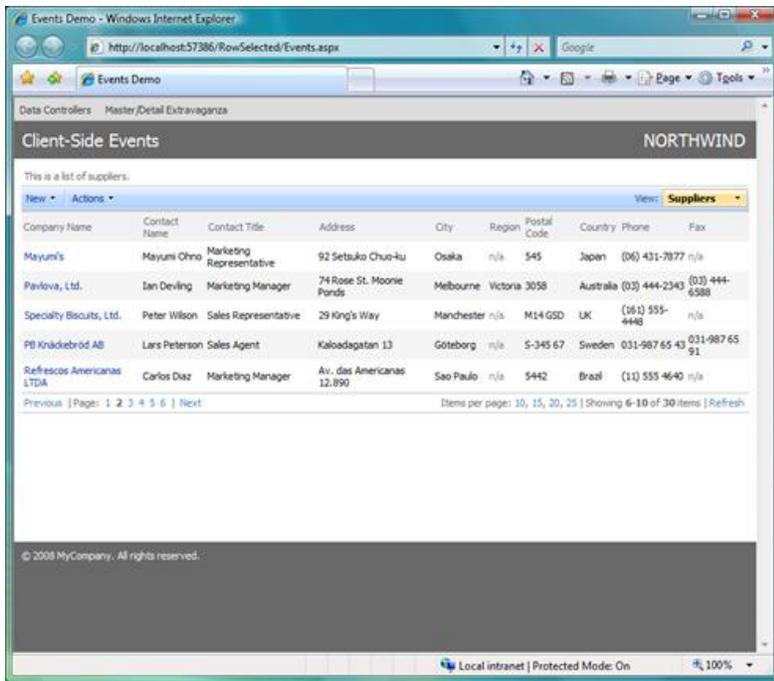
function updateProductsVisibility(visible) {
    Sys.UI.DomElement.setVisible($get('ProductsPanel'), visible);
}

</script>

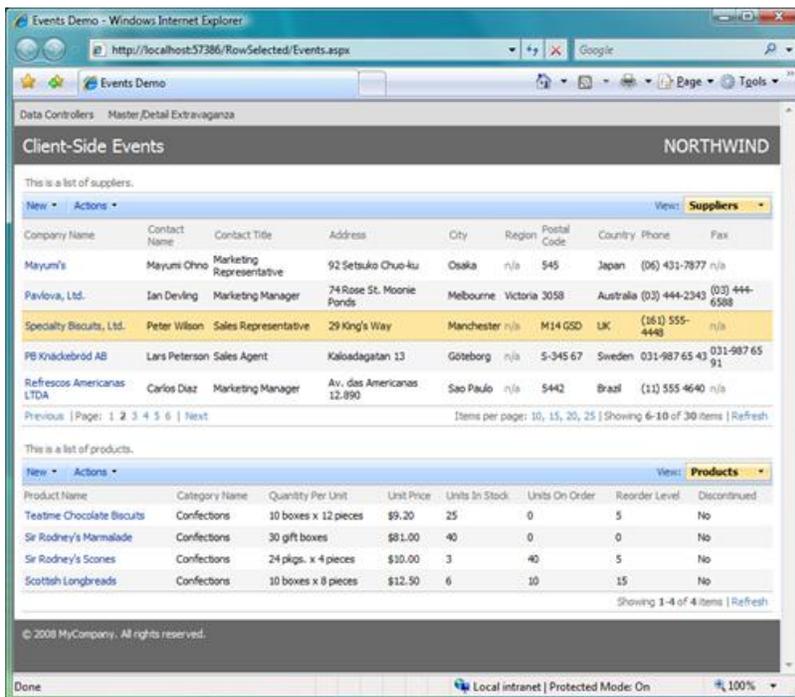
</asp:Content>
```

```
<asp:Content ID="Content2" ContentPlaceHolderID="Header1Placeholder"
runat="Server">
    Client-Side Events
</asp:Content>
<asp:Content ID="Content3" ContentPlaceHolderID="Header2Placeholder"
runat="Server">
    Northwind
</asp:Content>
<asp:Content ID="Content4" ContentPlaceHolderID="BodyPlaceholder"
runat="Server">
    <div id="Suppliers" runat="server" />
    <aquarium:DataViewExtender ID="SuppliersExtender" runat="server"
        Controller="Suppliers" TargetControlID="Suppliers" PageSize="5"/>
    <br />
    <div id="ProductsPanel" style="display:none">
        <div id="Products" runat="server" />
        <aquarium:DataViewExtender ID="ProductsExtender" runat="server"
            Controller="Products" TargetControlID="Products" PageSize="5"
            FilterFields="SupplierID" FilterSource="SuppliersExtender" />
    </div>
</asp:Content>
```

Open this page in a browser. You will see that *products* are not displayed under the *suppliers* grid.



Click on any *supplier* row and the page will display a list of relevant *products*. If you start adding a new supplier then the list of *products* will disappear. The list of *products* will show up again when you actually insert a *supplier* record. The list of *products* goes away when you delete a *supplier*.



How does this work?

Let's do a quick code walkthrough.

Element *ProductsPanel* is wrapped around the placeholder for *Products*. It doesn't matter if the *DataViewExtender* component is within the boundaries of the panel. Just make sure that the placeholder is hidden.

```
<div id="ProductsPanel" style="display:none">
  <div id="Products" runat="server" />
  <aquarium:DataViewExtender ID="ProductsExtender" runat="server"
    Controller="Products" TargetControlID="Products" PageSize="5"
    FilterFields="SupplierID" FilterSource="SuppliersExtender" />
</div>
```

ASP.NET AJAX Extensions library automatically invokes JavaScript function *pageLoad()* when the runtime client components are instantiated and initialized. It works very much like *Page_Load* function in your server code, which is called when *ASP.NET* is ready to pass you a control over a page with all server components ready for use. Method *Web.DataView.find* will find an instance of JavaScript *Web.DataView* component that was injected into the page by *SuppliersExtender* server component. This is a convenient spot to assign event handlers of *selected* and *executed* events.

```
function pageLoad() {
  var dataView = Web.DataView.find('SuppliersExtender');
  if (dataView) {
    dataView.add_selected(onSupplierSelected);
    dataView.add_executed(onSupplierExecuted);
  }
}
```

Function *onSupplierSelected* will be called whenever a *supplier* record is selected by a user. This method is invoking *updateProductsVisibility* function and passes *true* as a parameter if the last executed action command is not *New*. We are also making sure that

visibility of the *products* grid is not altered if we just have executed an insertion of a new *supplier*.

```
var justInserted = false;

function onSupplierSelected(sender) {
    if (justInserted)
        justInserted = false;
    else
        updateProductsVisibility(sender.get_lastCommandName() != 'New');
}
```

Actions that cause an insertion or a deletion of a *supplier* will result in *product* details being displayed or hidden accordingly.

```
function onSupplierExecuted(sender, args) {
    switch (sender._lastArgs.CommandName) {
        case 'Insert':
            updateProductsVisibility(true);
            justInserted = true;
            break;
        case 'Delete':
            updateProductsVisibility(false);
            break;
    }
}
```

The actual update of *products* grid visibility is very simple. We are using standard JavaScript components of *ASP.NET AJAX Extensions* library to hide DOM element with *ProductsPanel* identifier.

```
function updateProductsVisibility(visible) {
    Sys.UI.DomElement.setVisible($get('ProductsPanel'), visible);
}
```

You can enhance this code to suit your own needs by interacting with *div* elements on your page, which will typically result in hiding and showing of arbitrary page fragments to create less cluttered and more intuitive user interface presentation in your *AJAX ASP.NET* applications created with [Code OnTime Generator](#).