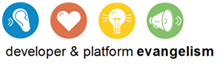
* 1. 

Demo Script

AdventureWorks using AJAX

Lab version: 1.0.0

Last updated: 12/30/2010



1. Contents

[Overview 3](#_Toc256779634)

[Key Messages 3](#_Toc256779635)

[Key Technologies 3](#_Toc256779636)

[Time Estimates 3](#_Toc256779637)

[Setup and Configuration 3](#_Toc256779638)

[Opening Statement 4](#_Toc256779639)

[Step-by-Step Walkthrough 4](#_Toc256779640)

[Segment #1 – Exploring the DataGrid class 5](#_Toc256779641)

[Segment #2 – Using client-templates, dataContext and dataView 8](#_Toc256779642)

[Segment #3 – Using ASP.NET Ajax Library Declarative Syntax 12](#_Toc256779643)

[Summary 14](#_Toc256779644)



Overview

* 1. This document provides setup documentation, systematic instructions, and a written script for showing the client templates capabilities in ASP.NET Ajax Library. This document can also serves as a tutorial or walkthrough of the exposed technologies.
  2. The ASP.NET Ajax Library includes a powerful client data access library. By taking advantage of ASP.NET Ajax client templates and client data-binding, you can build a database-driven web application that executes entirely within the web browser.
  3. The ASP.NET Ajax Library Beta provides a simple yet flexible way to bind JSON data to custom templates resulting in reduced code and simplified maintenance. The object that makes this possible is the DataView component located in the Sys.UI namespace.

### Key Messages

* 1. The DataView component provides a convenient way to access local or remote data and bind it to client-side templates for display in the browser. An instance of the DataView can be created imperatively using JavaScript or declaratively using HTML syntax.

### Key Technologies

This demo uses the following technologies:

* 1. Microsoft Visual Studio 2010
  2. ASP.NET Ajax Library

### Time Estimates

* + Estimated time to complete the demo: 20 min

Setup and Configuration

## System Requirements

* + Microsoft Visual Studio 2010

Opening Statement

* 1. Today, I would like to walk you through several demo web pages built to show how the DataView component can be used to fetch JSON data from remote services and bind it to client-side templates.
  2. The following is a brief description of the things that will be shown during this demo:
  3. Exploring a DataGrid implementation using a DataView component and JavaScript code.
  4. Exploring the same scenario using a plain DataView component and data Binding.
  5. Defining a DataView component declaratively using attributes.

Step-by-Step Walkthrough

1. This demo is composed of the following segments:
   1. Exploring DataGrid class
   2. Using client-templates, dataContext and dataView
   3. Using ASP.NET Ajax Library Declarative Syntax

### Segment #1 – Exploring the DataGrid class

|  |  |  |
| --- | --- | --- |
| Action | Script | Screenshot |
| * 1. Open Microsoft Visual Studio 2010 from **Start | All Programs**.   2. Open the **AdventureWorksAjax.sln** solution located under the **Source** folder of this demo (and choosing the folder that matches the language of your preference.)   3. In **Solution Explorer**, open the DataGrid.htm file by double-clicking it. | * + First, we will explore a possible implementation of a DataGrid based on the DataView template using the ASP.NET Ajax library components. To do this, we will open a prebuilt DataGrid page to show the approach used in this sample.   + The DataGrid.htm page consists only on a template table that will be populated by the DataGrid class. |  |
| * 1. Show the <script> tags including some basic **ASP.NET AJAX Library** script files.   2. Show that the DataGrid.js and Test.js files are being referenced. | * + The DataGrid.htm page, as well as all the demo pages, includes the required ASP.NET Ajax libraries script files. In this case, the core, the templates components, ADO.NET and Data Context scripts.   + The DataGrid.js script file contains the definition of the DataGrid class used during this part of the demo   + The Test.js is simply the starting point of the demo and creates the required ADO.NET data context as well as the DataGrid itself. |  |
| * 1. In **Solution Explorer**, open the Content\Scripts\DataGrid.js file by double-clicking it. | * + The DataGrid class inherits from the Ajax DataView component and adds the possibility of using a table tag to wrap the data.   + By inspecting the initialize method, you will find that it just creates headers and the columns for each field in the data context, and lets the DataView class to populate cells with the actual data.   + In addition, it sets the template attributes used by the template to perform some actions. For example, it attaches the select command on each row. |  |
| * 1. In **Solution Explorer**, open the Content\Scripts\Test.js file by double-clicking it. | * + Now, we will open and explore the Test.js script file, which contains the initialization code.   + First, it creates an ADO.NET data context using a prebuilt WCF Data Service named AdventureWorks.svc.   + Next, it uses the data context when creating the Datagrid component, as well as the fetchOperation and fetchParameters parameters.   + You should notice the alternateRowClass and the columns/fields definition parameters.   + Also, we used the table template and the customer placeholder in the DataGrid’s initialization. |  |
| * 1. Set the DataGrid.htm file as the start up page by right-clicking it, and selecting **Set as Start Page**.   2. Press F5 to run the application.   3. Wait for the grid to be populated with data and select different rows by clicking them.   4. Close the browser. | * + Let’s run the application.   + Notice that at first, a loading message is visible until the data is automatically populated in the table element. This is achieved by using client templates inside the DataGrid class. |  |



### Segment #2 – Using client-templates, dataContext and dataView

|  |  |  |
| --- | --- | --- |
| Action | Script | Screenshot |
| * 1. In **Solution Explorer**, open the Default.htm file by double-clicking it.   2. Show the table element inside the body tags.   3. Show the **thead** element with five header columns inside the table. | * + We will check another implementation of the same scenario. This uses only DataView templates. To do this, we will open the Default.htm page, which contains the customer list as a client template.   + Inside the body HTML element, you will find a table element including a table header for the list of customers. This has the same effect as shown in the previous DataGrid sample. |  |
| * 1. Show the tbody client-template below the thead element.   2. Show the binding syntax. | * + Below the headers we included a template that is equivalent to the one used in the previous sample. Notice that the tbody element with the class attribute is set to sys-template.   + You should notice the row attributes used to attach the select command and the alternating row class.   + You may notice the binding syntax used on each cell. This is the same syntax used in the DataGrid class in the previous sample. |  |
| * 1. Show the div element at the bottom of the page body. | * + Additionally, in this sample we create a right panel where the user will be able to edit items as they are selected.   + This is simply another client template that will help us show an explicit binding. |  |
| * 1. In the **Solution Explorer**, open the Content\Scripts\Default.js file by double-clicking it.   2. Show the code to create a DataContext and the first DataView components. | * + Now we will explore the code that populates the client template with the data context data. To do this, we will open the Default.js script file.   + In this file, we will find the same data context creation we saw in the previous sample but in this case, it is used in conjunction with a DataView component.   + Up to this point, the outcome will be the same that in the DataGrid sample. |  |
| * 1. Show the code to create the second DataView component and the binding between the two templates. | * + The second DataView component is used to show the item details and let the user edit the data.   + The way it is being populated is through a Binding component that ties both the customers list and the customer details.   + Notice the source and target properties of the Binding component reference the client templates created above. |  |
| * 1. Set the Default.htm file as the start up page by right-clicking it select **Set as Start Page**.   2. Press **F5** to run the application.   3. Wait for the grid to be populated with data and select different rows by clicking on them.   4. Edit some items in the right side pane and click the **Update** button.   5. Close the browser. | * + Let’s run the application to see how this works.   + We will obtain the customers list with the alternating row style and the ability to select items.   + In addition, when selecting a new item the data will be available on the right panel for in-memory edition. |  |

### Segment #3 – Using ASP.NET Ajax Library Declarative Syntax

|  |  |  |
| --- | --- | --- |
| Action | Script | Screenshot |
| * 1. In **Solution Explorer**, open the Declarative.htm file by double-clicking it.   2. Show the provided creation of an adoNetDataContext in the existing file. | * + We have walked through two different implementations of the same scenario using ASP.NET Ajax library using explicit syntax. This means you need to include script files in order to achieve the required result.   + Now, we will check the possibility of using a declarative syntax to obtain the same outcome without including any script file except the ASP.NET Ajax library files. To do this, we will open the Declarative.htm file.   + Notice that you still have to create the ADO.NET data context in order to retrieve data. However, there are no other custom script files needed. |  |
| * 1. Scroll down to the tbody element of the main table.   2. Highlight the sys:attach attribute.   3. Highlight the dataview:dataprovider attribute. | * + Below, we will check the client template to see that all the required initialization is placed directly in the HTML markup as **sys** and **dataview** attributes.   + The sys:attach attribute converts this simple tbody element into a client template using the DataView component.   + The dataprovider attribute uses the created ADO.NET data context to gather data on execution time.   + You can check the rest of the attributes having a correspondence with the previous samples. |  |
| * 1. Scroll down to the div element, located at the bottom of the page.   2. Highlight the **dataview:data** attribute. | * + The right panel, used for editing items, is initialized as a client template in the same way we did before.   + Notice the dataview:data attribute. This replaces the Binding component we created by hand in the previous sample. We used the id of the previous template to tie both templates declaratively. |  |
| * 1. Set the Declarative.htm file as the start up page by right-clicking it and selecting **Set as Start Page**.   2. Press **F5** to run the application.   3. Wait for the grid to be populated with data and select different rows by clicking on them.   4. Edit some items in the right side pane and click the Update button.   5. Close the browser. | * + Let’s run the application to see how this works.   + We will obtain the same customers list with the alternating row style and when selecting an item, the right panel will be populated with that data for editing. |  |

Summary

* 1. In this demo, you explored different approaches that can be used to obtain a data grid scenario using ASP.NET Ajax Library Beta. Additionally, you saw a simple data-binding scenario where two client templates can be tie to edit data.