Script Application Builder for IBM Web Experience Factory

Background and overview

The popularity of JavaScript for web application development continues to grow, and there are now a huge number of powerful JavaScript libraries for all sorts of purposes. With Web Experience Factory, developers have increasingly used libraries such as jQuery as part of their application development. They can mix features of Web Experience Factory such as the data service integration builders and Page Automation with their own JavaScript code. In some cases they're building the client UI with JavaScript/HTML/CSS, using Web Experience Factory for data integration and to support Portal features.

This project provides a Script Application builder and the Script Library builder for Web Experience Factory that makes it very easy for developers to build such script-based portlets. With this builder, developers can work on their JS/CSS/HTML in the Eclipse-based IDE, then create a simple model using the Script Application builder to reference your JS/CSS/HTML, and publish it to Portal. This IDE-based development of script-based applications is a complementary approach to the new IBM Script Portlet available from the Portal Catalog. With Script Portlet you can also use your own HTML/JS/CSS to implement a portlet, but it's done in a browser, and the “code” of the application is stored and managed in IBM Web Content Manager.

Script Application Builder

The Script Application builder automates a common pattern for building portlets or web applications using script libraries such as JQuery. The pattern includes these key pieces:

- References to script libraries such as jquery, jQuery UI, charting libraries, etc. These libraries can be provided in a portal theme, from a hosted server, or from within the Web Experience Factory project.
- Application JS, HTML, and CSS.
- JSON/REST services for accessing application data and services.

This builder completely automates that pattern, so that with a single builder and a bit of application JS/HTML you can generate a complete portlet or web application, using JSON/REST to access data and services from any Web Experience Factory Service Provider model. You select the script libraries you want to use, point to your application JS/HTML/CSS files, and optionally select a Data Service provider model that you want to access via JSON/REST. The builder generates the application, including the server-side code for REST/JSON services and JS variables containing URLs for invoking all the operations in the service.

The samples shown here use a single page application, but you can combine pages built with this builder with other Web Experience Factory pages and features. For example, you might have a transactional portlet with lists and forms built using the standard Web Experience Factory builders, then enhance it with a page containing a jQuery-based chart.

When using this builder you create and edit the application JS, HTML, and CSS in your project. The builder then automates the design pattern for pulling the whole application together into a portlet.

Script Application Builder Inputs

Libraries
Here is where you specify the JQuery or other libraries you want to use. You can add as many libraries as you want, picking from a list.

The list of libraries comes from the HTML files that are in the /samples/script_builder/libraries folder. Each of those HTML files contains links to JS and/or CSS files, either within the project or hosted on the web. If you want to add more libraries, simply create additional HTML files in that folder with links to the files needed for the library.

Include Libraries
Here you can specify whether the library links should be omitted when running in Portal. In general, the recommendation for using JS libraries in Portal is to include them in the Portal theme, not in individual portlet WARs. However, for running Web Experience Factory models standalone you will need the library links.

Application HTML
Here you specify the HTML file that will be imported to create a page where you'll be using JS. This page should include a HEAD tag.

Page Name To Create
This is the name of the generated page. You can use "main" to have the page loaded automatically when the application runs.

Application Script File
This is where you tell the builder what JS file contains your application code.

Application CSS File
This is an optional input for bringing in application CSS.

Add Service Provider Support
If selected, this lets you specify a Service Provider model which will automatically be REST-enabled for getting JSON data from your application code.

Service Provider
The name of a Service Provider model in the project. Any provider can be specified, and the provider can be implemented using any of WEF's data access and transformation builders.

JS Variable Name for Service URLs
A JS variable with the specified name will be generated for you, and it will have members that are URLs for each of the operations in the Data Service. Your JS application code can then use those URLs to call services and retrieve JSON data. For example, one of the samples uses this code to get and display a list of orders. "salesData" is the JS Variable Name that was specified, and here it's invoking the getSalesReports operation:

```
// Fetch dynamic JSON using WEF REST Enabled Data Service REST URL
```

```
$.getJSON(salesData.getSalesReportsURL, {}, function(ajaxData) {
      chartSample.displayBarChartFromService(
            'ChartDiv',  ajaxData.SalesData.MonthData);
});
```

Sample models

There are four sample applications included with the builder.

ChartSample model: This shows a data chart using the jqPlot charting library. In this example, data comes from the MonthlySalesProvider model, which is specified in the Script Application builder. Then the JSON/REST service is accessed using the code shown above.

TabsSample model: This is a very simple example showing some HTML that's displayed using jQuery UI. All of the UI is in the imported HTML page, with no data views or forms.

CustomersTable model: This samples uses the jQuery Datatables library to display a table of data. The data comes from the CustomerListProvider.model, and it's accessed using JSON/REST by the JavaScript code.

EventingListView and EventingDetails models: This pair of models shows how you can use jQuery eventing to communicate between portlets. When placed on a Portal page, you can click on one of the items in the list portlet and see the corresponding details in the Details portlet. For this sample, there's also an EventingContainer model included that displays both of those models using the Model Container builder, for testing the eventing when running standalone outside of WebSphere Portal.

Running the samples

To run the samples you should:

- Create a Web Experience Factory project in the Designer and publish it to a test server.
- Use the Eclipse project menu "Import" -> Web Experience Factory Archive menu action, to import the sample archive into your project.
- The charting sample uses the XML File Data Service builder available on the WEF wiki, so download that builder and import it into your project. It's available from here: http://www-10.lotus.com/ldd/pfwiki.nsf/dx/Sample_XML_File_Data_Service_Builder
- Publish the project on your server.
- Open one of the models such as TabsSample and Run it.
- Add the portlets to a Portal page to see them running in Portal.

Script Library Builder

The Script Library builder automates a common pattern for building portlets or web applications using script libraries such as JQuery. The pattern includes these key pieces:

- References to script libraries such as jquery, jQuery UI, charting libraries, etc. These libraries can be provided in a portal theme, from a hosted server, or from within the Web Experience Factory project.
- Application JS, HTML, and CSS.

- JSON/REST services for accessing application data and services.

This builder automates that pattern, so that with an existing model using a single builder you can add scripting libraries and customize a complete portlet or web application, using other script libraries to enhance the user experience. You select the script libraries you want to use, point to any optional application JS/CSS files, and optionally select a Data Service provider model that you want to access via JSON/REST. The builder adds these components to your application, including the server-side code for REST/JSON services and JS variables containing URLs for invoking all the operations in the service.

The samples shown here use Page Automation builders, but you can combine pages built with this builder with other Web Experience Factory pages and features. For example, you might have a transactional portlet with lists and forms built using the standard Web Experience Factory builders, then enhance it with a page containing a jQuery-based chart.

Script Library Builder Inputs

Name
Here is where you enter the optional name for this builder. You might want to enter info that helps describe what libraries are being used.

All Pages
Here you select weather you want the scripts to be added to all pages in the model or if unchecked select the pages you want targeted.

Choose Pages
This is a list of pages that you can select which pages have the other items added to. This is only used if All pages is unchecked.

Libraries
Here is where you specify the JQuery or other libraries you want to use. You can add as many libraries as you want, picking from a list.

The list of libraries comes from the HTML files that are in the /samples/script_builder/libraries folder. Each of those HTML files contains links to JS and/or CSS files, either within the project or hosted on the web. If you want to add more libraries, simply create additional HTML files in that folder with links to the files needed for the library.

Include Libraries
Here you can specify whether the library links should be omitted when running in Portal. In general, the recommendation for using JS libraries in Portal is to include them in the Portal theme, not in individual portlet WARs. However, for running Web Experience Factory models standalone you will need the library links.

Optional Script File
This is where you can optionally add common javascript that you would use on these pages

that perform common functions that use the script libraries.

Optional CSS File
This is where you can optionally add common style definitions that you would use on these pages.

Add Service Provider Support
If selected, this lets you specify a Service Provider model which will automatically be REST-enabled for getting JSON data from your application code.

Service Provider
The name of a Service Provider model in the project. Any provider can be specified, and the provider can be implemented using any of WEF's data access and transformation builders.

JS Variable Name for Service URLs
A JS variable with the specified name will be generated for you, and it will have members that are URLs for each of the operations in the Data Service. Your JS application code can then use those URLs to call services and retrieve JSON data. For example, one of the samples uses this code to get and display a list of orders. "salesData" is the JS Variable Name that was specified, and here it's invoking the getSalesReports operation:

```
// Fetch dynamic JSON using WEF REST Enabled Data Service REST URL
$.getJSON(salesData.getSalesReportsURL, {}, function(ajaxData) {
      chartSample.displayBarChartFromService(
            'ChartDiv',  ajaxData.SalesData.MonthData);
});
```

Sample models

There are two sample applications included with the builder.

MobileCustomerList model: This shows a list using the jQuery mobile library. In this example, the data comes from the MonthlySalesProvider model, and is displayed using a Data Page builder. To see the difference behavior you can disable the Script Library builder.

MobileViewAndForm model: This is an example of using the jQuery mobile library with the View and Form builder. In this example, the data comes from the MonthlySalesProvider model, and is displayed using a View and Form builder. To make this work you also need to use a theme that would be compatible with the jQuery library. To see the difference behavior you can disable the Script Library builder.

Running the samples

To run the samples you should:

- Create a Web Experience Factory project in the Designer and publish it to a test server.
- Use the Eclipse project menu "Import" -> Web Experience Factory Archive menu action, to import the sample archive into your project.
- The charting sample uses the XML File Data Service builder available on the WEF wiki, so download that builder and import it into your project. It's available from here:

http://www-10.lotus.com/ldd/pfwiki.nsf/dx/Sample_XML_File_Data_Service_Builder

- Publish the project on your server.
- Open one of the models such as TabsSample and Run it.

References

Blog with links to articles and samples for using jQuery and other JS libraries with Web Experience Factory:
https://www.ibm.com/developerworks/community/blogs/b75d3ff5-8534-43ff-8eb0-8e33fc67f50e/entry/new_samples_and_builder_for_using_jquery_and_other_script_libraries_with_web_experience_factory?lang=en

Web Experience Factory wiki:
http://www-10.lotus.com/ldd/pfwiki.nsf