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. There is also a sample Script Application wizard that allows you to create a model with a Script Application builder where you can create the application page, script and css files based on template files that you can add to the project. 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. One thing to note if you are using a scripting template library, the templating in the HTML might interfere with the JSP processing. For Apps that use templating the standard ERB style of embedding executable or interpolated bits of code inside angle brackets This cause issues with JSP processing. This is an example of one way to change the underscore template processing can be changed to use a Mustache style by setting the _.templateSettings in the underscore.js from

```
 _.templateSettings = {
        evaluate : /<%([\s\S]+?)%>/g,
        interpolate : /<%=([\s\S]+?)%>/g,
        escape : /<%-([\s\S]+?)%>/g
 }; to
 _.templateSettings = {
        interpolate: /\{\{\=(.+?)\}\}/g,
        escape: /\{\{\-(.+?)\}\}/g,
        evaluate: /\{\{(.+?)\}\}/g
};
```

You would also need to change any encoding for the template in any of the HTML files to match your template style. There are different ways to control the templating characters in different js libraries, you should use the information and recommendations from the libraries on how to customize these settings.

Script Application 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.

Libraries
Here you can specify whether you will be adding library references to the application HTML or using the HTML as is. If you are using the HTML as is the library references would already be in the application HTML. A good use for this is if you have an existing single page application that you want to use as a portlet or you want to add a rest provider to add data to the existing application.

Library References
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 which is used by the builder to add references to the script libraries.

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);
});
```

Rest Data Service Support
This is an input where you can add js code to page to reference any selected data service and variable name. Similar to the Service Provider inputs but with this input you will need to add the data services to the model yourself.

Disable Smart Refresh
If selected, this sets the setting for smart refresh in the theme to turn off smart refresh. This is

needed if your default theme included dojo which sometimes interferes with other script libraries.

Default RDD File
The name of a RDD file in the project that you want used by pages created by page automation. This can be used to choose a Rich Data Definition that does not have dojo widgets, which can interfere with your selected libraries.

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.

BankAccountService model: This is an example showing how to create a portlet that can be added to your portal page that will provide other script based portlets on the same page access to data. In this scenario a developer wants to write their portlet using client side scripting, but how do you get your data to the client in a format that the scripting can use. This example uses the sample xml data service for it's data but it could be any of Web Experience Factory service providers. This standard data service is rest enabled with the data turned into json for the clients access. So how does another portlet get access to the data. Well that is why this model provides a portlet that is only visible in edit mode. This portlet that isn't visible in view mode contains a json variable that has a list of url's for all the services that it wants to expose. The other scripting based portlets on the same page can access this variable to get the url that can be used to return the data in a json format. The AccountChart.model is an example of a script based portlet that access the data and uses a scripting library to display the data. This model was generated using the Script Application wizard that is part of this offering. It used a previously built Single Page Application that has been zipped and included in the project in the WebContent folder(spchartwithwef.zip). This zip and it's application could also be used in the Script Portlet demonstrating how other no Web Experience Factory portlets could use this same technique to access data.

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 Application Wizard

The Script Application wizard automates the creation of script applications by using the Script Application Builder and templates that are available for the wizard to create the script application and it's needed components. It also gives you the ability to take existing single page applications from a zip archive or individual files you add to your project and turn them into portlets or add rest based services to them.

Script Application Wizard Inputs

Name

Here is where you specify the name that will be used for the application to keep it's components unique. This will be used to create a folder in the /samples/apps directory that will contain the application and it's component files. It also uses this name as the name for the variable for the service URL if one is added in the wizard.

Deploy this model as a portlet

If selected, this lets you specify a the title for the portlet and adds a Portlet Adapter builder to the model. The Portlet Adapter allows you to expose profile values for customization of the portlet that will be used on the portal server.

Portlet Title

This lets you specify a the title for the portlet. This is the name that identifies this portlet when it gets added to a page in the portal. This name is also used when other portlets communicate directly to this portlet.

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.

Libraries

Here you can specify whether you will be adding library references to the application HTML or using the HTML as is. If you are using the HTML as is the library references would already be in the application HTML. A good use for this is if you have an existing single page application that you want to use as a portlet or you want to add a rest provider to add data to the existing

application.

If you choose to add library references you will be get the following inputs

Library References
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 Template
Here you can specify the template type for the application you want to use.
The list of templates comes from the HTML files that are in the /samples/script_builder/wizard_templates folder. The names of those HTML files are used as the template name. If there are corresponding script and css files they would have the same name as the HTML file. If you want to add mode templates, simply create new files in this folder containing what you want.

If you choose libraries already referenced in HTML you will get the following inputs

Application Location
Here you choose if you are going to reference an existing application that you have added to your project. If you have added the files that are part of the application then you would choose use existing file. If you have a zip archive of the application you can choose get application from zip. The zip can be added to your project or with the wizard be added unzipped and deleted if you want.

If you choose Use existing file an input is added to allow you to enter the applications HTML page.

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 which is used by the builder to add references to the script libraries.

If you choose Get application from zip

Application Zip

Here you specify the zip file that will be extracted into the app folder created by the wizard and if an index.html file is found this will be set to the application HTML page. If the zip archive you want to reference is not already in the project it can easily be added by using the file chooser and selecting the add file to project.

Delete zip when done
If selected the zip file will be removed from the project after it's contents are extracted into the wizards app directory. This is useful to use in tandem with adding the zip with the file chooser and then deleting it from the project after it's contents have been extracted into the project.

Convert any templating
If selected the HTML files that are unzipped are searched through and any templating is converted based on the following settings.

Template script to replace
This is the name of the js file that will be searched for and replaced by the file named in the New template script file.

New template script file
This file should be a version of the file that provides the templating with any needed changes. The default for this is a modified version of underscore.js, which has a change that modifies the characters that are used for the encoding the HTML. This file will be copied over the the file specified in the Template script to replace input.

Original templating strings
These are a column separated list of the characters that are used for the templating. The defaults are the values that are used by default by the underscore templating. If you use a different style of formatting you would set the list of characters separated by a commas. The default for this library do not work well with our server-side processing.

Original templating strings
These are a column separated list of the characters that will be used for the templating. The defaults are the values that are based on mustache style templating, which works well with our server-side processing.

After running the wizard the model is created and the files for the application can be found in the /samples/apps/{name} directory for your customization.

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.

Library References
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);
});
```

Rest Data Service Support
This is an input where you can add js code to page to reference any selected data service and variable name. Similar to the Service Provider inputs but with this input you will need to add the data services to the model yourself.

Disable Smart Refresh
If selected, this sets the setting for smart refresh in the theme to turn off smart refresh. This is needed if your default theme included dojo which sometimes interferes with other script libraries.

Default RDD File
The name of a RDD file in the project that you want used by pages created by page automation. This can be used to choose a Rich Data Definition that does not have dojo widgets, which can interfere with your selected libraries.

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