Script Portlet Samples for IBM Script Portlet

Overview

This is a set of examples that illustrate some basic techniques for using the IBM Script Portlet for WebSphere Portal. The samples show how to use some Portal features such as Public Render Parameters, and there are examples of using open source libraries.
To use these examples you will need to have WebSphere Portal 8.0.0.1 or later with the IBM Script Portlet installed. These samples were built and tested with the IBM Script Portlet release 1.3 from April 2015. See the References section below for information on obtaining IBM Script Portlet.

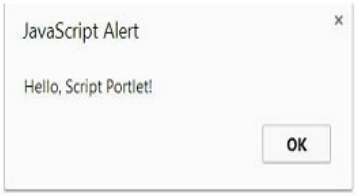
The samples include:

- Hello, World
- jQuery Eventing (two cooperating portlets)
- jqPlot Chart
- jQuery DataTables
- Load WCM Content
- Launch Script
- Public Render Parameters (two cooperating portlets)
- WCM tag samples
- Imported content files
- Portlet Preferences
- Media File Sample
- Angular Contacts Sample

There are instructions for using the sample code near the end of this document.

Hello, world

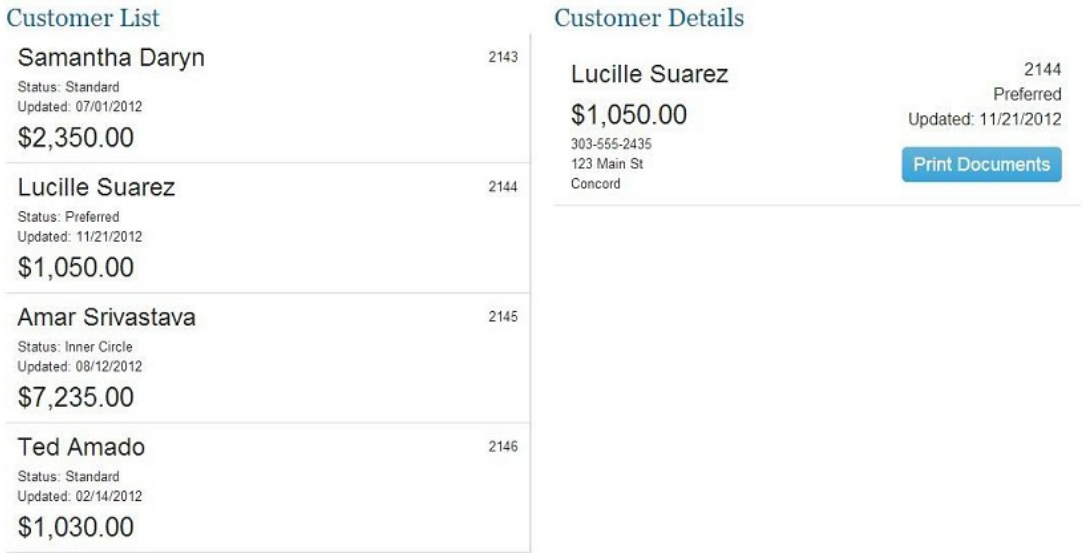
This is an extremely simple HTML snippet and JS function. Clicking the button shows a JS alert popup.



jQuery Eventing

This sample uses jQuery eventing, with bind() and trigger() functions. A list of customers is defined in the Customer List portlet, and a click event on each customer is used to fire the event by calling trigger() with a queue object. In the details portlet, the bind() function is called using the same queue object. When the event is received, it looks up a customer ID from the customersData array and updates the display.

To use this sample, put two Script Portlets on the page side-by-side. In the left-hand portlet use the event_source code, and in the right-hand portlet use the event_target code.



jqPlot Chart

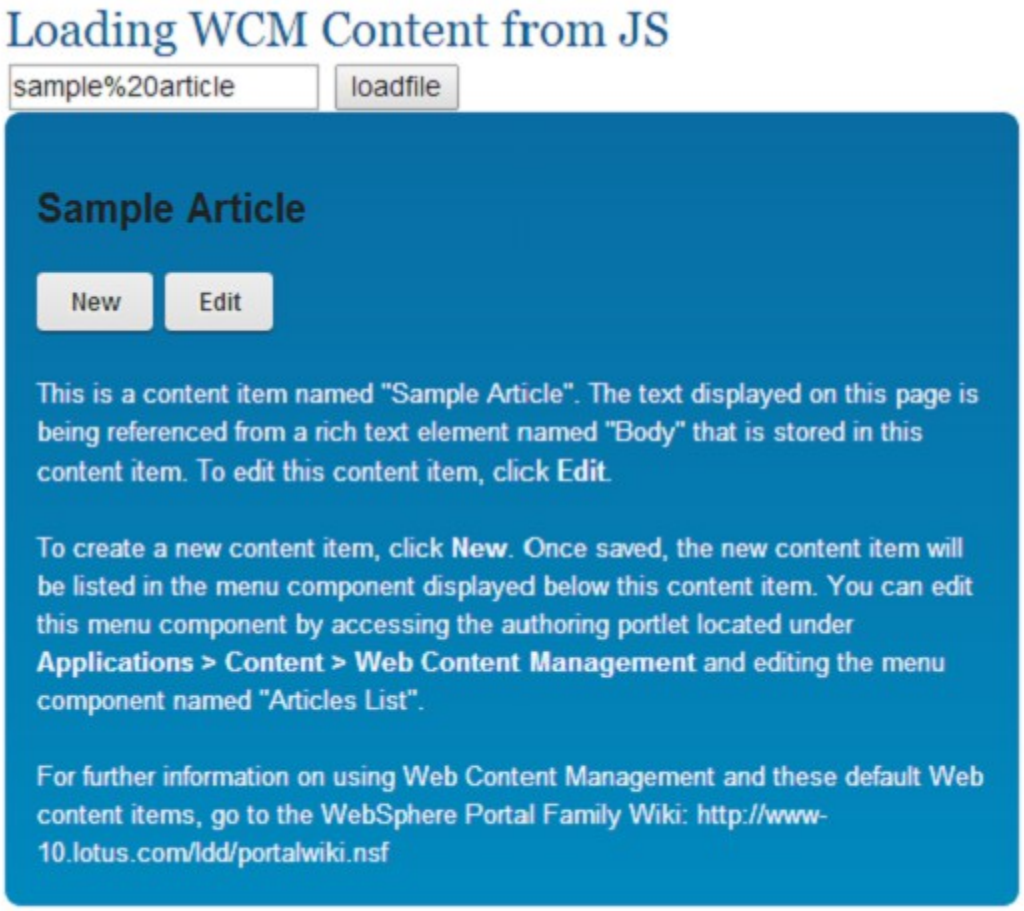
This example show a chart created with the jQPlot open source charting library. The data for this chart in this example is defined right in the JS code. In a real world implementation the data would come from a REST service.

jQuery DataTables

This sample shows the jQuery DataTables library. The Customers data in this case comes from a JS variable. See this web site for more information on the jQuery DataTables library: https://datatables.net/. This sample also has a simple example of using the Script Portlet namespace tag __SPNS__. Using this tag helps you make sure your portlet is independent of other portlets or multiple instances of the same portlet on a page. It can be used in your Html for making sure your id's are unique, in your css if you are using selectors that are based on an item id and in your js to make sure your functions and data are uniquely named to prevent collisions.

Load WCM Content

This example shows how to load any WCM content via Ajax. In this example, the url points to the web content library and the "articles" site area. You can type any content in that site area and this script will load it and display it in a div.

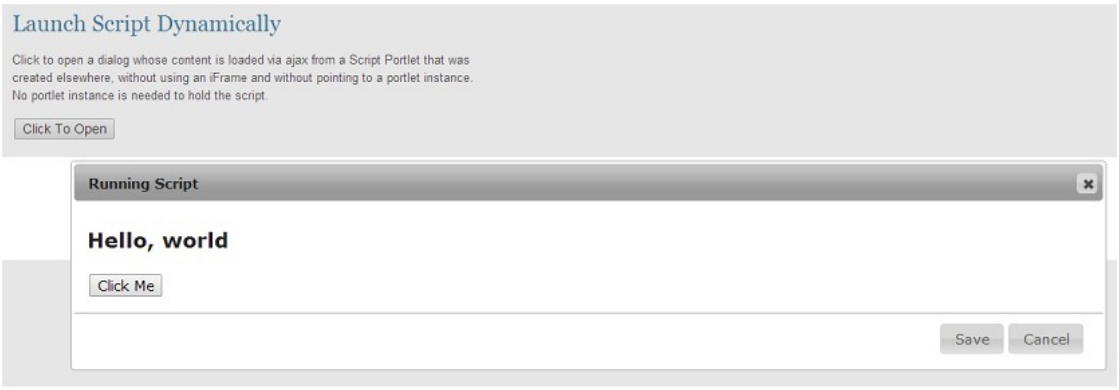


Launch Script

This example shows how to get the contents of a script via Ajax, and how to take that content and render it in a dynamically-created modal dialog. This can be useful when you want to launch dialogs/applications programmatically and you don't want to use iFrames, for example if you need to support mobile devices. You could also use the WCM Menu Component to pick which scripts should be displayed based on a Personalization rule.

NOTE: To run this sample you must update the URL in the JavaScript to the URL to a script content item in your WCM library. For example, the screenshot shows the "Hello, world" sample Script Portlet launched in a popup dialog. There are a few ways of finding the URL for a Script Portlet Content Item. One way is to use a browser tool to get the URL from the "src" attribute of the iFrame which is used by the preview window of the

Script Portlet Editor. Another way is to use the public WCM URL tag, or you could use the public WCM servlet rendering URL format.



Public Render Parameters

This sample shows how to set and retrieve a Public Render Parameter (PRP). The parameter that's used is the pre-defined "CUSTOM_CONTEXT" parameter defined by the WCM Rendering Portlet. The top portlet shows a list of customer names that are defined in a JS variable and used to populate a SELECT list. When the form is submitted it sets the PRP value. The bottom portlet then retrieves the PRP value and uses it to select the customer from the list.

To use this sample, put two Script Portlets on the page, one above the other. In the top portlet use the prp_set code, and in the bottom portlet use the prp_retrieve code.

WCM Tag Samples

This sample shows how to use some of the WCM tags that are available. For example, you can retrieve the current user's name, or get a "namespace" value that will be unique for this instance on a page.

WCM Tags Sample

Tag	Example Value	Notes
User display name	Jonathan Booth	Uses a combination of two EL beans
LDAP cn	Jonathan Booth	Common Name from LDAP
LDAP sn	Booth	Surname from LDAP
LDAP uid	jbooth	UID from LDAP
LDAP preferredLanguage	en_US	Preferred language from LDAP
Authors	wpsadmin	Authors of this Script Content Item
Title	Script Portlet Content Item_3-WG	Title of this Script Content Item
Parent Title	Portal Context Sample	Title of the parent of this Script Content Item
Portlet Namespace	ns_Z7_N0LA11O0KO22F0AS8JOEJK3876_	A unique identifier for this portlet/page
Locale	en-US	Current rendering locale
Display only in page edit mode	Edit mode is ON	The value column is only shown when this page is in edit mode.

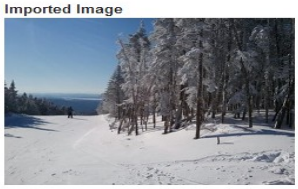
Imported Content Files

This sample shows the use of imported content: an image, a JSON data file, and an HTML fragment. All are loaded via Ajax calls. Each of these files is referenced with a relative link in the index.html page, and each relative link is replaced by a WCM Element tag when imported. The WCM element name is the same as the relative pathname. The JSON file and the HTML fragment are loaded by the application JavaScript code. The URLs are read from the hidden "a" tags that are in the index.html file.

The Javascript that loads the HTML fragment and the JSON file may not run standalone (from the file system) from all browsers due to security restriction.

Imported Content Files

Imported Image



HTML Fragment Loaded by JS

This text comes from a separate HTML fragment file, dynamically loaded into the main page by Javascript running in the browser.

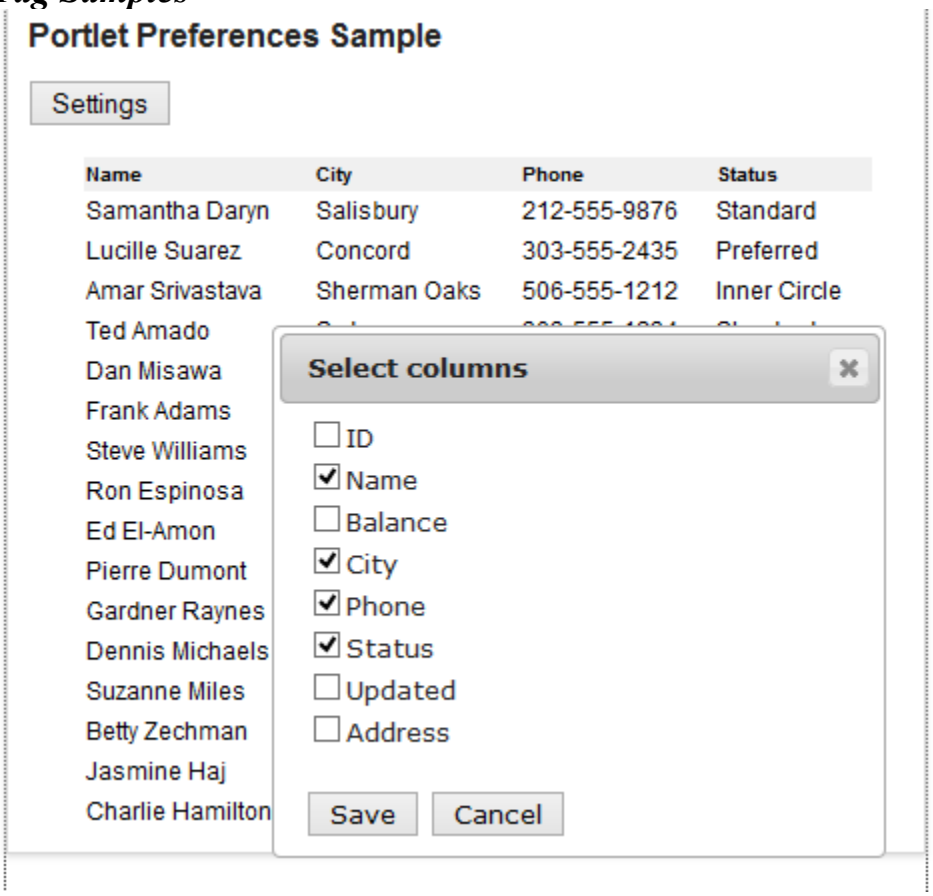
JSON Data Loaded by JS

Name	Address	City
Samantha Daryn	2001 West Rd.	Salisbury
Lucille Suarez	123 Main St	Concord
Amar Srivastava	South Mill Pond Ave.	Sherman Oaks
Ted Amado	44 Center St.	Sydney
Dan Misawa	1204 Mountain Boulevard	Canterbury
Frank Adams	133 Rock Boulevard	Denver
Steve Williams	1 Circle Lane	San Jose
Ron Espinosa	2425 Mountain Rd.	Dublin
Ed El-Amon	26B Seiching Road	Beijing

Portlet Preference Files

This sample shows the use of preference js api. The data that is read from a JSON file is displayed as a table using Jquery. The columns that are displayed in the table are controlled by preference values that are stored in the portlets preferences. This allows a user to use the same portlet to display different data depending on the preferences that are set for that instance of the portlet. For this sample the preferences are an array of strings that indicate weather a column is visible or not. The preference api supports the setting different kinds of preferences from simple strings, js arrays or JSON objects, it is up to you to identify the type of data your portlets need. Every script portlet can have preferences as needed and the preference definition can be customized to the specific instance. This sample also has a more complex example of using the Script Portlet namespace tag __SPNS__. Using this tag helps you make sure your portlet is independent of other portlets or multiple instances of the same portlet on a page. It can be used in your Html for making sure your id's are unique, in your css if you are using selectors that are based on an item id and in your js to make sure your functions and data are uniquely named to prevent collisions.

WCM Tag Samples



Media File Sample

This sample shows how you can reference other file types such as video. The portlet include a reference to a video file and automatically plays this video when the portlet's page is refreshed. The media files are now imported into your script portlet. Prior to version 1.3 this was not so, a new feature was added that allows you to control the files that are imported into your portlet. The way to modify this is described in our documentation. The property info that controls other file types is found in these settings.

# List of other types of binary based files to be added by import(case insensitive)
scriptportlet.import.extensions.OtherBinary=mp4,mp3,wav,ogg

This is the default which now allows these files to be imported. To change these you would follow the instructions in the documentation. One thing to note these are treated as binary files and so there is no processing or fixing up stored information. So for files that might contain references like swf files which might contain references to other swf files, these references are not changed as the files are imported. These binary files are not necessarily media files but can be any type of file you want imported as binary data.

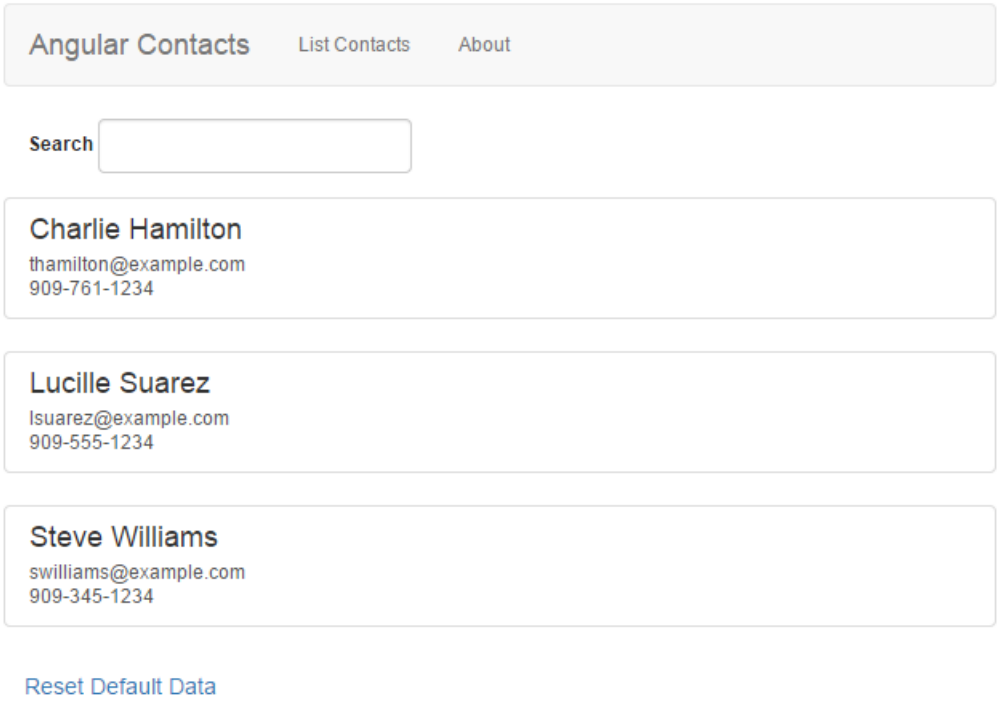
Angular Contacts Sample

This is an example of a simple application built with the AngularJS framework and the Bootstrap CSS library. It can run standalone by loading index.html, and it can be imported or pushed into a Script Portlet. It's an example of a "single page application" where the different views within a single index.html page are dynamically loaded by the AngularJS framework. When used in a Script Portlet, a single page application like this is displayed as one portlet on a portal page, typically alongside other portlets.

The application is a simple contact list application for viewing and updating a list of contacts. Here are the key features illustrated in the sample:

- The four different views (list, details, update, and about) are separate HTML files loaded as partial pages using the AngularJS $routeProvider service, configured in app.js.

- An AngularJS service named "contactsLocalStorageService" is used to retrieve and update a list of contacts. Data initially comes from a JSON file but after that is stored using HTML5 local storage. It can be reset to the file data by clicking "Reset Default Data."
- The AngularJS $http service is used to load the default JSON data file, contacts.json.
- There are three simple AngularJS controllers used, for the list view, the details view, and the update view.
- There are three local JS files used, and when running in Script Portlet they are combined into a single resource by WebSphere Portal's resource aggregation feature (available in version 8.5, CF03 or later). See the comment at the top of index.html for how this is enabled.
- The Bootstrap library is used for styling of the application UI.



How to use the sample code

First, unzip the script_portlet_samples.zip file onto your computer. Each sample has a folder that contain the files for one Script Portlet application. For each application there is a main index.html file, at least one a JavaScript file, and other application files such as CSS. There is also a .zip file containing all the files for the application.

For the samples that have no dependency on WebSphere Portal, you can actually test the application on its own simply by using a web browser to view the index.html file on your file system. For example, the jqPlot Chart and jQuery DataTables samples can be tested

in this way. Then you can create Script Portlets on your Portal server for each sample as described next.

To use the code for these samples there are two basic approaches: importing an individual sample .zip into an existing Script Portlet instance, and using command line "push" to install all the applications at once into a WCM library. With Portal 8.5 the command line approach is the simplest way to make all the samples easily available using the site toolbar. For Portal 8.0.0.1 you can also push all the samples at once, but the toolbar functionality is not available.

To use the command line approach to add all the samples at once:

1. You should have your server information configured in the sp-config.json file where you unzipped the command line support (sp_cmdln.zip). You should also have the root folder of the command line support in your system path.
2. In a command window, go to the root directory for the samples and run the "push_all_samples" command.

Once you have pushed them to your server they will be in the site area specified in the main sp-config.json file (Script Portlet Library/Script Portlet Applications by default).

Then to use the samples, if you are on Portal 8.5 you can use the site toolbar:

3. Go to the Portal page where the portlet will be and go into Edit mode for the page.
4. Click on the Create button to open the toolbar, choose the Content tab, and click on Script Portlet Applications.
5. Find the sample you want, and either click the sample and the Add button or drag it onto your portal page.

If you are using Portal 8.0.0.1 you will need to use the "Import" button to copy from the library:

3. Go to the Portal page where the portlet will be and go into Edit mode for the page.
4. Add an empty Script Portlet to the page by selecting it from the Applications tab of the toolbar.
5. Click on the Import button, then in the popup dialog box select the "Copy Existing Content" tab.
6. Find the sample you want to use and click the "Copy" button.

If you want to import individual .zip files instead of using the command line batch approach here are the steps:

1. Go to the Portal page where the portlet will be and go into Edit mode for the page.
2. Add an empty Script Portlet to the page by selecting it from the Applications tab of the toolbar.
3. Click on the import button.
4. Select the .zip file you want from the samples folder and click the "Import" button.

Note there is a difference between linking to a Script Portlet content item in a library and copying it. If you have a link, any future changes to that content item affect all instances of portlets that are linked to it. With a link it also means that if the original is deleted, all the linked portlets no longer have any content. If you have a copy and then update the original content item is updated, you would need to copy it again in order to get the updated version.

You should then see the sample running in the Portlet.

References

The Script Portlet is available as a download on the Portal Catalog here: IBM Script Portlet for WebSphere Portal