Create your own widget ecosystem on IBM Bluemix. This article focuses on building Bluemix widgets, the developers creating them, and how they can deliver new tools to line-of-business users. It briefly describes the dynamism between experience creators and experience composers. Both Bluemix and WebSphere Commerce have ecosystems of apps being developed daily. This article describes the beginning of an ecosystem merge, benefiting both your experience composers and your experience creators.

IBM Bluemix changes the way businesses develop applications. The agility, capability, and connectable ecosystem are simply tools that empower developers to not only do their work, but to be creative, innovative, and productive as well.

IBM WebSphere Commerce feature pack 7 onward empowers line-of-business users to do more and do it easily. It enables marketers and merchandisers to compose shopper experiences using a tool belt of widgets (or applications). Your business (not IT) can build pages using WebSphere Commerce Composer, mixing and matching out-of-the-box widgets, third-party widgets, home-grown widgets, and now Bluemix widgets.
Figure 1. Interplay of experience creators and experience composers

Concept

Your business needs to move fast to stay ahead of competition and cater to the changing needs of your customers. Often you need to make build or buy decisions. From a developer perspective, whether software is built or purchased, there is always the question of integration -- how we integrate software is integral to business success. Bluemix can help you meet business demands in several ways. In this article, we discuss evolving your widget ecosystem. We encourage you to sign up for a free Bluemix trial so you can not only experiment, but more importantly browse the catalog and envision how this can help you and your business.

Figure 2 shows how line-of-business users create pages using WebSphere Commerce Composer. Your business user selects the page they want to design, chooses a template, and adds widgets to the page. Each widget has different capability. Currently you can extend the widget palette to include any capability that you build. Commerce Composer also allows you to plug in third-party widgets. This article shows you how to extend Commerce Composer to pull widgets and capabilities built on Bluemix. Moreover, developers can use Bluemix capabilities to develop new applications fast and deliver them into the hands of business users without them knowing how you built it. By separating your widget infrastructure into smaller composable units, you can deliver changes to them quicker and independent of your core platform.
Figure 2. Creating pages with WebSphere Commerce Composer

Architecture

Figure 3 is a visual representation of the architectural concepts described in this article. The WebSphere Commerce Management Center interacts with the Bluemix Widget Registration service to get a listing of available widgets. The Aurora storefront uses widgets that run on Bluemix. Our sample Bluemix applications are written in Node.js, which is a trending runtime environment.
Figure 3. Architecture

What you will need

- A Bluemix account
- Knowledge of WebSphere Commerce development
- WebSphere Commerce 7 Feature Pack 7 with Aurora published

Developer details

First, you need to build a widget registration service on Bluemix. This is the centralized registry that WebSphere Commerce Composer communicates with to find available Bluemix widgets. Figure 4 illustrates the business experience. When a business user selects to add a widget to a page, this window appears with a mixture of local widgets and Bluemix widgets. Next, create a sample widget (application) on Bluemix to get you started. Finally, learn how to customize WebSphere Commerce Composer to create the window in Figure 4.
Figure 4. Available widgets

Dataflow pattern

To get widgets, the request initiates from the WebSphere Commerce Server (see Figure 5). A REST request is made to the Bluemix Widget Registry service. The registry only knows a list of widgets to include and a URL to get additional details.

Figure 5. Dataflow

The widget registry loops through and queries each widget to get further details about them. This allows each widget to change its description, endpoint, and parameters independently of
the widget registry. The widget registry consolidates all widget responses and returns a result to WebSphere Commerce. The following is an example of what the Stock widget returns to the Widget registry service. The service wraps all these responses into a JSON array.

```json
{
    "widgetName": "Stock Ticker Widget",
    "description": "Shows the current value of a stock symbol",
    "widgetRestrictionGroups": "AnyPage,CatalogEntryPage",
    "widgetObjectType": "StockTicker",
    "jspEndPoint": "http://stockticker.mybluemix.net/init",
    "widgetProperties": "http://stockticker.mybluemix.net/properties",
    "controlProperties": "ticker=IBM"
}
```

Deploying the Bluemix Widget Registry

The Widget Registry has a RESTful API design. It is constructed using Node.js because it is an effective tool for developing APIs quickly and easily.

To create your own version, fork the code located in the IBM DevOps repository:

1. Navigate to the URL [https://hub.jazz.net/project/madeluca/article1-widget-registration](https://hub.jazz.net/project/madeluca/article1-widget-registration).
2. Click **Edit Code**.
3. Click **FORK** to create your own copy.

   **Figure 6. Save a copy of the code window**

4. When your repository is created, click **DEPLOY** and configure your Bluemix deployment properties.
5. Select a Target, Organization, Space, Application Name, and Host. Make sure you change the default value for the host name. The host name represents a portion of the URL used to access the application. For example, if the host name is "mystockapp" the URL to access the application is http://mystockapp.mybluemix.net/.

Log into Bluemix and you should now see the application on your dashboard. Click the launch application icon shown in Figure 6.

The result from launching the application should look like this:

```json
{
"recordSetCompleteIndicator": "false",
"recordSetReferenceId": "ef01c590-03a7-11e4-ab6b-82125384c64b:31",
"recordSetStartNumber": "0",
"recordSetCount": 0,
"recordSetTotal": 0,
"objects": []
}
```
No results are returned because you have not yet created any widgets -- in the next step, you'll create a sample widget. If you open the app.js file using the IBM DevOps Services Web IDE, you will notice the line:

```javascript
var widgets = [['stockticker-tst.mybluemix.net/widgetRegistration']];
```

You need to configure that line to specify the widgets you want the registry to look up. Specify the Bluemix URL for the Stock application that you create in the next section.

**Deploying the Stock application on Bluemix**

These steps show how to create a Bluemix application that acts as a widget in WebSphere Commerce Composer. The example code is a stock ticker application that acts as a shell for any other application you may want to create.

To create your own version, fork the code located in the IBM DevOps repository:

1. Navigate to the URL [https://hub.jazz.net/project/madeluca/article1-stockticker](https://hub.jazz.net/project/madeluca/article1-stockticker).
2. Click **Edit Code**.
3. Click **FORK** to create your own copy.
4. Click **DEPLOY** or **DEPLOY AS** to open the Configure Application Deployment window.
5. Select a Target, Organization, Space, Application Name, and Host. Make sure you change the default value for the host name. The host name represents a portion of the URL used to access the application. For example, if the host name is "mystockapp" the URL to access the application is http://mystockapp.mybluemix.net/.

**Test your deployments**

To test that the applications are both running properly, navigate back to your Widget Registration application (Figure 6) and you should find your widget listed.

You can also test the application direction with the URL http://<stockticker-tst>.mybluemix.net/init?ticker=IBM.

**Configuring Commerce to call the Bluemix Widget Registry**

This section requires WebSphere Commerce development skills. In WebSphere Commerce, configure a WidgetRegistry widget to use as a place holder for all Bluemix widgets. Figure 4 shows the business experience you can accomplish after finishing this task.

1. Download the sample code for this section located at [https://hub.jazz.net/project/madeluca/article1-commerce](https://hub.jazz.net/project/madeluca/article1-commerce) and unzip the file to the directory c:\Commerce. Locate the subdirectory c: \Commerce\Bluemix-WidgetRegistry. Deploy the files from this subdirectory. Note that you need to override some files marked restricted. Future upgrades may overwrite these files and you will then have to reapply these changes.
2. Stop the WebSphere Commerce server.
3. Dataload the new widget:
   1. Open the file c:\DataLoad\dataload\common\widget\StoreConfiguration.csv and edit the endpoint URL. This is the location of your widget registry on Bluemix.
2. Open a command prompt and type:
   
   cd c:\WCDE_INT70\bin

3. Dataload the widget with the command:
   
   dataload c:\DataLoad\dataload\common\widget\wc-dataload.xml

4. Update the Stores project. Copy the Stores directory to your development environment Stores directory, WCDE_INT70\workspace\Stores.

5. Update the LOBTools project:
   1. Do not overwrite any files. Duplicate files need to be merged manually. The file NewPageLayoutWidgetDialog.lzx is marked with delimiters that say "Start custom code" and "End custom code".
   2. Copy all the files in c:\LOBTools to your development environment directory WCDE_INT70\workspace\LOBTools.
   3. Open the file WCDE_INT70\workspace\LOBTools\WebContent\WEB-INF\web.xml. Add the new struts reference /WEB-INF/struts-extension-custom.xml after the other struts entries.
   4. Optionally, you can filter out the BluemixRegistryWidget so business users cannot select the empty widget. Open the file WCDE_INT70\workspace\LOBTools\WebContent\jsp\commerce\pagelayout\restricted\GetPageLayoutWidgetDefinitions.jsp. Surrounding the <object> tags, add the code:

   ```
   <c:if test="${widgetDefinition.widgetObjectName!='BluemixRegistryWidget'}">..</c:if>
   ```

6. Build, restart, and test:
   1. Open your WebSphere Commerce Development Environment.
   2. Right-click the LOBTools project and select Refresh. Right-click the LOBTools project again and select Build OpenLaszlo Project.
   3. Start the WebSphere Commerce server.
   5. Create a new product page and add the Bluemix widgets you created. Figure 7 is an example of how you might configure your page.
Figure 9. Store page with a Bluemix stock widget

Conclusion

This article described the business reasons for using Bluemix and demonstrated an approach to integrating Bluemix with WebSphere Commerce. We welcome your comments and innovation experience. Log in to Bluemix and get started!
Resources

- To deploy the **Bluemix Widget Registry**, you'll need to fork the code in the IBM DevOps repository.
- To deploy the Stock application on Bluemix, you'll need to fork the **stock ticker widget** code in the IBM DevOps repository.
- To configure Commerce to call the Bluemix widget registry, download the sample code in the IBM DevOps repository.
- Learn more about **WebSphere Commerce Composer** in the online documentation.
About the authors

Marco Deluca

Marco Deluca is a Software Architect with the WebSphere Commerce Development team at the IBM Toronto Lab, Canada. He brings over ten years of progressive experience to the enterprise software domain. Currently, Marco is responsible for architecting, designing and developing solutions for WebSphere Commerce.

David Finn

David Finn is a Software Developer Intern at the Dublin, OH lab. He is a senior at The Ohio State University studying Electrical and Computer Engineering. As an IBM intern, he has worked to demystify Bluemix in its early stages and demonstrate its potential for IBM software such as WebSphere Commerce.

Octavio Roscioli

Octavio is a Software Engineer Intern at the Dublin, OH IBM campus. He has been working on integration of on-premise applications on Bluemix with his partner, David Finn. His expertise resides mainly in web development. Octavio is a senior at Vanderbilt University.

© Copyright IBM Corporation 2015
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)