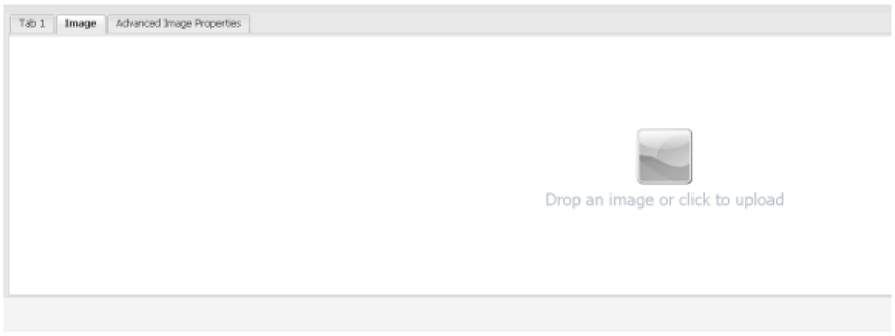
Adobe CQ Help / Create a CQ widget that supports image drag and drop

What's covered

Article summary

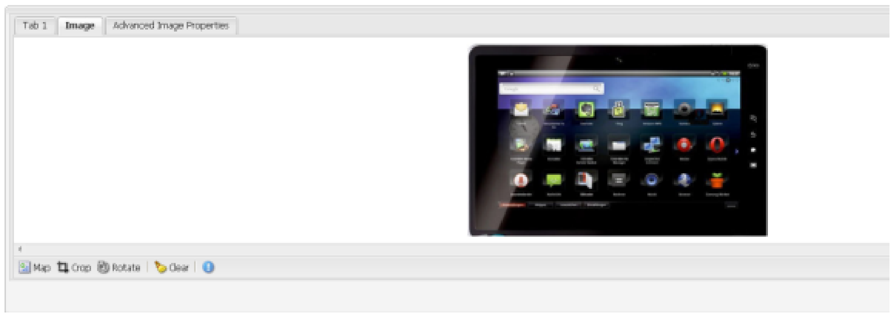
Summary	Discusses how to develop a CQ component that the parsys system uses. The component supports a dialog box with multiple tabs, supports an image, and hooks into the functionality offered by the Content Finder.
Digital Marketing Solution(s)	Adobe Experience Manager (Adobe CQ)
Audience	Developer (intermediate)
Required Skills	JavaScript, JCR, Sling
Tested On	Adobe CQ 5.5

You can create an AEM component that manages text and an image. Also, the component that you learn to create in this article has a CQ dialog box that lets a user select an image during design time.



An AEM widget that lets an author select an image during design time

When the user selects an image, it is displayed in the widget.



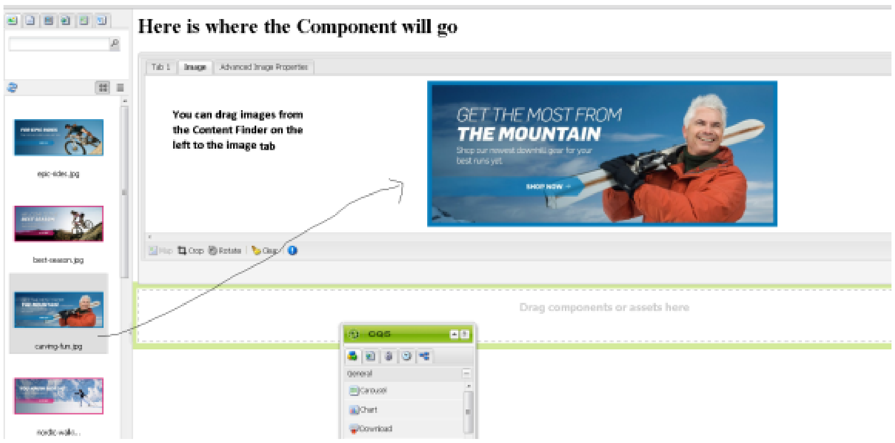
An image is displayed in the AEM widget

In this article, the image is editable by a content author and can be dragged from the Content Finder.

For this article, the image:

- Is a Dialog Widget (xtype of smartimage).

- Must be configured in the Component's `cq:editConfig` to allow for dragging images from the Content Finder, as shown in this illustration.


Here is where the Component will go
Tab 1
Image
Advanced Image Properties
You can drag images from the Content Finder on the left to the image tab
GET THE MOST FROM
THE MOUNTAIN
Drag components or assets here
CQ5

The component supports dragging from the CQ Content Finder

The rich text:

1. Is a Dialog Widget (xtype of richtext).
2. Widget enables all the features of the rich text editing plugin table.
3. Widget has default text.

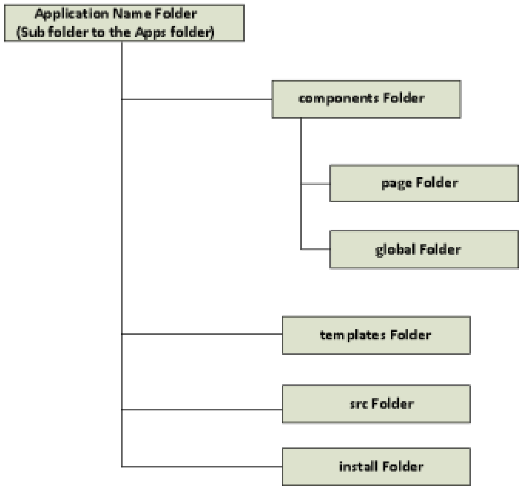
The path of a page:

1. Is a Design Dialog Widget, most likely an xtype of pathcompletion.
2. Widget should have property regex with a regular expression validating the user's input (for example, "/^\/content\/training\/(.)*$/").
3. Widget should have property regexText with an error message if regular expression fails.

Create a CQ application folder structure

To the top

Create an Adobe CQ application folder structure that contains templates, components, and pages by using CRXDE Lite.



A CQ application folder structure

The following describes each application folder:

- **application name**: contains all of the resources that an application uses. The resources can be templates, pages, components, and so on.
- **components**: contains components that your application uses.
- **page**: contains page components. A page component is a script such as a JSP file.
- **global**: contains global components that your application uses.
- **template**: contains templates on which you base page components.
- **src**: contains source code that comprises an OSGi component (this development article does not create an OSGi bundle using this folder).

- **install**: contains a compiled OSGi bundles container.

To create an application folder structure:

1. To view the CQ welcome page, enter the URL http://[host name]:[port] into a web browser. For example, http://localhost:4502.

2. Select CRXDE Lite.

3. Right-click the apps folder (or the parent folder), select Create, Create Folder.

4. Enter the folder name into the Create Folder dialog box. Enter **imagecomponent**.

5. Repeat steps 1-4 for each folder specified in the previous illustration.

6. Click the Save All button.

Note: You have to click the Save All button when working in CRXDELite for the changes to be made.

Create a template

To the top

You can create a template by using CRXDE Lite. A CQ template enables you to define a consistent style for the pages in your application. A template comprises of nodes that specify the page structure. For more information about templates, see http://dev.day.com/docs/en/cq/current/developing/templates.html.

To create a template, perform these tasks:
1. To view the CQ welcome page, enter the URL http://[host name]:[port] into a web browser. For example, http://localhost:4502.
2. Select CRXDE Lite.
3. Right-click the template folder (within your application), select Create, Create Template.
4. Enter the following information into the Create Template dialog box:

- **Label**: The name of the template to create. Enter **templateImage.**
- **Title**: The title that is assigned to the template.
- **Description**: The description that is assigned to the template.
- **Resource Type**: The component's path that is assigned to the template and copied to implementing pages. Enter **imagecomponent/components/page/templateImage**.
- **Ranking**: The order (ascending) in which this template will appear in relation to other templates. Setting this value to 1 ensures that the template appears first in the list.

5. Add a path to Allowed Paths. Click on the plus sign and enter the following value: /content(/.*)?.
6. Click Next for Allowed Parents.
7. Select OK on Allowed Children.

Create the Image component

To the top

Components are re-usable modules that implement specific application logic to render the content of your web site. You can think of a component as a collection of scripts (for example, JSPs, Java servlets, and so on) that completely realize a specific function. In order to realize this functionality, it is your responsibility as a CQ developer to create scripts that perform specific functionality. For more information about components, see http://dev.day.com/docs/en/cq/current/developing/components.html.

By default, a component has at least one default script, identical to the name of the component. To create a render component, perform these tasks:
1. To view the CQ welcome page, enter the URL http://[host name]:[port] into a web browser. For example, http://localhost:4502.
2. Select CRXDE Lite.
3. Right-click /apps/imagecomponent/components/page, then select Create, Create Component.
4. Enter the following information into the Create Component dialog box:

- **Label**: The name of the component to create. Enter templateImage;
- **Title**: The title that is assigned to the component.
- **Description**: The description that is assigned to the template.

5. Select Next for Advanced Component Settings and Allowed Parents.
6. Select OK on Allowed Children.
7. Open the templateImage.jsp located at: /apps/imagecomponent/components/page/templateImage/templateImage.jsp.

8. Enter the following JSP code.

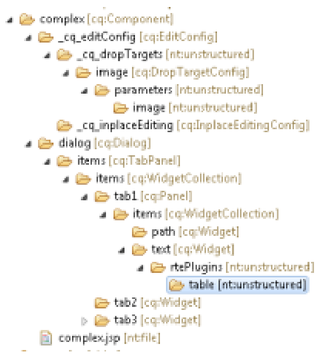
```
<%@include file="/libs/foundation/global.jsp" %>
<cq:include script="/libs/wcm/core/components/init/init.jsp"/>


<body>
<h1>Here is where the Component will go</h1>
<cq:include path="par" resourceType="foundation/components/parsys" />
</body>
</html>
```

Note: This the page component where you will add the image component that you create in this development article. The remaining part of this article talks about how to create the component that allows you to use an image.

Develop the component that support drag and drop of images

To the top

Now you create the complex component that supports all the requirements that are listed at the start of this article. The following illustration shows the structure of the component that is created in this section.



```
complex [cq:Component]
  _cq_editConfig [cq:EditConfig]
    _cq_dropTargets [nt:unstructured]
      image [cq:DropTargetConfig]
        parameters [nt:unstructured]
          image [nt:unstructured]
    _cq_inplaceEditing [cq:InplaceEditingConfig]
  dialog [cq:Dialog]
    items [cq:TabPanel]
      items [cq:WidgetCollection]
        tab1 [cq:Panel]
          items [cq:WidgetCollection]
            path [cq:Widget]
            text [cq:Widget]
              rtePlugins [nt:unstructured]
                table [nt:unstructured]
        tab2 [cq:Widget]
        tab3 [cq:Widget]
  complex.jsp [nt:file]
```

The structure of the complex component that is created in this section

Perform these tasks using CRXDE:

1. Right click on /apps/imagecomponent/components and then select New, Component.

2. Enter the following information into the Create Component dialog box:

- **Label**: The name of the component to create. Enter **complex**.
- **Title**: The title that is assigned to the component. Enter **Training Complex**.
- **Description**: The description that is assigned to the template. Enter **Complex Component**.
- **Super Resource Type**: Enter **foundation/components/parbase**.
- **Group**: The group in the side kick where the component appears. Enter **Training**.
- **allowed parents**: Enter ***/*parsys**

3. Open complex.jsp and enter the following JavaScript code:

```
<%@include file="/libs/foundation/global.jsp"%>
<%@ page import="com.day.cq.wcm.foundation.Image" %>
<%
// getting the image from the Dialog/resouce
// notice the use of the second parameter "image" -- this signifies that the
// image will live on a resource (called image) below the requested resource
Image image = new Image(resource, "image");
// setting the selector so that the "parbase" can work its magic
image.setSelector(".img");
// getting the rich text from the Dialog
String text = properties.get("text", "TEXT NA");
// getting the path from the Design Dialog
String path = currentStyle.get("path", "PATH NA");
%>
<h2><%= path %></h2>
<%= text %>
<%
image.draw(out);
%>
```

4. Add a Dialog to the complex component. Select **/apps/imagecomponent/components/complex**, right click and select New, Dialog.

5. Create an `items` node (nodeType `cq:WidgetCollection`) under the tab1 node of your Component's Dialog.

6. Create a `text` node (nodeType `cq:Widget`) under the newly created `items` node.

7. Assign the following properties to the newly created `text` node:

Name	Type	Value	Description
name	String	./text	Defines where content is stored.
xtype	String	richtext	Defines the Widget type
hideLabel	Boolean	true	Specifies to hide the label of the Widget
defaultValue	String	This is the default text	Defines the default text value.

8. Create an `rtePlugins` node (nodeType nt:`unstructured`) under the newly created text node.

9. Create a `table` node (nodeType `nt:unstructured`) under the newly created `rtePlugins` node.

10. Assign the features property (Type = `String`, Value = `*`) to the newly created `table` node.

11. Copy the nodes tab2 and tab3 under the node `/libs/foundation/components/textimage/dialog/items`. Paste to the complex Component's Dialog so that they are a peer of tab1 (for example, `/apps/imagecomponent/components/complex/dialog/items/items`).

The tab2 is the smartimage tab and tab3 is advanced image properties. It is often more efficient to copy-and-paste existing Dialogs/Widgets that meet your needs. It is recommended that you review and understand what you have just copied to better understand the internal workings of CQ.

If you examine closely enough, you will see the Widget is actually storing image related content at a level deeper than current resource (for example, ./image/file, ./image/fileReference, etc.). This corresponds to the code in complex.jsp (Image image = new Image(resource, "image");).

12. Create an `items` node (nodeType `cq:WidgetCollection`) under tab1 node of your Component's Design Dialog.

13. Create a `path` node (nodeType `cq:Widget`) under the `items` node.

14. Assign the following properties to the `path` node:

Name	Type	Value	Description
name	String	./path	Defines where content is stored
xtype	String	pathfield	Defines the Widget type
fieldLabel	String	Enter a Path	Defines the label applied to the Widget
rootPath	String	/content/trainingSite	Defines the root path to display
regex	String	/^\/content\/trainingSite\/(.)*$/	Defines the regular expression used to evaluate user input
regexText	String	Please insert a Page under /content/trainingSite	Define the error message if a user's input fails the regular expression

15. Copy the node /libs/foundation/components/textimage/cq:editConfig. Paste to the root node of your complex Component (/apps/imagecomponent/components/complex).

This action enables drag-and-drop capabilities from the Content Finder.

In order to be able to drag-and-drop assets from the Content Finder to a component on a Page, there must be a drop targets configuration node called `cq:dropTargets` (of type `nt:unstructured`) below the edit configuration node (`cq:editConfig`) of a component.

16. Using CRXDE, navigate to this path:

`/apps/imagecomponent/components/complex/cq:editConfig/cq:dropTargets/image`

Validate that the image node has the following properties:

- **accept** (Type = String) - the media types to be accepted (e.g. image/.*, etc.)
- **groups** (Type = String) - the groups in the Content Finder assets can be accepted from (e.g. media)
- **propertyName** (Type = String) - the property the reference should be stored

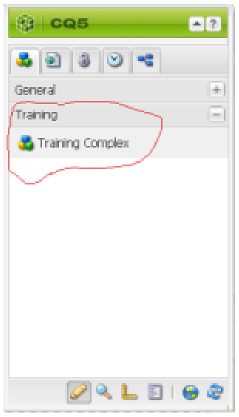
(e.g. ./image/fileReference)

17. Navigate to the parameters node that is a child to the image node. Change the value of the `sling:resourceType` property to have the value: `training/components/complex`. The `sling:resourceType` property should reference the location to find a rendering script.

Create a CQ web page that uses the new component

To the top

The final task is to create a site that contains a page that is based on the templateImage (the template created earlier in this development article). This CQ page will let you select the complex widget that you just created from the CQ side kick, as shown in the following illustration.



The CQ sidekick displaying the Training Complex component that is created in this development article

1. Go to the CQ welcome page at http://[host name]:[port]; for example, http://localhost:4502. Select Websites.

2. From the left hand pane, select Websites.

3. Select New Page.

4. Specify the title of the page in the Title field.

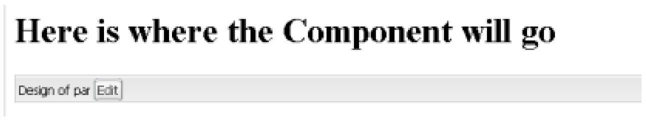
5. Specify the name of the page in the Name field.

6. Select templateImage from the template list that appears. This value represents the template that is created in this development article. If you do not see it, then repeat the steps in this development article. For example, if you made a typing mistake when entering in path information, the template will not show up in the New Page dialog box.

6. Open the new page that you created by double-clicking it in the right pane. The new page opens in a web browser.

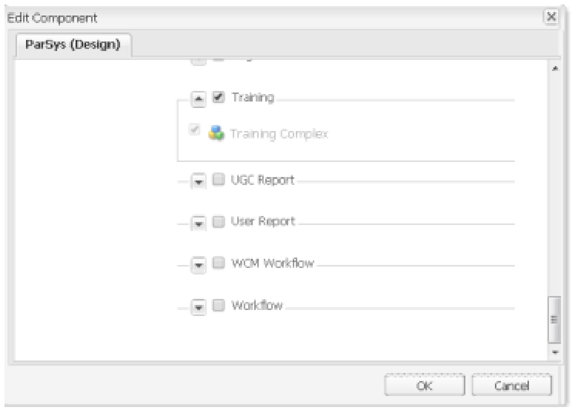
7. From the sidekick, click the Design icon located at the bottom.

8. Click the edit button that appears.



The Edit button that appears on the CQ web page

9. From the Edit Component dialog that appears, select the component under training as shown in this illustration.



The CQ component that is created in this development article

10. Click OK.

11. Now the new component appears on the sidekick. Drag the component to the web page. Now you will see the component as shown at the beginning at this article. If if does not appear, ensure that you created all the nodes and properties that are created in this article.
