



IBM Case Manager 5.2.1 Enablement

Lab

Custom widget
development using plugins

Contents

IBM Case Manager 5.2.1 Enablement.....	1
Before You Begin.....	3
Introduction.....	3
Documentation Conventions.....	3
Custom widget development using plugins.....	5
Exercise 1 – Setting up your development environment with Eclipse.....	6
Exercise 2 – Developing IBM Content Navigator custom plugin.....	9
Exercise 3 – Developing the catalog and definition JSON files.....	12
Exercise 4 – Adding Dojo AMD loading.....	22
Exercise 5 – Exposing widget attributes in the content pane.....	25
Exercise 6 – Handling and exposing an event.....	27
Exercise 7 – Incorporating a menu action into the custom widget.....	29
Exercise 8 – Incorporating a toolbar action into the custom widget.....	31
Exercise 9 – Creating and deploying your custom widget package.....	33
Additional eLearning Resources.....	38
Troubleshooting.....	40




Before You Begin

Introduction


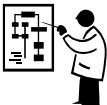
This hands on lab, Custom widget development using plugins , will cover the steps necessary to create a custom widget that works with IBM Case Manager V5.2 with Dojo and a IBM Content Navigator plugin.

Documentation Conventions

The following documentation conventions are used to assist in performing each task:

Convention	Explanation
Bold	Words that appear in boldface represent menu options, buttons, icons, or any object you click to cause the software to perform a task. This typeface also represents anything that you must type or enter.
<i>italics</i>	In addition to book titles, italics are used to emphasize certain words, especially new terms when they are first introduced.
Note	This signifies information that emphasizes or supplements important points of the main text.
 Important	This signifies information essential to the completion of a task. You can disregard information in a note and still complete a task, but you should not disregard an important note.
 Caution	This alerts you to follow a recommended procedure carefully. Failure to do so may result in installation or configuration problems or other preventable conditions.
 Tip	This suggests alternative methods that may not be obvious and helps you understand the benefits and capabilities of a feature or function. A tip is not

	essential to the basic understanding of the text.
□	This symbol indicates the end of a note, caution, or tip.

Convention	Explanation
<div>  <div> Presentation </div> </div>	<p>The presentation provides conceptual information and background knowledge. Presentations take many forms: formal presentations, instructor lecture, or discussion.</p>
<div>  <div> Exercise </div> </div>	<p>These are hands-on exercises used to reinforce the concepts and information covered in a presentation.</p>

Unit AD6

Custom widget development using plugins

In the labs for this unit you will practice the steps required to create a custom widget for IBM Case Manager V5.2.

What you'll learn in this lab:

1. How to set your Eclipse development environment
2. How to create the catalog and widget definition files
3. How to set up Dojo AMD loading
4. How to expose widget attributes in the content pane
5. How to handle and expose an event
6. How to incorporate a menu action into the custom widget
7. How to incorporate a toolbar button into the custom widget
8. How to package a custom widget package

Exercise 1 – Setting up your development environment with Eclipse

Use Eclipse to set up the IBM Case Manager V5.2.1 custom widget development environment. Specifically, this example uses the Eclipse IDE for Java EE Developers package with the JavaScript editor plug-in. The Navigator API plug-in jar can be obtained by finding your navigatorapi.jar file on your environment. It is detailed in the IBM Content Navigator Redbook, which can be found at the following link: <http://www.redbooks.ibm.com/redbooks/pdfs/sg248055.pdf> and the IBM Case Manager 5.2 Redbook, which can be found at the following link: <http://www.redbooks.ibm.com/abstracts/sg247929.html?Open>


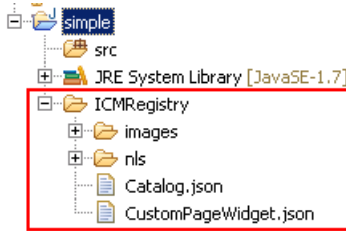
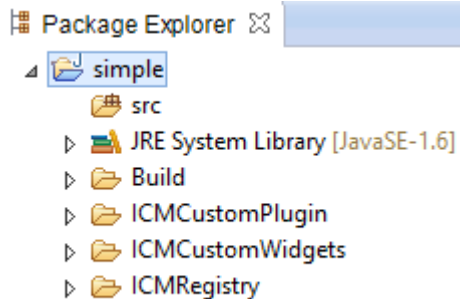
More details about setting up your development environment and custom widget setup can be found in the following article:

https://www.ibm.com/developerworks/mydeveloperworks/blogs/e8206aad-10e2-4c49-b00c-fee572815374/resource/ACM_LP/ICM52CustomWidgets.pdf

It is important to note that the **Simple_Widget_Workspace** sample provides 4 folders: Build, ICMRegistry, ICMCustomPlugin, and ICMCustomWidgets.

ICMRegistry is the folder that provides a simple widget definition JSON file and a catalog JSON file. The ICMCustomWidget folder contains the template for the IBM Case Manager simple custom widget and is the location where you will construct the simple custom widget. The ICMCustomPlugin folder is where you construct the jar file needed to create a plugin for IBM Content Navigator.

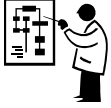
Step	Action
1	Double click on the Eclipse shortcut on the desktop and create your workspace.
2	<p>Create a Java Project:</p> <p>a) In your development, environment, click File > New > Project > Java Project. Click Next.</p> <p>b) In this example, give the project name as “simple” (for “simple custom widget”). Leave the rest of the options as default.</p> <p>c) Click Finish to create the project.</p>

Step	Action
3	<p>Double-click on the Simple_Widget_Workspace shortcut on the desktop, and find the folder ICMRegistry. Right-click the ICMRegistry folder and select Copy. The ICMRegistry folder contains the template for the widget definition JSON file and the catalog JSON file.</p> <p> If your copy of Eclipse does not have Java Project as one of the choices, or it doesn't include the JavaScript editor, use Help * install New Software... to add both Java and/or JavaScript support packages as needed.</p>
4	<p>Navigate back to Eclipse and paste the ICMRegistry folder into your project. Validate that your simple Java project resembles the screenshot below.</p> 
5	<p>Navigate back to the Simple_Widget_Workspace folder. Locate the ICMCustomPlugin folder. Copy and paste the ICMCustomPlugin folder file into your Eclipse project simple.</p>
6	<p>Navigate back to the Simple_Widget_Workspace folder. Copy and paste ICMCustomWidgets folder into your Eclipse project simple.</p>
7	<p>Navigate back to the Simple_Widget_Workspace folder. Copy and paste the Build folder into your Eclipse project simple.</p> <p>The directory should look like the screenshot here:</p> 
8	<p>Enable line numbers under Window > Preferences > General > Editors > Text Editor > Show Line Numbers</p>

Summary

In this section you:

- Prepared a development environment for the simple custom widget in IBM Case Manager V5.2.1
-



Exercise 2 – Developing IBM Content Navigator custom plugin

The next step is to develop the IBM Content Navigator custom plugin.

If you get stuck at any time, you may reference the finished code at
C:\Simple_Widget_Workspace\SimpleCustomWidget_AnswerKey.

Step	Action
1	<p>Before we start to create a new Content Navigator project, ensure that you have access to the <code>navigatorAPI.jar</code>. The <code>navigatorAPI.jar</code> is located in the <code>lib</code> folder of the Content Navigator installation directory, for example: C:\IBM\ECMClient\lib</p> <p>Navigate to <code>ICMCustomPlugin</code> → <code>lib</code> and validate the <code>nexus.jar</code> file is there. Note that we have renamed it from <code>navigatorAPI.jar</code> to <code>nexus.jar</code>. This is the same jar file that you find in the Content Navigator installation directory.</p>
2	<p>Within the simple project in Eclipse, open the ICMCustomPlugin folder and expand the src folder in the project and open ICMCustomPlugin.java in src → com → ibm → icm → extension → custom</p>
3	<p>Under the method public string getId, note that it returns “ICMCustomPlugin”. This is near line 16. This provides an identifier for the plug-in.</p> <pre> @Override public String getId() { // TODO Auto-generated method stub return "ICMCustomPlugin"; } </pre>
4	<p>Under the method public String getName, we are providing a descriptive name for this plug-in. The name identifies the plug-in in the IBM Content Navigator administration tool. This is near line 22.</p> <pre> @Override public String getName(Locale locale) { // TODO Auto-generated method stub String name = NLSResources.getMessage(locale, "icm.plugin.name"); return name; } </pre>

Step	Action
5	<p>Under the method public PluginAction[] getActions, this provides a list of actions that this plug-in adds to the main toolbar of the web client. This is near line 35.</p> <p>Note that these actions are defined under ICMCustomPlugin → src → com → ibm → icm → extension → custom → actions</p> <pre> public PluginAction[] getActions() { return new PluginAction[] { new CustomAddCaseAction(), new CustomAddToAttachmentAction(), new CustomAddTaskAction(), new CustomAddCasePerRoleAction() }; } </pre>
6	<p>Under the method public String getScript, this returns the name of a JavaScript file provided by this plug-in. This file is downloaded to the web browser during the initialization of the web client, before login. A script can be used to perform customization that cannot be performed by other extension mechanisms, such as features or layouts. However, it is preferable to use these other extension mechanisms to provide more flexibility to the administrator when configuring desktops.</p> <p>This is near line 45.</p> <pre> public String getScript() { return "ICMCustomPlugin.js"; } </pre>
7	<p>Next, open ICMCustomPlugin.js under ICMCustomPlugin → src → com → ibm → icm → extension → custom → WebContent.</p> <p>This file includes custom code for any actions and functions to add any global JavaScript methods that the plug-in requires.</p> <p>Note that in ICMCustomPlugin.js, we are loading the CSS and performing some debugging functions.</p>
8	<p>In the Eclipse navigation pane on the left, double click the widget definition JSON file, CustomPageWidget.json to open it. You may notice that the beginning looks similar to the end of the Catalog.json file.</p>
9	<p>The steps in this exercise did not have any action items. Continue to the next section to go over how to create a custom widget in IBM Case Manager v5.2.1.</p>

Summary

In this section you:

- Reviewed the structure of an IBM Content Navigator custom plug-in
-



Exercise 3 – Developing the catalog and definition JSON files

The next step is to develop the catalog and widget definition JSON files. These files are needed to define the custom widget properties and define any event broadcasting or handling within your custom widget.

You may use the JSON Viewer application that is available in the JSONView folder on the Desktop of the image.

If you get stuck at any time, you may reference the finished code at
C:\Users\Simple_Widget_Workspace\SimpleCustomWidget_AnswerKey.

Step	Action
1	Within the simple project in Eclipse, open the ICMRegistry folder and double-click the file Catalog.json to open it.
2	Note that the Name , Description , and other fields are not filled in. Starting with the Name and Description fields near the top of Catalog.json, enter a value for the name of this custom widget package into the name field. Likewise, enter a description into the description fields.

See the following example:

```
{
  "Name": "IBM Case Manager simple custom widget package",
  "Description": "IBM Case Manager simple custom widget package",
```

3 Add in the value for Categories which is the category values for this specific custom widget package that you will see in the palette bar on the left in Page Designer.

Enter values for the **id** and **title** for the Categories section.

See the following example:

```
"Categories": [
  {
    "id": "SimpleCustomWidget",
    "title": "Simple Custom Widget"
  }
],
```

Step	Action
------	--------

- | | |
|---|---|
| 4 | Add the values for the Widgets section in the Catalog.json file. Enter values for the id and title fields. See the following example: |
|---|---|

```
"Widgets": [  
  {  
    "id": "CustomPageWidget",  
    "title": "Simple Custom Page Widget",  
  }  
]
```

- | | |
|---|---|
| 5 | Add the value for the category . Pay special attention to name your category the same name that you provided for the category section in the previous step. See the following example: |
|---|---|

```
"Widgets": [  
  {  
    "id": "CustomPageWidget",  
    "title": "Simple Custom Page Widget",  
    "category": "SimpleCustomWidget",  
  }  
]
```

- | | |
|---|---|
| 6 | Add in the description value for the widget. |
|---|---|

See below for an example:

```
"Widgets": [  
  {  
    "id": "CustomPageWidget",  
    "title": "Simple Custom Page Widget",  
    "category": "SimpleCustomWidget",  
    "description": "This is a simple custom widget",  
  }  
]
```

Step Action

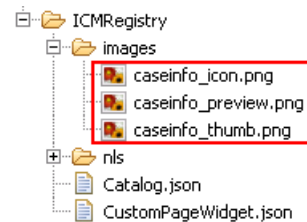
- 7 Enter the value for the **widget definition file**. Since it is already created in the ICMRegistry folder, enter the file name of the widget definition file, **CustomPageWidget.json**.

See the following example:

```
"Widgets": [
  {
    "id": "CustomPageWidget",
    "title": "Simple Custom Page Widget",
    "category": "SimpleCustomWidget",
    "description": "This is a simple custom widget",
    "definition": "CustomPageWidget.json",
  }
]
```

- 8 Enter the value for the **preview**, **icon**, and **previewThumbnail images** for the custom widget. The image files are already imported into the ICMRegistry folder.

In your Eclipse navigation pane on the left, expand the images folder under ICMRegistry and enter the path names for the corresponding **preview**, **icon**, and **previewThumbnail images**.



See below for an example:

```
"preview": "images/caseinfo_preview.png",
"icon": "images/caseinfo_icon.png",
"runtimeClassName": "icm.custom.pgwidget.customWidget.CustomPageWidget",
"help": "acmwrh126.htm",
"previewThumbnail": "images/caseinfo_thumb.png",
"properties": {
```

Step Action

- 9** For the Catalog.json file, define the **runtimeClassName**. Because the ICMCustomWidget folder with the widget source code is already imported, find the main runtimeClassName.

In your Eclipse navigation pane on the left, expand **ICMCustomWidgets > WebContent > icm > custom > pgwidget > customWidget > CustomPageWidget.js**

This is the path we want to define in our **runtimeClassName** field in our Catalog.json file.

See below for an example:

```
"Widgets": [
  {
    "id": "CustomPageWidget",
    "title": "Simple Custom Page Widget",
    "category": "SimpleCustomWidget",
    "description": "This widget show payload from Search widget",
    "definition": "CustomPageWidget.json",
    "preview": "images/caseinfo_preview.png",
    "icon": "images/caseinfo_icon.png",
    "runtimeClassName": "icm.custom.pgwidget.customWidget.CustomPageWidget",
    "help": "acmwrh126.htm",
    "previewThumbnail": "images/caseinfo_thumb.png"
  }
]
```

- 10** In the Eclipse navigation pane on the left, double click the widget definition JSON file, **CustomPageWidget.json** to open it. You may notice that the beginning looks similar to the end of the Catalog.json file.

- 11** Using the same values you entered for the Catalog.json file, fill in the fields for the **id**, **title**, **category**, **description**, and so on for the CustomPageWidget.json file. See the following example:

```
"id": "CustomPageWidget",
"title": "Simple Custom Page Widget",
"category": "SimpleCustomWidget",
"description": "This widget show widget properties and event name.",
"definition": "CustomPageWidget.json",
"preview": "images/caseinfo_preview.png",
"icon": "images/caseinfo_icon.png",
"runtimeClassName": "icm.custom.pgwidget.customWidget.CustomPageWidget",
"help": "acmwrh126.htm",
"previewThumbnail": "images/caseinfo_thumb.png",
```

Step	Action
------	--------

- | | |
|----|--|
| 12 | For the properties section in your CustomPageWidget.json file, you can enter one or more properties to be displayed in the settings window of your custom widget in Page Designer. |
|----|--|

Set a **PreferredHeight** property for sizing of the custom widget. Hide this property and set it to be invisible. This step describes one way to set a property to make it invisible to the user.

Set the **id**, **disabled**, **required**, **visibility**, and **title** fields.

See below for an example:

```
"properties": [  
  {  
    "propertyType": "property",  
    "type": "string",  
    "id": "PreferredHeight",  
    "defaultValue": "100%",  
    "disabled": true,  
    "required": false,  
    "visibility": false,  
    "title": "Preferred Height"  
  },  
]
```

Step Action

- 13** Define the next property as a custom string in the widget attributes that the user can set.

Set the values for **id**, **remapNeeded**, **defaultValue**, **required**, **visibility**, and **title**. See the following example:

```
{
  "propertyType": "property",
  "type": "string",
  "id": "customProperty1",
  "remapNeeded": true,
  "defaultValue": "http://",
  "required": false,
  "visibility": true,
  "style": "width:95%;",
  "title": "String property"
},
```



remapNeeded is for situations where the custom property has some value which ties it closely to the TOS where the solution is deployed and when you export/import the solution on a new TOS, you may need to remap it to a new value. This is for such properties that may need to be modified during export.

Step Action

- 14** Repeat the previous step for custom integer properties. The the custom boolean property is already completed.

Set the values for **id**, **remapNeeded**, **defaultValue**, **required**, **visibility**, and **title** for both properties.

See the following example:

```
{
  "propertyType": "property",
  "type": "integer",
  "id": "customProperty2",
  "remapNeeded": true,
  "defaultValue": 20,
  "required": true,
  "visibility": true,
  "title": "Integer Property",
  "description": "Integer property"
},
{
  "propertyType": "property",
  "type": "boolean",
  "id": "customProperty3",
  "remapNeeded": true,
  "defaultValue": false,
  "required": false,
  "visibility": true,
  "title": "Boolean property"
},
}
```

Step Action

- 15** For the next section, we will define the menu action for the custom widget. This allows the user to right click on the custom widget to get a list of options to invoke.

Fill in the following fields for this custom property: **id**, **title**, and **actionList**. Make sure to fill these fields as the screenshot below.

```

"propertyType": "group",
"type": "tab",
" id": "Menu",
" title": "menu",
"propertyMembers": [
  {
    "propertyType": "property",
    "type": "contextualMenu",
    "id": "customContextualMenu",
    "context": ["CustomContext"],
    "defaultValue": {
      "actionList": [
        {
          "actionDefinitionId": "icm.action.utility.OpenWebPage",
          "propertiesValue": {
            "label": "IBMWebPage",
            "targetURL": "http://www.ibm.com"
          }
        }
      ]
    },
    "required": false,
    "visibility": true,
    " title": "custom Menu"
  }
]

```

Step Action

- 16** For the next section, we will define the toolbar action for the custom widget. This adds a toolbar with a button that allows the user to click to invoke an action.

Fill in the following fields for this custom property: **id**, **title**, **actionList**. Make sure to fill these fields as the screenshot below:

```

"propertyType": "group",
"type": "tab",
'id': "Toolbars",
'title': "toolbar",
"propertyMembers": [
  {
    "propertyType": "property",
    "type": "toolbar",
    "id": "customToolbar",
    "context": ["CustomContext"],
    "defaultValue": {
      "actionList": [
        {
          "actionDefinitionId": "icm.action.utility.OpenWebPage",
          "propertiesValue": {
            "label": "IBMWebPage",
            "targetURL": "http://www.ibm.com"
          }
        }
      ]
    },
    "required": false,
    "visibility": true,
    'title': "custom toolbar"
  }
]

```

- 17** Define the events section of the custom widget. This section determines whether to handle or broadcast an event and sets the function name to handle the event if applicable. In this example, set the id to **icm.customEvent**. This event id allows this custom widget to handle any event that is wired to the custom widget.

See the screenshot below:

```

"events": [
  {
    "id": "icm.CustomEvent",

```

Step Action

- 18** Set the **title** and **description** for the event. These are not specific like `icm.CustomEvent`, so provide a title and description to those fields

See below for an example:

```
"events": [
  {
    "id": "icm.CustomEvent",
    "title": "Custom Event 1",
    "functionName": "",
    "direction": "",
    "description": "Custom Event 1"
  }
]
```

- 19** Set the **functionName** and **direction** fields for the event. Because the widget is handling an event from another widget, set direction to **subscribed**. You can also set it to broadcast if you want to send an event to another widget. Set the functionName to **handleICM_CustomEvent** as that is the function name in the .js file.

The following example shows the updated events section of your CustomPageWidget.json file:

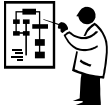
```
"events": [
  {
    "id": "icm.CustomEvent",
    "title": "Custom Event 1",
    "functionName": "handleICM_CustomEvent",
    "direction": "subscribed",
    "description": "Custom Event 1"
  }
]
```

- 20** Make sure to save all of your changes to the two JSON files in Eclipse.
-

Summary

In this section you:

- Created a catalog JSON file
 - Created a definition JSON file
-

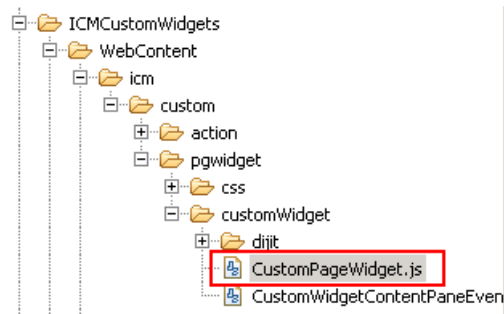


Exercise 4 – Adding Dojo AMD loading

In this exercise, you will add Dojo AMD loading into CustomPageWidget.js. Dojo AMD loading allows the application developer to load different classes to use into the custom widget. This is useful when using the IBM Case Manager JavaScript API reference or the IBM Content Navigator API references as the application developer will need to define these classes.

If you get stuck at any time, you may reference the finished code at
C:\Users\Simple_Widget_Workspace\SimpleCustomWidget_AnswerKey.

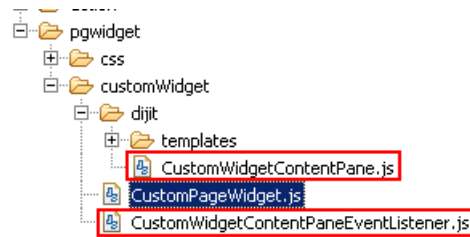
Step	Action
1	In your Eclipse project navigation pane on the left, navigate to CustomPageWidget.js under ICMCustomWidgets > WebContent > icm > custom > pgwidget > customWidget
2	Double click on CustomPageWidget.js to open it.



Step Action

- 3** We will begin by defining some of our other JS files in the custom widget package using Dojo AMD. Notice that the Dojo libraries have already been loaded like dojo/dom-style. We also have other IBM Case Manager JS files referenced like icm/base/BasePageWidget.

In your Eclipse project navigation pane on the left, note the path for **CustomWidgetContentPaneEventListener.js** and **CustomWidgetContentPane.js**.



- 4** We want to define these files in our CustomPageWidget.js file so that we can call the functions if need be in those classes.

Fill in the last two empty quotes in CustomPageWidget.js file in the **define** header.

See below for an example:

```
define([ "dojo/_base/declare",
        "dojo/_base/lang",
        "dojo/dom-geometry",
        "dojo/dom-style",
        "icm/base/BasePageWidget",
        "icm/custom/pgwidget/customWidget/CustomWidgetContentPaneEventListener",
        "icm/custom/pgwidget/customWidget/dijit/CustomWidgetContentPane",
```

- 5** We want to also add our two JS file definitions into our **function** method at the top.

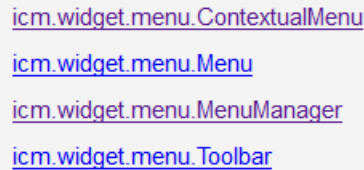
See below for an example:

```
"icm/base/BaseActionContext"], function(declare, lang, domGeom, domStyle, I
    eventListener, contentPaneWidget, MenuManager, toolBar, ContextualMenu,
```

Step Action

- 6** Next, we want to define some IBM Case Manager JavaScript API model classes. Specifically, we want to define “icm/widget/menu/ContextualMenu” and “icm/widget/menu/Toolbar” so we can use the Toolbar and Menu functionalities in the simple custom widget.

We can see the classes in the IBM Case Manager JavaScript API Reference on the IBM Case Manager V5.2 Information Center:



icm.widget.menu.ContextualMenu
icm.widget.menu.Menu
icm.widget.menu.MenuManager
icm.widget.menu.Toolbar

Repeat the previous steps and add the path for these **two model classes** into the **define** section and into the **function** section.

See below for an example:

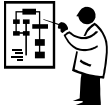
```

    "icm/widget/menu/MenuManager",
    "icm/widget/menu/Toolbar",
    "icm/widget/menu/ContextualMenu",
    "icm/base/BaseActionContext"], function(declare, lang, domGeom, domStyle,
    eventListener, contentPaneWidget, MenuManager, toolbar, ContextualMenu
  
```

Summary

In this section you:

- create Dojo AMD loading for the simple custom widget
-



Exercise 5 – Exposing widget attributes in the content pane

In the simple custom widget, we want to expose the custom properties that we defined in the widget definition JSON file on the widget's content pane itself. This involves editing the HTML file for the custom widget content pane and setting up a way in the JavaScript file to show the properties onto the content pane when the user enters a value into the string, boolean, and integer fields.

Within Case Manager Client, the user would click the Edit Settings icon and then enter a value for the custom properties. The widget would then show the values.

Here is an example:

A simple custom page widget

Widget Attributes:

String property: http://
Integer property: 20
Boolean property: false

Receive event:

RightClickMeForShowingContextualMenu

Below is the buttons in toolbar

IBMWebPage

If you get stuck at any time, you may reference the finished code at
C:\Users\Simple_Widget_Workspace\SimpleCustomWidget_AnswerKey.

Step	Action
1	Open the CustomWidgetContentPane.html file located in ICMCustomWidgets → WebContent → icm → custom → pgwidget → customWidget → dijits → templates in Eclipse.

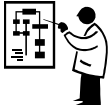
Step	Action
2	<p>We want to define a <code>dojoAttachPoint</code> so we can display the widget attributes on the HTML content pane page in <code>CustomWidgetContentPane.html</code>.</p> <p>On line 6 of <code>CustomWidgetContentPane.html</code>, add a <code>dojoAttachPoint</code> and label it "displayAttributes".</p> <p>See below for an example:</p> <pre><div dojoAttachPoint="displayAttributes" style="width: 100%; "></div></pre>
3	<p>Open <code>CustomWidgetContentPaneEventListener.js</code> under the <code>ICMCustomWidgets → WebContent → icm → custom → pgwidget → customWidget</code>.</p>
4	<p>In <code>CustomWidgetContentPaneEventListener.js</code>, we want to take that <code>dojoAttachPoint</code> called displayAttributes and display the string, integer, and boolean widget attribute values.</p> <p>In line 25 of <code>CustomWidgetContentPaneEventListener.js</code>, define a variable called <code>props</code> and display each property using <code>this.contentPane.widgetProperties['customproperty1']</code> and so on.</p> <p>See below for an example:</p> <pre>var props = 'String property: ' + this.contentPane.widgetProperties['customProperty1'] + '
 Integer property: ' + this.contentPane.widgetProperties['customProperty2'] + '
 Boolean property: ' + this.contentPane.widgetProperties['customProperty3'];</pre>
5	<p>Next, we want to display the variable <code>props</code> and add it to the <code>dojoAttachPoint</code> that we defined earlier and add it in line 26.</p> <p>See below for an example:</p> <pre>this.contentPane.displayAttributes.innerHTML = props;</pre>
6	<p>Make sure to save your changes to both files before moving forward.</p>

Summary

In this section you:

- exposed widget attributes using the content pane
-





Exercise 6 – Handling and exposing an event

This exercise allows you to handle an event that is wired to the simple custom widget and expose the event name to the content pane of the custom widget.

In Case Manager Client, any event that is wired to the simple custom widget will have its event name exposed in the content pane. In the screenshot below, we wired the Search widget's icm.SearchCases event to the simple custom widget.

A simple custom page widget

Widget Attributes:

String property: http://
Integer property: 20
Boolean property: false

Receive event:

icm.SearchCases
RightClickMeForShowingContextualMenu

Below is the buttons in toolbar

IBMWebPage

If you get stuck at any time, you may reference the finished code at
C:\Users\Simple_Widget_Workspace\SimpleCustomWidget_AnswerKey.

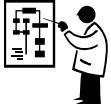
Step	Action
1	Open the CustomWidgetContentPane.html file located in ICMCustomWidgets → WebContent → icm → custom → pgwidget → customWidget → dijit → templates in Eclipse.
2	We want to add a dojoAttachPoint named displayEvent to expose the event's name in the content pane. Add this on line 8 of CustomWidgetContentPane.html . See below for an example: <div dojoAttachPoint="displayEvent" style="width: 100%; "></div>
3	Open the CustomPageWidget.js file located in ICMCustomWidgets → WebContent → icm → custom → pgwidget → customWidget in Eclipse.
4	Starting on line 74, notice that we have the handler function that was defined earlier in

Step	Action
	<p>the widget definition JSON file. These names should have matched.</p> <p>We want to call the function displayPayload within CustomWidgetContentPaneEventListener.js that was defined earlier using Dojo AMD loading. Add this line on line 79.</p> <p>See below for an example:</p> <pre>this.contentPaneEventListener.displayPayload(payload);</pre>
5	<p>Navigate to CustomWidgetContentPaneEventListener.js and go to the displayPayload function that is defined on line 19.</p> <p>On line 20, use the dojoAttachPoint, displayEvent, that was defined earlier and assign it to the eventName field of the payload using payload.eventName.</p> <p>See below for an example:</p> <pre>this.contentPane.displayEvent.innerHTML = payload.eventName;</pre>
6	<p>Make sure you save all files that were edited before proceeding forward.</p>

Summary

In this section you:

- handled and exposed an event for the simple custom widget
-



Exercise 7 – Incorporating a menu action into the custom widget

In this exercise, you will incorporate a menu action into the custom widget that can be invoked when the user right clicks on an area on the simple custom widget. This will allow the user to open a web page to ibm.com in this lab. However, you can add additional menu action items when you click the Edit Settings icon in Page Designer for the simple custom widget.

This is what occurs when you right click on the specified area of the custom widget in Case Manager Client:

A simple custom page widget

Widget Attributes:

String property: `http://`
 Integer property: 20
 Boolean property: false

Receive event:

`icm.SearchCases`

`RightClickMeForShowingContextualMenu`

IBMWebPage

Below is the buttons in toolbar

IBMWebPage

If you get stuck at any time, you may reference the finished code at
 C:\Users\Simple_Widget_Workspace\SimpleCustomWidget_AnswerKey.

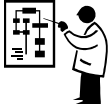
Step	Action
1	<p>Open CustomWidgetContentPane.html in Eclipse. We want to create a data-dojo-attach-point named contextualMenuStore for the menu action in line 9.</p> <p>See below for an example:</p> <pre><div data-dojo-attach-point="contextualMenuStore"></div></pre>

Step	Action
2	<p>Next, navigate to CustomPageWidget.js. We first want to instantiate the contextualMenu with the dojoAttachPoint that we defined in CustomWidgetContentPane.html and add that to line 48 in CustomPageWidget.js.</p> <p>See below for an example:</p> <pre>dojoAttachPoint: "customContextualMenu"</pre>
3	<p>Append the menu in the custom widget in order to invoke the menu action configuration from the page widget in line 52.</p> <p>See below for an example:</p> <pre>this.contextualMenuStore.appendChild(this.menu.domNode);</pre>
4	<p>Next, set the target reference of the contextualMenu so that it can bound to the target point in line 54.</p> <p>See below for an example:</p> <pre>MenuManager.setTargetReference(this.menu, "contextualMenuTargetRefPoint");</pre>
5	<p>Set your context defined for CustomContext so that action configured in contextualMenu can get it for running as required in line 56.</p> <p>See below for an example:</p> <pre>MenuManager.setSelectedItems(this.id, "contextualMenuTargetRefPoint", [{customProperty: true}], "CustomContext");</pre>
6	<p>Activate the menu in line 60.</p> <p>See below for an example:</p> <pre>this.menu.startup();</pre>

Summary

In this section you:

- incorporated a menu action into the custom widget
-



Exercise 8 – Incorporating a toolbar action into the custom widget

In this exercise, you will incorporate a toolbar action into the custom widget that can be displayed as a button within a toolbar area on simple custom widget. This will allow the user to open a web page to ibm.com in this lab, which is the same action as the menu action earlier. However, you can add additional toolbar action items when you click the Edit Settings icon in Page Designer for the simple custom widget.

This is what occurs when incorporate a toolbar action into the custom widget:

A simple custom page widget

Widget Attributes:

String property: <http://>
 Integer property: 20
 Boolean property: false

Receive event:

icm.SearchCases
 RightClickMeForShowingContextualMenu

Below is the buttons in toolbar



If you get stuck at any time, you may reference the finished code at
 C:\Users\Simple_Widget_Workspace\SimpleCustomWidget_AnswerKey.

Step	Action
1	<p>Open CustomWidgetContentPane.html in Eclipse. We want to create a data-dojo-attach-point named wrapTopToolbar for the toolbar action in line 12. We also want to set the data-dojo-type to be dijit.layout.ContentPane, the data-dojo-props to be region:'bottom', and the style to be 'border:none;' in line 12.</p> <p>See below for an example:</p> <pre><div data-dojo-type="dijit.layout.ContentPane" data-dojo-attach-point="wrapTopToolbar" data-dojo-props="region:'bottom', style:'border:none;'"></div></pre>

Step	Action
2	<p>Next, navigate to CustomPageWidget.js. We first want to instantiate the toolbar with the <code>dojoAttachPoint</code> that we defined in <code>CustomWidgetContentPane.html</code> and add that to line 36 in CustomPageWidget.js.</p> <p>See below for an example:</p> <pre>dojoAttachPoint: "customToolbar"</pre>
3	<p>Set the toolbar as a content of the page widget in order to get the action configuration from the page widget in line 39.</p> <p>See below for an example:</p> <pre>this.wrapTopToolbar.set("content", this.topToolbar.domNode);</pre>
6	<p>Activate the menu in line 42.</p> <p>See below for an example:</p> <pre>this.topToolbar.startup();</pre>

Summary

In this section you:

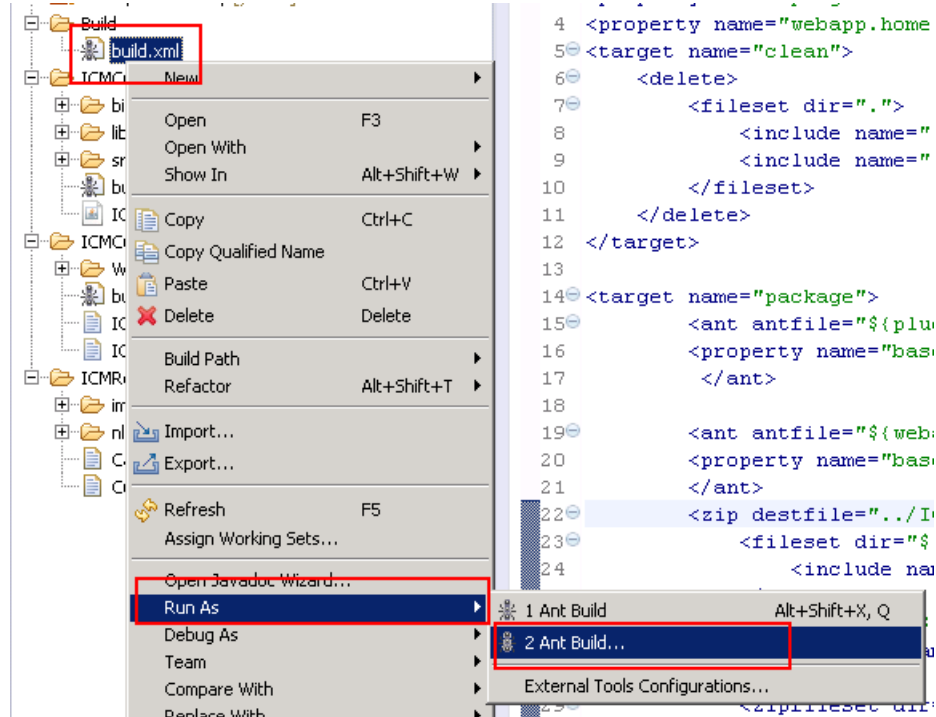
- incorporated a toolbar action into the custom widget
-



Exercise 9 – Creating and deploying your custom widget package

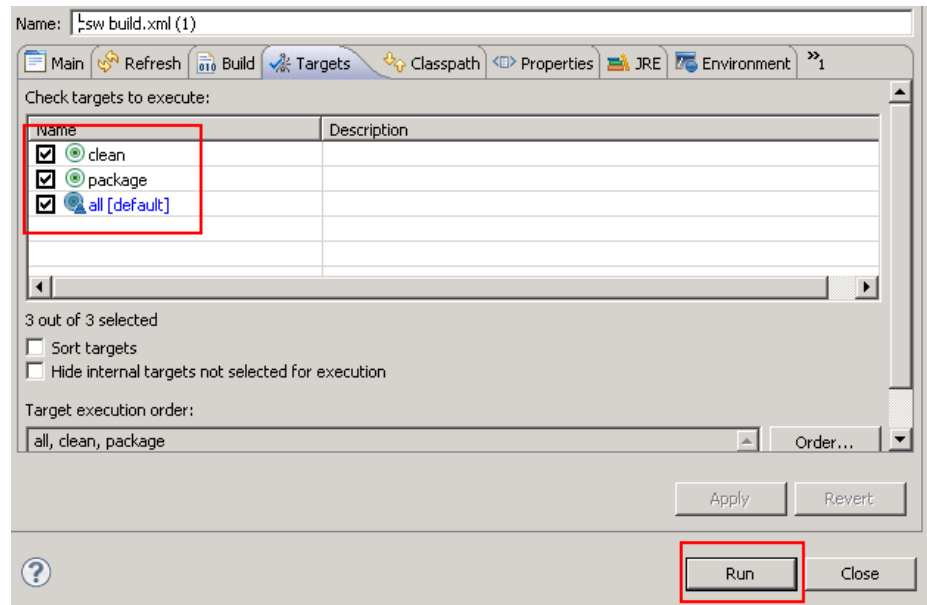
In this exercise, you will create a zip file containing your widget code. This zip file is needed for the next lab where you will deploy this zip file using IBM Case Manager Configuration Tool. This zip file contains an EAR file, which is optional. If you did not have an EAR file, you would use IBM Case Manager Administration Client with just a JAR file.

Step	Action
1	Within the Eclipse environment, right click on build.xml and select “Run as...”, then select “2. Ant Build...”

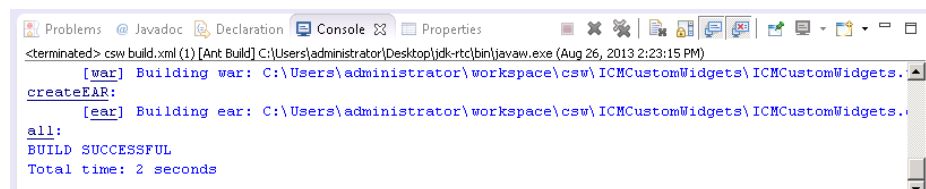


Step	Action
------	--------

- | | |
|---|---|
| 2 | Make sure to select “all [default]”, “clean”, and “package”, then select run to create your jar file with your source code. Make sure the execution order matches the screenshot below. |
|---|---|

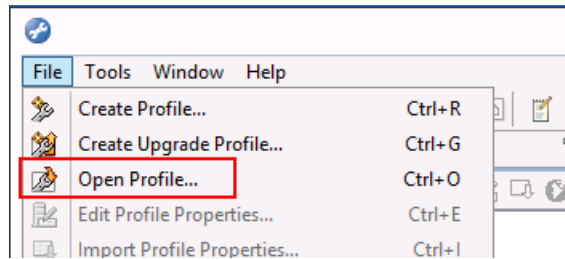


- | | |
|---|--|
| 3 | Validate that you see Build Successful with no warnings in the Console window. |
|---|--|



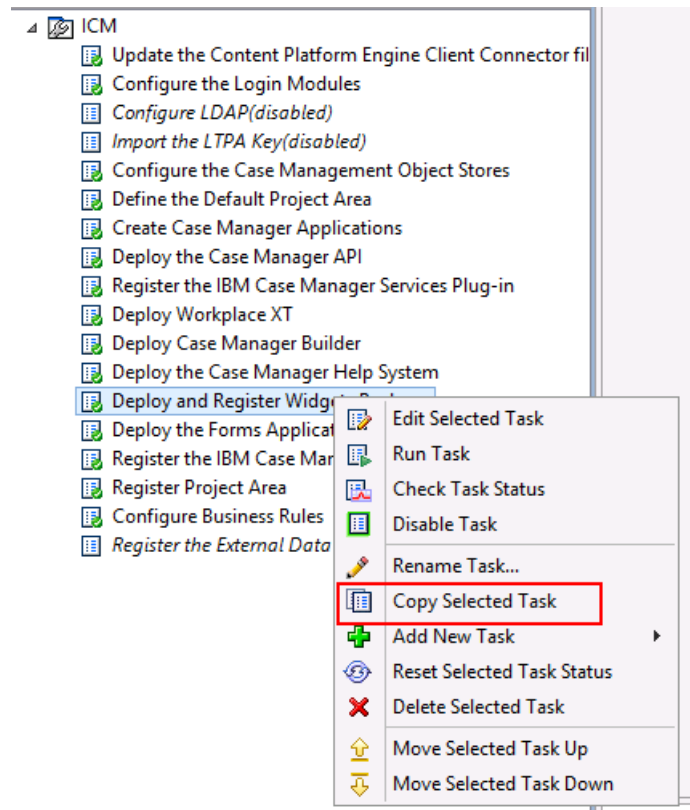
- | | |
|---|---|
| 4 | Navigate to your workspace and validate that you see your newly created ICMCustomWidgets.zip file. |
| 5 | Given that there are no errors, we are going to deploy this custom widget package now. Navigate to the Windows Start button , then All Programs , then IBM Case Manager , then click on Case Manager Configuration tool . |

Step	Action
6	When Case Manager Configuration tool opens, go to File and click Open Profile .



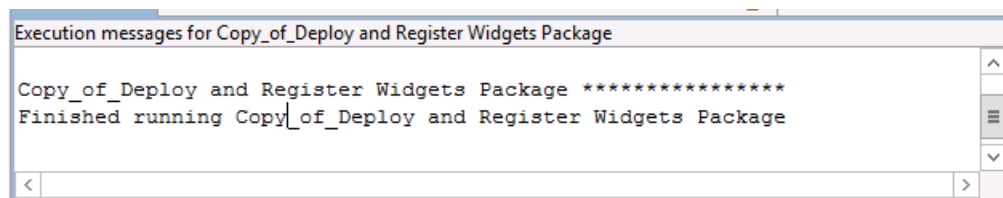
7	Navigate to C:\IBM\CaseManagement\configure\profiles\ICM and select ICM.cfgp . Note, because we have an EAR file in our custom widget, we are using IBM Case Manager Configuration Tool.
---	--

8	Expand the ICM profile and right click on Deploy and Register Widgets Package . Then select Copy Selected Task .
---	---



Step	Action
9	Double click on the newly created task, Copy_of_Deploy and Register Widgets Package to open it.
10	Click the Browse button for Widgets package file path. Select your newly created widget package at C:\Users\p8admin\workspace-eclipse\simple\ICMCustomWidget.zip .

- 11 Click the **Save** button and then click the **Run Task** button. Verify that you do not get any errors and it displays a message that it is finished.



- 12 Next, open Firefox and navigate to Case Manager Builder. Click Edit on the **Credit Card Disputes HOL** solution and navigate to the **Pages** tab.
- We want to add our custom widget to the **Cases** page in this example. Click **Solution Pages** then click on **Cases** to open Page Designer.

Step	Action
13	Drag and drop the Simple Custom Widget onto the page and click the Save and then click the Close button. Commit and Deploy the solution.
14	Click Test on the Credit Card Disputes HOL solution to open Case Manager Client and validate your custom widget.

Summary

In this section you:

- Performed an Ant build within Eclipse, creating your custom widget's war and ear files
 - Deployed the custom widget using Case Manager Configuration Tool
-

Additional eLearning Resources

- IBM Case Manager V5.2 Information Center – Developing case management applications with the JavaScript API topic
<http://pic.dhe.ibm.com/infocenter/casemgmt/v5r2m0/topic/com.ibm.casemgmt.development.doc/acmjs001.htm>
- IBM Case Manager V5.2 Information Center – Creating custom page widgets and actions topic
<http://pic.dhe.ibm.com/infocenter/casemgmt/v5r2m0/topic/com.ibm.casemgmt.development.doc/acmwt000.htm>
- Selected sessions from IBM Case Manager V5.2 Product Implementation and Maintenance Training (PIT / PMT) (220246):

09-ICM 5.2 API Overview PIT PMT (47 minutes)

10-ICM 5.2 Javascript Tool Kit - Model Classes PIT PMT (31 minutes)

13-ICM 5.2 Javascript Tool Kit-UI Classes PIT PMT (58 minutes)

14-ICM 5.2 Event Wiring Examples-Debugging Tips PIT PMT (36 minutes)

15-ICM 5.2 Building Custom Page Widgets PIT PMT (43 minutes)

29-ICM 5.2 Upgrading Widgets From 5.1.1 to 5.2 PIT PMT (44 minutes)

- Selected developerWorks articles:

IBM Case Manager V5.2.0 Custom Page Widgets and Actions Development

https://www.ibm.com/developerworks/community/blogs/e8206aad-10e2-4c49-b00c-fee572815374/entry/ibm_case_manager_v5_2_0_custom_page_widgets_and_actions_development?lang=en

Converting a Custom Search Widget from ICM 5.1.1 to ICM 5.2 v2

https://www.ibm.com/developerworks/community/blogs/e8206aad-10e2-4c49-b00c-fee572815374/entry/converting_a_custom_search_widget_from_icm_5_1_1_to_icm_5_2?lang=en

Mapping Case REST API calls to Model API calls when converting custom widgets to ICM 5.2

https://www.ibm.com/developerworks/community/blogs/e8206aad-10e2-4c49-b00c-fee572815374/entry/mapping_case_rest_api_calls_to_model_api_calls_when_converting_custom_widgets_to_icm_5_2?lang=en

[fee572815374/entry/mapping_case_rest_api_calls_to_model_api_calls_when_converting_custom_widgets_to_icm_5_2?lang=en](https://www.ibm.com/developerworks/community/blogs/fee572815374/entry/mapping_case_rest_api_calls_to_model_api_calls_when_converting_custom_widgets_to_icm_5_2?lang=en)

Using the Model API and REST API call in custom widgets for IBM Case Manager V5.2

https://www.ibm.com/developerworks/community/blogs/e8206aad-10e2-4c49-b00c-fee572815374/entry/using_the_model_api_and_rest_api_call_in_custom_widgets_for_ibm_case_manager_v5_2?lang=en

Troubleshooting

Q: When I run the build ANT script, I get a Java major class version error.

A: Make sure that there are no .class files in your project directory. If there are, delete them and make sure you have a valid Java JRE/JDK.

Q: Is there a place where I can cross-reference my custom widget's code?

A: Yes, please reference the file at

C:\Users\Simple_Widget_Workspace\SimpleCustomWidget_AnswerKey.

Q: I do not see a WAR file/JAR file/my JSON files in my custom widget package.

A: Make sure that you have three ANT build scripts and that you ran the build script in the Build folder.

Q: My widget is not loading!

A: Make sure you followed each step closely and replicated what is shown in the screenshots. Make sure you included all appropriate syntax like commas, quotation marks, and so on. There are case-sensitive items like the Dojo AMD loading and the paths of the classes.