Back-end implementation of a Code Widget

Implement AbstractCodeTemplate

The AbstractCodeTemplate implementation handles code widgets at the back end. This is a Java class which code widget plugins should extend. Yellowfin uses the implementation of this class to determine how a code widget will be available for the Yellowfin instance. There are a number of events and back-end requests that can be made within this widget.

Required methods

The following methods must be overridden in the AbstractCodeTemplate implementation.

Method name	Description	Example
String getTemplateTitle()	This is the name of your widget. It will be the name displayed to users when the widget is available in the canvas widget panels.	*public String getTemplateTitle() {* *return "My Code Widget";* *}*
void setupResources()	This is the method to define any front-end resources your widget may require. This will be called when the class is initialised. Whatever resources are added here will tell Yellowfin which files the front-end is allowed to load. If you attempt to load a file from the front end that is not defined here, it will be rejected. Use the addResource(Resource); function to add your resources. See Resource for information about Resource objects Any resource you add here will be relative to the AbstractCodeTemplate implementation.	Any resources defined by this function are relative to the location of the AbstractCodeTemplate location. So if, for example, our Java package is: *my.code.widget* my_widget.js could be located in the *my.code.widget.resource* package: *my.code.widget* *-MyCodeWidget.java* *my.code.widget.resource* *-my_widget.js* We can define *my_widget.js* us ing the following *void setupResources() {* *addResource(new Resource ("resource/my_widget.js", "text /javascript"));* *}*
String getMainJavascript Path()	This defines the entry point for your code widget. This will be the first file loaded by Yellowfin, and then called. The file you choose must return a constructor. See the JavaScript section for details.	*String getMainJavascriptPath() {* *return "my_widget.js";* *}*

CanvasWidgetPanel getPanel(CanvasWidgetPanelInfo panelInfo)	Used to define the Widget Properties Panel allowing you to define any custom options you wish for your widget. If null is returned from this method, the default widget properties panel will be used.	CanvasWidgetPanel getPanel (CanvasWidgetPanelInfo panelInfo) {
		
		  return new MyCodeWidgetPanel (panelInfo);
		
		}
		
		//To use the default canvas panel
		
		CanvasWidgetPanel getPanel (CanvasWidgetPanelInfo panelInfo) {
		
		  return null;
		
		}
		
		See the Widget Properties Panel section for details

Next step

Once the back-end code is complete, it's time to write the front-end code.

To further assist you during code widget development, visit the code widget reference page for samples, API links and detailed descriptions.

Back-end implementation of a Code Widget#top