Apogee Client Web App

Version 1.0.0-p1

An Apogee workspace can easily be embedded in a standard web page. It is possible to expose selected component views on a web page, with all the underlying code included to support full operation including updates and edits.

Simple Example Page

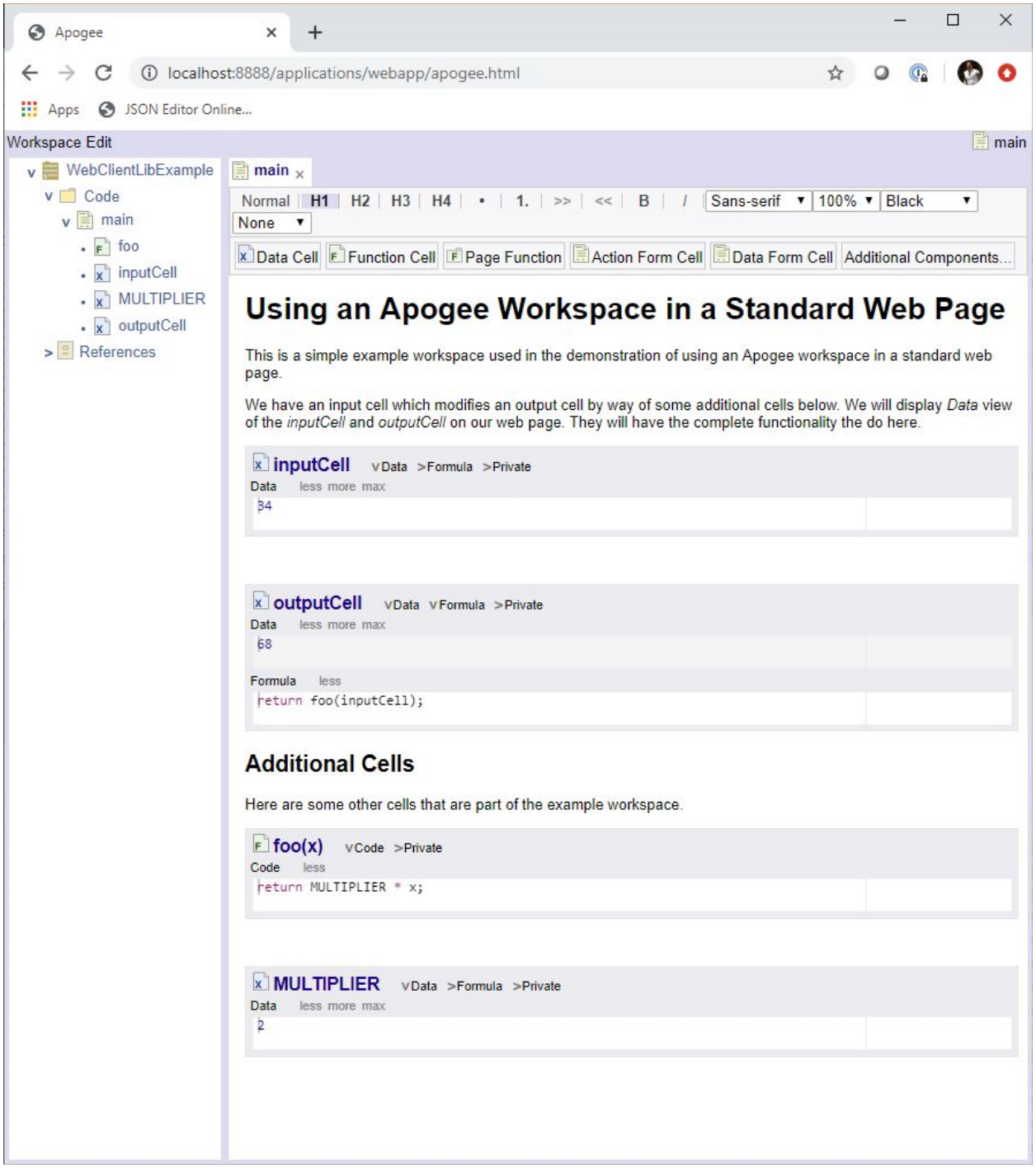
We will create a simple web page that uses the very simple Apogee workspace.

The example shown here can be downloaded from the following locations:

- **Example Workspace** - https://apogeejs.com/lib/v1.0.0-p1/webAppTestWorkspace.json
- **Example Web Page** - https://apogeejs.com/lib/v1.0.0-p1/webTest.html
- **Required ES Module** - https://apogeejs.com/lib/v1.0.0-p1/apogeeWebClientLib.es.js
- **Required CSS Page** - https://apogeejs.com/lib/v1.0.0-p1/cssBundle.css

Workspace

The following is the workspace used in this example.



Web Page

The following web page shows how to embed cell views from an Apogee workspace into a web page. The data views are fully functional. In this case you can edit the value of *inputCell*.

```
<!DOCTYPE html>
```

```html
<html>
    <head>
        <title></title>
        <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <link rel="stylesheet" type="text/css" href="http://apogeejs.com/lib/v1.0.0-p1/cssBundle.css">
        <style>
            body {
                padding: 10px;
            }
            .cellViewContainer {
                width: 600px;
            }
        </style>
        <script type="module">
            import {ApogeeWebView} from "http://apogeejs.com/lib/v1.0.0-p1/apogeeWebClientLib.es.js";

            //configure app and load the workspace
            let workspaceUrl = "webAppTestWorkspace.json";
            let appView;

            window.init = function() {
                appView = new ApogeeWebView();

                //this adds a display view to the page to the given element
                //should be done before the workspace is loaded.
                appView.addDisplay("main.inputCell","inputDisplayElement",true,"Data");
                appView.addDisplay("main.outputCell","outputDisplayElement",true,"Data");

                //this adds a listener for creation or updated to a component
                //should be done before workspace is loaded
                let callback = component => alert(component.getName() + " value updated!");
                appView.addComponentListener("main.outputCell",callback);

                appView.openWorkspace(workspaceUrl);
            }

        </script>
    </head>
    <body onload="init()">
        <h1>Simple Example of Embedding a Apogee Workspace in a Web Page.</h1>
        <p>You can expose the display elements from an Apogee workspace in a web application. This page is
very ugly but you can make it much prettier.</p>

        <p>This is the input cell:</p>
        <div class="cellViewContainer" id="inputDisplayElement"></div>

        <p>This is the output cell:</p>
        <div class="cellViewContainer" id="outputDisplayElement"></div>
    </body>
```

```
</html>
```

The following needs to be done:

- Include the css file *cssBundle.css*. (Check the web site for the appropriate version.)
- Add an ES Module script (or if desired do this in another format).
    - Import the class *ApogeeWebView* from the library *apogeeWebClientLib.es.js*.
    - In an oninit function for the page
        - Instantiate an instance of ApogeeWebView
        - Add an displays of the workspace data desired
        - Add an listeners for component (cell) changes desired
        - Open the workspace.

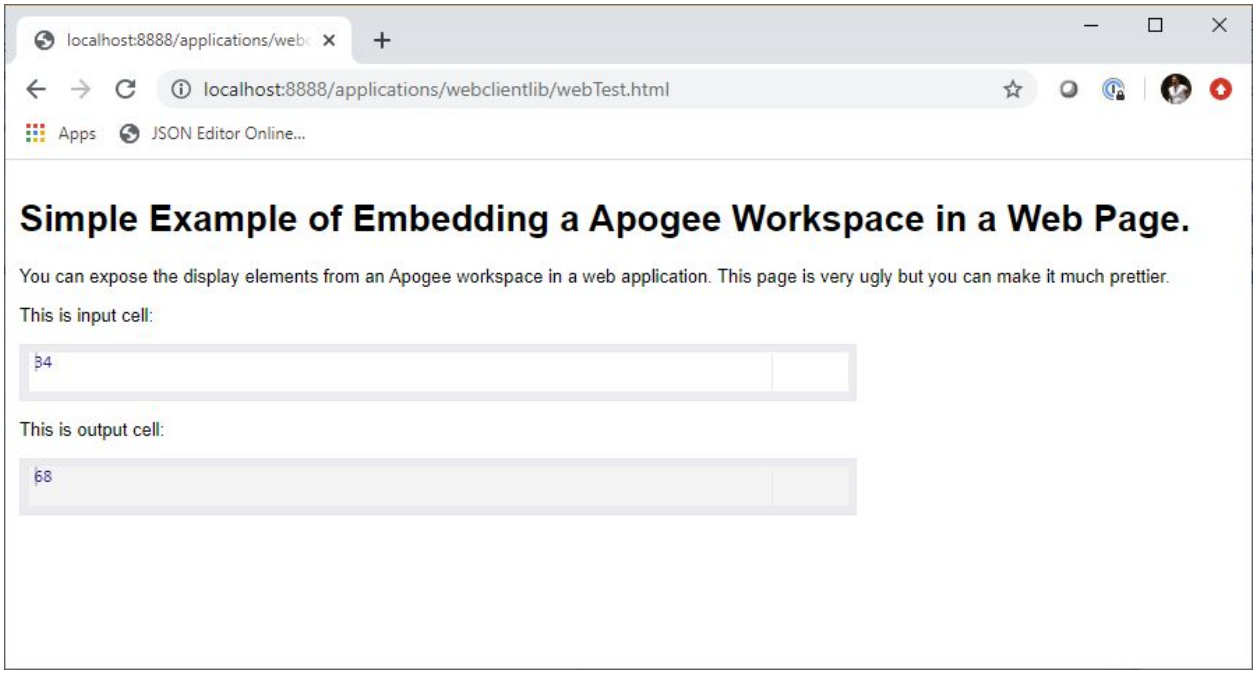
This example adds two displays. The arguments of *addDisplay* are:

- **Component Name** - The full name of the component (cell) from the Apogee workspace.
- **Target DOM element ID** - This is the DOM element that will hold the display.
- **Is Showing** - We need to say if the element is already showing in the DOM. And if we hide it, we need to tell the ApogeeWebView. See below.)
- **Display View Name** - This is the name of the view as taken from the dropdown in the Apogee application. In this case we took the view "Data".

We also add a listener that fires whenever the given component (cell) changes, including on creation. This is why the web page has the annoying alert message when you open the page or change the value in the input cell. The arguments of *addListener* are:

- Component Name - The full name of the component (cell) we are listening to.
- Callback - The function to call when the component changes.

Here is a view of our web page.



Showing and Hiding Elements

The Apogee display views may behave differently if they are showing or not showing. To guarantee proper functionality, an additional method should be used if a display view is shown or hidden.

```
appView.setIsShowing(componentName,viewName,parentIsShowing)
```

The arguments are:

- Component Name - This is the full name of the component, as used in the call to *addDisplay*.
- viewName - This is the name of the data view, as used in the call to *addDisplay*.
- Is Showing - This is true or false depending on whether the container element is showing (*true*) or hidden (*false*).

CSS for Parent Elements and Sizing

The apogee display views should be allowed to size themselves as follows:

- The display view will take up all the horizontal space available to it
- The display view will size itself in the vertical direction. The container should expand or contract to fit the displays desired size.

There are some options for controlling the size in the Apogee application. These are currently not available in the web library. However, if the size is adjusted as desired in the Apogee app, it should be that same size when exposed on the web page.

Horizontal Resizing

Some displays may subscribe to an event that fires when the horizontal size is updated. (As of the time of writing, 5/8/20, I believe this is only the data grid.) This event will fire whenever the web browser is resized. However, the element can change widths for other reasons.

There is a function available on ApogeeWebView which will fire the width resize event for all components simultaneously.

```
appView.triggerResizeWait()
```