

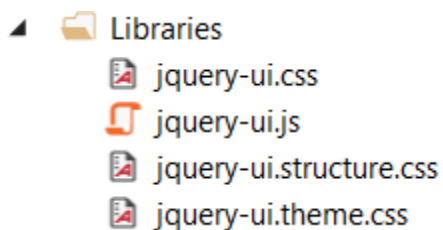
1. Introduction

- Show Power Point "Framework Control - External Libraries"
- This sample shows how to integrate a web library inside a Framework Control
- Show final solution
 - Framework Control, that uses jQuery UI
 - Two rectangles places inside the control
 - Button for lock / unlock --> Enables the user to move and anchor the rectangles
 - Rectangles can be moved at any place inside the control

2. Create new Framework Control "TcHmiExternalLibraries"

3. Include jQuery UI inside the Framework Control

- Add new folder "Libraries"
- Copy jQuery UI files on file layer in the folder (share them via mail or USB before)
- Add files to the project (Add Existing Item):



4. Open Description.json and reference the files

- Files must be included before the Source.js, that the functionalities are available there
- Include files

c.

```
48  "dependencyFiles": [  
49  {  
50    "name": "Libraries/jquery-ui.js",  
51    "type": "JavaScript",  
52    "description": "jQuery UI"  
53  },  
54  {  
55    "name": "Libraries/jquery-ui.css",  
56    "type": "Stylesheet",  
57    "description": "jQuery UI"  
58  },  
59  {  
60    "name": "Libraries/jquery-ui.structure.css",  
61    "type": "Stylesheet",  
62    "description": "jQuery UI"  
63  },  
64  {  
65    "name": "Libraries/jquery-ui.theme.css",  
66    "type": "Stylesheet",  
67    "description": "jQuery UI"  
68  },  
69  {  
70    "name": "Source.js",  
71    "type": "JavaScript",  
72    "description": "Contains all the main logic."  
73  },  
]
```

4. Add HTML elements to the control

a. Open Template.html:

```
Template.html  Description.json  Template.html  Source.js  Description.json  Desktop.view
1  <div class="tc-hmi-external-libraries-template tchmi-box">
2    <div class="tc-hmi-external-libraries-container">
3      <div class="tc-hmi-external-libraries-rectangle-1"></div>
4      <div class="tc-hmi-external-libraries-rectangle-2"></div>
5      <button class="tc-hmi-external-libraries-button">
6        Lock / Unlock
7      </button>
8    </div>
9  </div>
```

2. Add CSS classes in Style.css on project layer:

```
16 .tc-hmi-external-libraries-rectangle-1 {
17   width: 50px;
18   height: 25px;
19   position: absolute;
20   top: 10px;
21   left: 10px;
22   background-color: blue;
23 }
24
25 .tc-hmi-external-libraries-rectangle-2 {
26   width: 50px;
27   height: 25px;
28   position: absolute;
29   top: 100px;
30   left: 10px;
31   background-color: red;
32 }
33
34 .tc-hmi-external-libraries-button {
35   width: 100%;
36   height: 30px;
37   position: absolute;
38   bottom: 0px;
39 }
```

c. Build project, link the control with the HMI project, intantiate the control

5. Use jQuery UI in Source.js:

a. Declare member variables for current state (Lock/Unlock):

```
function TcHmiExternalLibraries(element, pcElement, attrs) {
  /** Call base constructor */
  _super.call(this, element, pcElement, attrs);

  // initialize internal members
  this.__draggableActive = false;
}
```

2. Grab HTML elements in __prevInit:

```
69 TcHmiExternalLibraries.prototype.__previnit = function () {
70     /** Handle template elements. Should be done before call to __previnit of super class. */
71     this.__elementTemplateRoot = this.__element.find('.tc-hmi-external-libraries-template');
72     this.__elementContainer = this.__elementTemplateRoot.find('.tc-hmi-external-libraries-container');
73     this.__elementRectangle1 = this.__elementContainer.find('.tc-hmi-external-libraries-rectangle-1');
74     this.__elementRectangle2 = this.__elementContainer.find('.tc-hmi-external-libraries-rectangle-2');
75     this.__elementButton = this.__elementContainer.find('.tc-hmi-external-libraries-button');
```

- c. Attach: Add jQuery UI classes (jQuery UI can be used directly, all of its functions are available now in the Source.js)

```
// add draggable function from JQuery UI to rectangles
this.__elementRectangle1.draggable(
{
    addClasses: true,
    disabled : true
});

this.__elementRectangle2.draggable(
{
    addClasses: true,
    disabled: true
});
```

- d. Add event listener for the button in the attach function and switch the jQuery UI classes there:

```
// reference to this for other scopes
var $this = this;

this.__elementButton.on('click', function (e) {

    if ($this.__draggableActive) {

        // set draggable active to false
        $this.__draggableActive = false;

        // disable draggable function for rectangles
        $this.__elementRectangle1.draggable('disable');
        $this.__elementRectangle2.draggable('disable');

    } else {

        // set draggable active to true
        $this.__draggableActive = true;

        // enable draggable function for rectangles
        $this.__elementRectangle1.draggable('enable');
        $this.__elementRectangle2.draggable('enable');

    }

});
```

- e. Detach: Remove event listener and remove jQuery UI classes

```
// remove event listener for button
this.__elementButton.off('click');

// unregister draggable classes
this.__elementRectangle1.draggable('destroy');
this.__elementRectangle2.draggable('destroy');
```

- 6. Build project, reload Desktop.view, open LiveView and show functionality