---

title: The MashupJS Menu

tags:

- AngularJS

- Bootstrap

- Router

- Menu

- JSON Merge

- Grunt

- Gulp

---

The MashupJS Menu

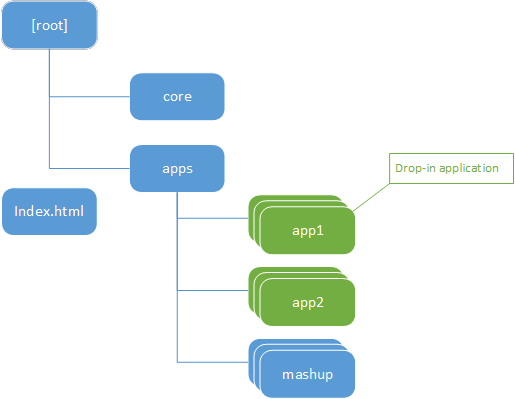
Most applications have some type of menu system linking the user to modules within the application. The MashupJS is a composite of many applications so it links the user to multiple applications via its menu system.

Developers will likely develop their applications in a separate implementation of the MashupJS and copy their applications directory to the deployment MashupJS. For this to work the MashupJS must adopt a “drop-in” approach to adding applications.

Using Grunt/Gulp the mashup not only combines routes between multiple applications into a single routing system but pulls and displays application menu items giving the user a unified view of corporate applications.

# File Structure for Drop-in Applications

In the example below app1, app2, and mashup all have items for the mashup menu.



MashupJS menu items are stored in a JSON file with the name “menu.json.txt”.

NOTE: Notice the “.txt” extension of the JSON file. IIS is initially set up to understand HTML and TXT files but not JSON file. The mime can be added but for the sake of simplicity I’ve decided to use “txt”.

<staticContent>

<remove fileExtension=".json" />

<mimeMap fileExtension=".json" mimeType="application/json" />

</staticContent>

Reference: <http://stackoverflow.com/questions/26699795/http-get-of-json-file-always-returns-404>

# JSON Merge

Using a task manager, such as Grunt/Gulp, “the menu.json.txt” files from each app is merged into a single “menu.json.txt” file and placed into the “dist” directory.

A simple Grunt configuration makes this possible.

<https://www.npmjs.com/package/grunt-merge-json>

Installing the Grunt plugin  
npm install grunt-merge-json --save-dev

Loading the Grunt module

grunt.loadNpmTasks('grunt-merge-json');

Grunt configuration

"merge-json": {

menu: {

src: ['apps/\*\*/menu.json.txt'],

dest: '<%= distFolder %>/menu.json.txt',

},

},

# Boiler plate pages

The Mashup’s core does not have its own UI controls or pages. One of the drop-in applications in the “apps” directory must host your menu and menu logic. The “apps/mashup” application starts with a couple boilerplate pages.

The “apps/mashup” should be replaces by your company’s boiler plate pages.

welcome.html

* A basic welcome page.

login.html

* Each mashup app can have its own login.html page or share a common corporate login page.

about.html

* Provides basic information about the users sessions and cached data elements.

menu.html

* Provides links between Mashup applications.
* The menu.controller.js of the application hosting the mashups menu will read and use the new “menu.json.txt”.

# menu.json date object

The menu.json data object is basic. This example only contains two levels but there is no limit on the number of levels possible.

The first level, the root of the JSON object, is the category. Initially the MashupJS has four categories but yours may different in number and name.

The four initial categories are:

Applications

* Applications you build will likely have at least one menu item in the Application category.

Utilities

* Often we are required to build simple utility screens that aren’t large enough to be considered an applications. These can be organized in the Utilities category.

Administrative

* Place to put basic user and application management pages.

Examples

* The MashupJS is a learning application. Code examples can be embedded into the MashupJS but hidden from users. Another option is to simply have another implementation of the MashupJS just for developers as a Front-End code library.

# Menu Attributes

Category Attributes

name – Name of the menu item and what will be displayed on the item

id – The id of the menu item created.

isOpen – Indicator of whether the category is open or closed.

icon – A class representing a Font Awesome icon.

session – The name of the users session used to determine if the user has access to the menu item.

role – The role required by the session to determine menu item access.

groups – List of controls to be created in the category.

Menu Item Attributes are similar to those of the Category Attributes with the exception of “isOpen”. Menu items that line to applications are not in an open or closes state.

In addition to the attributes similar to Category are the following:

Desc – This is a longer description of the menu item. Display of this depends on your applications needs and what resolution your responsive interface is loaded into.

url – The link created when the menu item is pressed.

Example of the “apps/app1” menu. This is combined with the menus from “apps/app2” and “apps\mashup” to form the final menu.json file uses by the menu system.

[

{

"name": "Applications",

"id": "catApps",

"isOpen": "true",

"icon": " fa-power-off ",

"session": "coreSession",

"role": "MashupUser",

"groups": [

{

"name": "app1",

"id": "menuItemApp1",

"desc": "Application 1, page 1.",

"url": "/app1/page1",

"icon": " fa-bar-chart ",

"session": "coreSession",

"role": "MashupUser"

}

]

},

{

"name": "Utilities",

"id": "catUtilities",

"isOpen": "false",

"icon": " fa-cogs ",

"session": "coreSession",

"role": "MashupUser",

"groups": [ ]

},

{

"name": "Administrative",

"id": "catAdmin",

"isOpen": "false",

"icon": " fa-users",

"session": "coreSession",

"role": "MashupUser",

"groups": [ ]

},

{

"name": "Examples",

"id": "catExamples",

"isOpen": "false",

"icon": " fa-file-code-o ",

"session": "coreSession",

"role": "MashupUser",

"groups": [ ]

}

]

# menu.html

The menu.html uses a simple ng-repeater to build the menu and the menu item attributes to set properties and css classes.

<div class="panel-group" id="dynamicMenu">

<div class="panel panel-default" ng-repeat="category in menuJson">

<div class="panel-heading">

<h4 class="panel-title">

<a data-toggle="collapse" data-parent="#dynamicMenu" data-target="#collapse{{category.name}}">

<i class="fa fa-lg fa-fw {{category.icon}}"></i> {{category.name}}

</a>

</h4>

</div>

<div id="collapse{{category.name}}" class="panel-collapse collapse" ng-class="{ 'in': $first }">

<div class="panel-body" ng-click="close()">

<div class="row">

<div class="col-sm-4" ng-repeat="menuitem in category.groups">

<a href="#{{menuitem.url}}" class="list-group-item vp-menu-btn">

<i class="fa fa-lg fa-fw {{menuitem.icon}}"></i><span class="h4">{{menuitem.name}}</span>

<p class="list-group-item-text">{{menuitem.desc}}</p>

</a>

</div>

</div>

</div>

</div>

</div>

</div>

# Responsive Design

The menu is responsive. You’ll likely replace this with a Bootstrap menu or some other menu of your choosing.



