

BEMT Development Guide

Version 1.1

prepared by: Daniel Reed, dan.reed@wrycan.com

Initial Setup

In order get this project running there are a few things that you need to review.

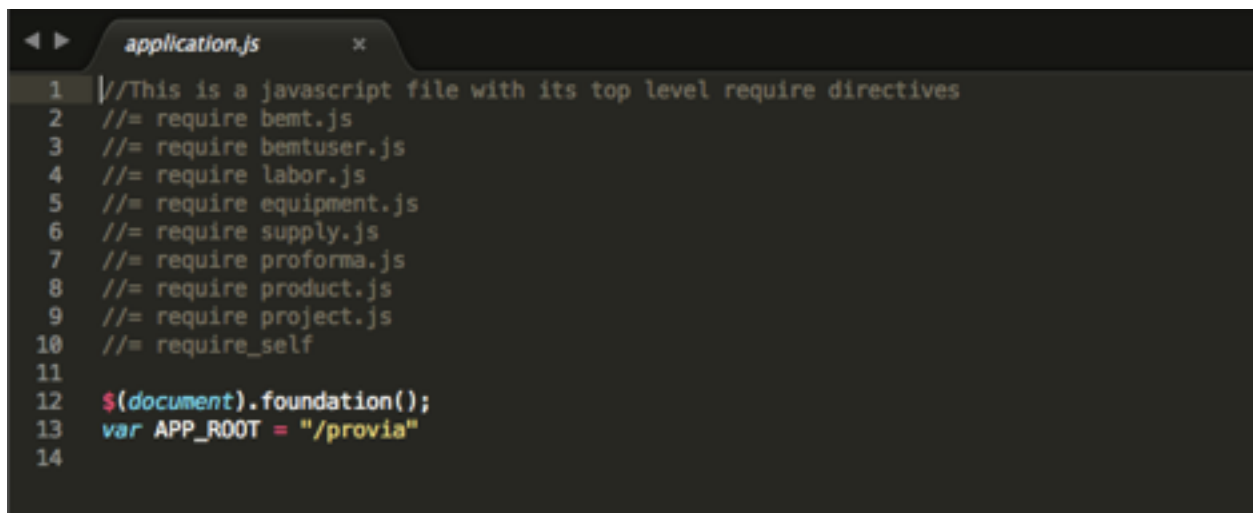
1. BootStrap file (/conf/BootStrap.groovy)

This file has all the data that should be loaded to a fresh install of this applications.
This should be executed only on a fresh install or when in DEV mode using in-memory DB.

2. Set App Root for Javascript files

There are a few places where javascript will use the REST API to access some information about equipment or labor/etc. To do this, the javascripts need the app root, something like "/provia" or whatever you set the webapp path to. This can be set in the application.js file:

```
var APP_ROOT = "/provia"
```



```
1 |//This is a javascript file with its top level require directives
2 |//= require bemt.js
3 |//= require bemtuser.js
4 |//= require labor.js
5 |//= require equipment.js
6 |//= require supply.js
7 |//= require proforma.js
8 |//= require product.js
9 |//= require project.js
10 |//= require_self
11 |
12 |$(document).foundation();
13 |var APP_ROOT = "/provia"
14 |
```

see line 13 above.

Set it to whatever you make the project root (/bemt, /tool, etc) and it will be used wherever it is needed throughout the rest of the Javascript.

Loading Data

Create the DATABASE

The easiest way to create the database is to leave the db environment in “validate” or “update” mode, and just apply the DDL to the ORACLE DB directly. I won't include instructions for applying the DDL given the tools/environments can be different.

The DDL is located in the delivery package in the /db folder.

Loading Seed Data

In order to use the application, seed data needs to be loaded. There are four seed data files that are located in the /userScripts folder:

loadAppSeed.groovy: contains data for product types, labor types, etc.

loadSurveyEnumdata.groovy: contains all the enumerations for the Survey data

loadSurveys.groovy: calls Surveys.groovy to load all the surveys

loadTemplateBiobanks.groovy: loads the template biobank (despite the name, there is only one)

The following command line should load them:

NOTE: Please keep in mind, the DB config file for the grails will be honored, so if you have the DB set to create, each script that runs will create the database, that can create some issues with the next script relying on data that is now gone. Please be sure to have your ENV set correctly, or include the ENV in the command line (dev, test, prod).

```
grails -Dtomcat=false run-script --stacktrace ./userScripts/  
loadTemplateBiobanks.groovy
```

The order to load the scripts is:

1. loadAppSeed.groovy
2. loadSurveyEnumdata.groovy
3. loadSurveys.groovy
4. loadTemplateBiobanks.groovy

UI Design and Dependencies

Design Notes

The UI relies on Javascript to do many of the operations, including automatic calculations, simple form validations (though the domain model will still rule when it comes to data validations) and some cases where asynchronous data retrieval and submission make things a but more tidy for the user.

All the javascript files they were developed as part of this project are located in the “assets/ javascripts” folder.

The “application.js” is the point for including all the others. For this project, javascript files were built for all the domain objects that needed be rendered to the user. (e.g. Equipment.js, Proforma.js, etc).

There is a Bemt.js which has common functions that are using throughout.

Javascript Modules

All the javascript files are built using a module-pattern. The idea behind the module pattern is that you can isolate your methods in a namespace'd object, so you can set up standard methods with worry about overloading methods or collisions.

Example: Equipment.js

The code in this module part of the “Equipment” namespace.

```
13 //set up namespaces so we can use these without collison
14 var Equipment = Equipment || {};
15
16 Equipment = function() {
17
18     //private properties
19
```

In order to call public methods from this module, they need to be declared as public like this:

```
83 //make module methods public
84 var oPublic = {
85     changeCategory: changeCategory,
86     updateEquipmentAllocation : updateEquipmentAllocation,
87     updateEquipmentAllocationQty : updateEquipmentAllocationQty
88
89 };
90
91 return oPublic;
92 }();
```

Once declared, one would call the “changeCategory() function by:

```
Equipment.changeCategory();
```

This is how all the javascript modules are built.

Init() and Custom Methods

The main layout file (default.gsp) will call BEMT.init() (in the bemt.js file). This is called in ALL pages. This will set up standard event handlers in the UI and set up session timeouts/etc if applicable. Notice this is where the APP_ROOT is passed in as well.

```
27     <script>
28         $(document).ready(function() {
29             BEMT.init(true, APP_ROOT);
30             <g:if test="${pageInitScript}">${pageInitScript}</g:if>
31         });
32     </script>
33     <g:render template="/components/google"/>
34 </body>
35 </html>
```

In order for different pages to call specific javascript functions to set up event listeners or execute custom logic, the `${pageInitScript}` can be passed up from views with function calls that will be called after the Init is called.

example:

```
< show.gsp x
1 <g:set var="pageInitScript" scope="request">BEMTUser.initPage();</g:set>
2 <html>
```

Vendor Libraries

The design of this application uses some css/javascript libraries in order to ensure the CSS/HTML/javascript used are compatible with all browsers.

These are found in web-app/vendor.

We use BOWER for managing these but froze the versions quite some time ago. Foundation is a tad behind and might be worth upgrading. This isn't a requirement but if you see "bower.json" files, they are used/managed by bower.

<http://bower.io/>

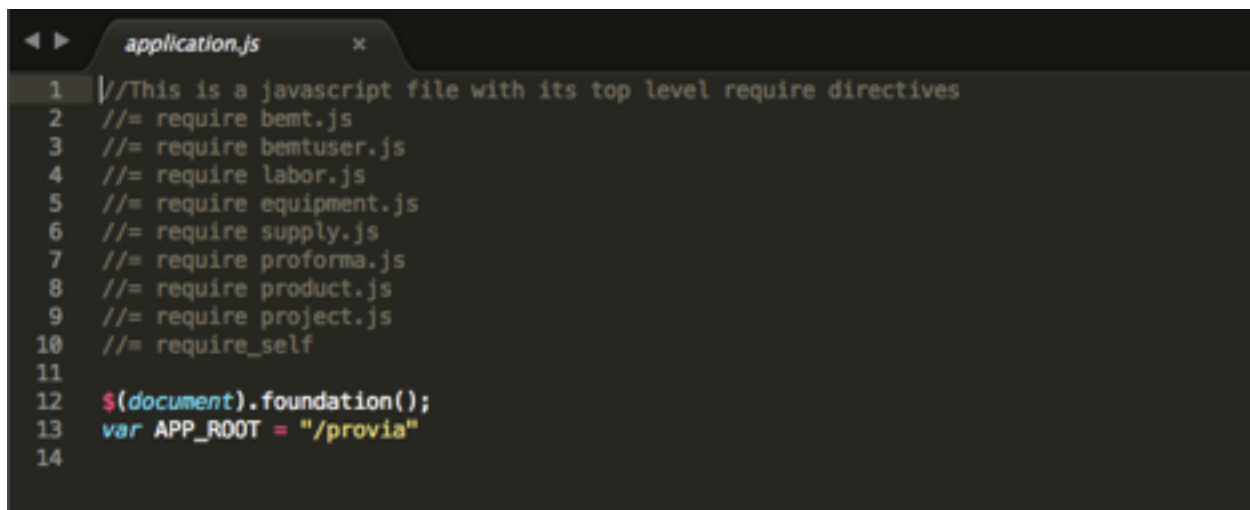
Foundation: (<http://foundation.zurb.com/>) (v5.0.2)

Foundation is the most advanced, responsive front-end framework in the world. The framework is mobile friendly and ready for you to customize it any way you want to use it.

My company has been using Foundation for 3 years and it serves as a basis for designing sites that are easy to lay out in a way that will be both desktop and mobile friendly. Best I can tell there wasn't an explicit mobile requirement for this project, but because Foundation is serving as the basis for the sign design, tweaking things to account for mobile form-factors will be easy.

Foundation consists of both CSS and JS components.

The CSS is simply included in the layouts. The javascript needs to be initialized, and is initialized in the application.js by calling `.foundation()` on the `$(document)`.



```
1 |//This is a javascript file with its top level require directives
2 |//= require bemt.js
3 |//= require bemtuser.js
4 |//= require labor.js
5 |//= require equipment.js
6 |//= require supply.js
7 |//= require proforma.js
8 |//= require product.js
9 |//= require project.js
10 |//= require_self
11
12 |$(document).foundation();
13 |var APP_ROOT = "/provia"
14
```

Modernizr (<http://modernizr.com/>) (v2.7.1)

Modernizr is a JavaScript library that detects HTML5 and CSS3 features in the user's browser. This library is required by Foundation to ensure consistent functionality across many browser/browser versions.

jQuery (<http://jquery.com/>) (v2.0.3)

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

Font-Awesome (<http://fontawesome.github.io/Font-Awesome/>) (v4.2.0)

Font Awesome gives you scalable vector icons that can instantly be customized — size, color, drop shadow, and anything that can be done with the power of CSS.

Sass/Css

The CSS files built for this project were built in Sass format.

<http://sass-lang.com/>

Sass allows for building of frameworks with CSS and ensures that style hierarchies are easier to manage.

The source sass files are located in:

src/stylesheets and the results are compiled to grails-app/assets/stylesheets

While developing using the following command will auto-compile changes to sass files to css (assuming you have sass installed).

```
sass --watch src/stylesheets:grails-app/assets/stylesheets
```

```
>>> Change detected to: /Users/dreed/Wrycan/code/svn/provia/trunk/beat_grails_app/src/stylesheets/public.scss
overwrite grails-app/assets/stylesheets/public.css
>>> Deleted template detected: /Users/dreed/Wrycan/code/svn/provia/trunk/beat_grails_app/src/stylesheets/page-templates/_home-page.scss
>>> Deleted template detected: /Users/dreed/Wrycan/code/svn/provia/trunk/beat_grails_app/src/stylesheets/public.scss
>>> Deleted template detected: /Users/dreed/Wrycan/code/svn/provia/trunk/beat_grails_app/src/stylesheets/page-templates/_public-page.scss
>>> Change detected to: /Users/dreed/Wrycan/code/svn/provia/trunk/beat_grails_app/src/stylesheets/app.scss
overwrite grails-app/assets/stylesheets/app.css
>>> Change detected to: /Users/dreed/Wrycan/code/svn/provia/trunk/beat_grails_app/src/stylesheets/app.scss
overwrite grails-app/assets/stylesheets/app.css
>>> Change detected to: /Users/dreed/Wrycan/code/svn/provia/trunk/beat_grails_app/src/stylesheets/app.scss
overwrite grails-app/assets/stylesheets/app.css
>>> Change detected to: /Users/dreed/Wrycan/code/svn/provia/trunk/beat_grails_app/src/stylesheets/app.scss
overwrite grails-app/assets/stylesheets/app.css
>>> Change detected to: /Users/dreed/Wrycan/code/svn/provia/trunk/beat_grails_app/src/stylesheets/app.scss
overwrite grails-app/assets/stylesheets/app.css
```