Manohar Nagaiah

Creating and Publishing Bower Components

Author: Manohar Nagaiah
Version: 26
Date: 13-Nov-2015 17:04

Table of Contents

This document guides you through on how to create an angular directive and publish it into private bower repository so that it can be used by any modules in AMS.

1 Step 1: Download Boiler Plate Code

Boiler plate code is nothing but a template for creating your bower component. It has the below features:

1. Predefined folder structure for developing a bower component along with npm dependencies and bower dependencies
2. Ready with Karma and Jasmine frameworks for unit testing.
3. Concatenates all the source js files into a single file (puts it in dist folder)
4. Compiles Stylus files and puts it into a single css file in dist folder

Download the bower component boiler plate

```
svn export
http://tiger.in.alcatel-lucent.com/svn/axs/web.components/templates/b
bower-component-boilerplate <your-app-name>
```

2 Step 2: Modify package.json

Modify your package.json to replace ##app.name## with your <your-app-name>

3 Step 3: install npm packages and bower packages

Run the below commands to install npm dependencies and bower dependencies

```
npm install
```

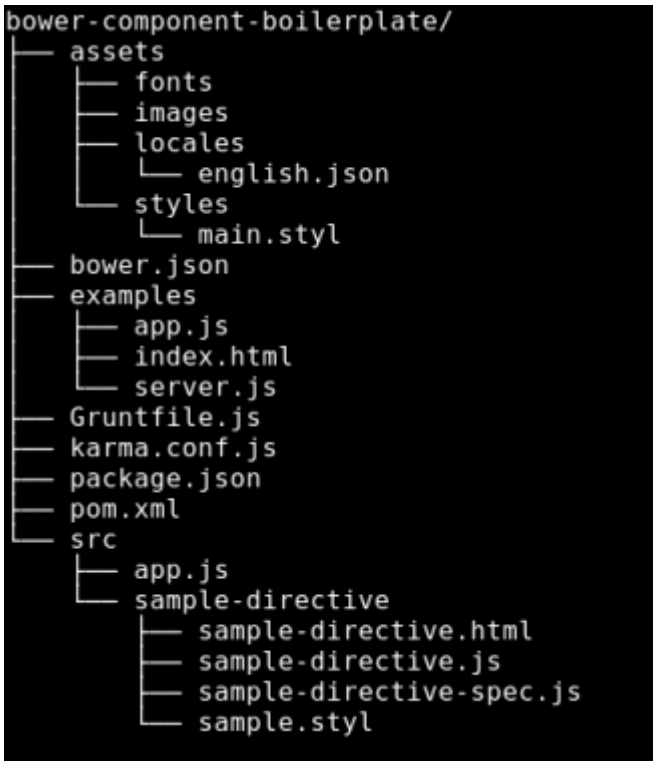
4 Step 4: initialize the app

All the files inside the folders has reference to app name as ##app.name##. Now you need to replace all these references with <your-app-name> to do this run the below command

```
grunt init-app
```

5 Step 5: Write your code

5.1 Folder Structure



assets: This folder will have all your fonts, images, locales and styles (you need to write your stylus files in the same folders as your js folder so that all the stylus files from src/**/*.styl will be injected into assets/styles/main.styl and compiled together)

examples: All your examples go inside this folder

src:

app.js will contain module definition (for angular apps) when all the js files are concatenated, contents inside app.js will be placed at the top.

under src folder multiple folders can be created for each of the directives in your bower component. each directive folder will contain stylus file, js file and unit test js .

6 Step 6: Write unit tests

Unit test js file should end with spec.js otherwise it will be concatenated to the dist file.

Use Jasmine framework to write your unit tests

7 Step 7: Build

Run maven clean install or grunt to build your app. dist folder will be generated on building your app.

8 Step 8: Commit your code

It should be made sure that *dist* folder is also committed along with src.

9 Step 9: Publish your app

9.1 Create svn folders in publish directory.

Create SVN folders in Publish repository

```
svn mkdir
http://tiger.in.alcatel-lucent.com/svn/axs/web.components/publish/bow
-m "em-commit: <your commit message here>"
svn mkdir
http://tiger.in.alcatel-lucent.com/svn/axs/web.components/publish/bow
-m "em-commit: <your commit message here>"
svn mkdir
http://tiger.in.alcatel-lucent.com/svn/axs/web.components/publish/bow
-m "em-commit: <your commit message here>"

svn co
http://tiger.in.alcatel-lucent.com/svn/axs/web.components/publish/bow
<wc location>/web.components/publish/bower
svn up
cd <your-package-name>
# Create a softlink from trunk to dist folder of your actual
source code so that the bower registry will point to the latest
revision of your source code.
svn propset svn:externals 'trunk <acutal-sourcecode-url>/dist' .
svn commit . -m "em-commit: committing svn property"
# example:
# svn propset svn:externals 'trunk
http://tiger.in.alcatel-lucent.com/svn/axs/web.components/dev/bower/a
.
# Please note the "." at the end of above commands. Which is the
target folder ( in this case target is current directory i.e. ".")
```

9.2 Create Initial version/tag in svn

```
#copy from dist folder to a tag directory

svn copy -r <last changed revision of dist folder>
<acutal-sourcecode-url>/dist
http://tiger.in.alcatel-lucent.com/svn/axs/web.components/publish/bow
version> -m "em-commit: creating initial version"
# example:
# svn copy -r 234132
http://tiger.in.alcatel-lucent.com/svn/axs/web.components/dev/bower/a
http://tiger.in.alcatel-lucent.com/svn/axs/web.components/publish/bow
```

9.3 public Bower component into private bower registry

You need to publish your bower component into AMS private bower repository so that it can be used by any of the AMS applications

Publish to private bower

```
# bower register <your-package-name> <svn repository url>
# Note: svn url should start with svn+http://
#example:
bower register ams-hotkeys
svn+http://tiger.in.alcatel-lucent.com/svn/axs/web.components/publish
```

9.4 Create new versions of your bower components

Bower follows Semantic Versioning

Create tag for your bower component

```
svn copy -r <revision>
http://tiger.in.alcatel-lucent.com/svn/axs/web.components/dev/bower/a
http://tiger.in.alcatel-lucent.com/svn/axs/web.components/publish/bow
```

10 Step 10: Use your bower component in other applications

Your bower component can be used in any of the AMS applications by referring to the private bower registry

create a file named .bowerrc under the root folder of your application and put the below contents there.

```
{
    "registry":"http://avatar.in.alcatel-lucent.com:5678"
}
```

Install the newly created bower component

```
bower install ams-hotkeys --save
# using --save will add an entry into your bower.json.
# If you want to use a specific version of your bower component,
then run bower install ams-hotkeys#1.0.1
```