Resources Plugin

Why do I want this? *

Paul Woods

8/1/2012

https://github.com/paulwoods/resources-talk

* = because is Good For You™

What's the problem?

- Our web pages are growing.
- We used store all of our page code on one file.
- The server only had to send one file to render a page *

* = not including images and media, of course

What's the problem?

- In 1994 CSS was invented, so we can put CSS into its own file *
- Now our servers are sending 2 files to render a page – HTML as CSS

- * = http://www.w3.org/Style/LieBos2e/history/

What's the problem?

- Somewhere in the early 2000s, we started extracting our javascript from the HTML file and put it into its own file

- Now our servers are sending 3 files to render a page – HTML, CSS and JS

What's the problem?

- But now, those CSS files grew and grew and grew
- So, we split our CSS files into multiple CSS files
- Now our servers are sending lots files to render a page – HTML, multple CSS and JS

What's the problem?

- Guess what happened to the javascript?
- Yep – same thing. It got so big, we needed to split it into multiple files.
- Now our servers are sending Lord-knows-how-many files to render a page – HTML, multple CSS and multiple JS

What's the problem?

- Did I forget to mention 3rd party javascript?

- We use lots of it to make our sites better, but that's still more files for our server to send.

The servers were sad

- They were serving more and more files to more and more users.
- They responded by making our sites slow.
- Our users (and bosses) don't like slow sites.

Solutions! FTW

- Very Smart People™ came up with methods to fix this problem:
  - Caching – telling the browser to retain data so it doesn't need to be downloaded again.
  - GZip – Shrinking the files so there is less data to download.
  - Minification – Rewriting CSS and Javascript files so that they are smaller and combined into one file.

Wouldn't it be nice if I could…

- join all of the .css files into one file on the server.
- join all of the .js files into one file on the server.
- serve just the one css file.
- serve just the one js file.

- configure which files go into the css file.
- configure which files go into the js file.

- control the order of the data in the file

- configure globally and by-page
- cache those files
- minify those files

One Solution to Rule Them All*

- We can use the Grails Resources** plugin!
- Can combine multiple css or javascript files into a single file, at runtime.
- Configurable, to allow us to change which files are used on which pages.
- Extendable – With additional plugins we can add caching and minifying.

- * = actually, multiple solutions combined into one.
- ** = your VSPs are Marc Palmer and Luke Daley

Demo Site 1

- Site one is a one-page app that has several css widgets and javascript files.
- We want to use the resources plugin to combine the .css into one file, and the .js files into one file.

Demo Site 1 – Head 1

- Four CSS files are linked.

```
<link rel="stylesheet" href="${resource(dir: 'css', file: 'widget1.css')}" type="text/css">
<link rel="stylesheet" href="${resource(dir: 'css', file: 'widget2.css')}" type="text/css">
<link rel="stylesheet" href="${resource(dir: 'css', file: 'widget3.css')}" type="text/css">
<link rel="stylesheet" href="${resource(dir: 'css', file: 'widget4.css')}" type="text/css">
```

Demo Site 1 – Head 2

- Seven JavaScript files are linked.

```
<script type="text/javascript" src="${g.resource(dir:'js', file:'jquery-1.7.2.js')}"></script>
<script type="text/javascript" src="${g.resource(dir:'js', file:'underscore-1.3.3.js')}"></script>
<script type="text/javascript" src="${g.resource(dir:'js', file:'modernizr-2.5.2.js')}"></script>
<script type="text/javascript" src="${g.resource(dir:'js', file:'backbone-0.9.2.js')}"></script>
<script type="text/javascript" src="${g.resource(dir:'js', file:'handlebars-1.0.0.beta.6.js')}"></script>
<script type="text/javascript" src="${g.resource(dir:'js', file:'jasmine-1.2.0.js')}"></script>
<script type="text/javascript" src="${g.resource(dir:'js', file:'index-1.0.0.js')}"></script>
```

Demo Site 1 – Head 3

- And one local JavaScript

```
<script type="text/javascript">

(function(window, $, undefined) {
        $(function() {
                $("#jquery").html( jQuery.fn.jquery );
                $("#underscore").html( _.VERSION );
                $("#modernizr").html( Modernizr._version );
                $("#backbone").html( Backbone.VERSION );
                $("#handlebars").html( Handlebars.VERSION );
                $("#jasmine").html( jasmine.getEnv().versionString() );
                $("#index").html( site1.VERSION );
        });
}(window, jQuery));

</script>
```

Demo Site 1 – Body

```
<table>
<thead>
<tr><th>File</th><th>Version</th></tr>
</thead>
<tbody>
<tr><td>jquery</td><td id="jquery">---</td></tr>
<tr><td>underscore</td><td id="underscore">---</td></tr>
<tr><td>modernizr</td><td id="modernizr">---</td></tr>
<tr><td>backbone</td><td id="backbone">---</td></tr>
<tr><td>handlebars</td><td id="handlebars">---</td></tr>
<tr><td>jasmine</td><td id="jasmine">---</td></tr>
<tr><td>index</td><td id="index">---</td></tr>
</tbody>
</table>

<h2 class="widget1">Widget 1</h2>
<h2 class="widget2">Widget 2</h2>
<h2 class="widget3">Widget 3</h2>
<h2 class="widget4">Widget 4</h2>
```

Demo Site 1 – Screen Shot

Lets Use Resources - 1

grails install-plugin resources

* = grails 2.0 already has it installed by default.

Lets Use Resources - 2

- Create or Edit the /conf/ApplicationResources.groovy file*
- Add a module closure variable
- Insert a named closure for your files (aka the 'Module')

* = applications prior to grails 2.0 will not have this file. The file name does not have to be ApplicationResources.groovy. It can be anything that ends in Resources.groovy

Lets Use Resources - 3

site1b/grails-app/conf/ApplicationResources.groovy

```
modules = {

        site1b {
                resource url:'css/widget1.css'
                resource url:'css/widget2.css'
                resource url:'css/widget3.css'
                resource url:'css/widget4.css'
                resource url:'js/jquery-1.7.2.js'
                resource url:'js/underscore-1.3.3.js'
                resource url:'js/modernizr-2.5.2.js'
                resource url:'js/backbone-0.9.2.js'
                resource url:'js/handlebars-1.0.0.beta.6.js'
                resource url:'js/jasmine-1.2.0.js'
                resource url:'js/index-1.0.0.js'
        }
}
```

The resources are similar to the g.resource tag, but these are now r.resource.

Lets Use Resources - 4

- In the head section, add r:require and r.layoutResources

```
<head>
  <r:require module="site1b"/>
  …
<r:layoutResources/>
</head>
```

Lets Use Resources - 5

- Change any <script> tags to <r:script>

```
<r:script>
(function(window, $, undefined) {
    $(function() {
            $("#jquery").html( jQuery.fn.jquery );
            $("#underscore").html( _.VERSION );
            $("#modernizr").html( Modernizr._version );
            $("#backbone").html( Backbone.VERSION );
            $("#handlebars").html( Handlebars.VERSION );
            $("#jasmine").html( jasmine.getEnv().versionString() );
            $("#index").html( site1.VERSION );
    });
}(window, jQuery));
</r:script>
```

Demo Site2 – Using Resources

Demo Site2 – Source Code 1

- Looks exactally the same, but look under the hood (view sources):
  - In the header the CSS files have been reduced into one file*:
  - The <script> javascript code is gone from the header.

<link href="/site1b/static/bundle-bundle_site1b_head.css" type="text/css“ rel="stylesheet" media="screen, projection" />

* = there are other .css files in the header. The are part of the layout main.gsp We will remove those later.

** = Click on the link to see what is in the .css file.

Demo Site 2 – Source Code 2

- At the end of the body, these is a single linked javascript file.

```
<script src="/site1b/static/bundle-bundle_site1b_defer.js"
    type="text/javascript" ></script>
```

* = click on the javascript link. You will see that all of the .js files have been bundled into one.

Demo Site2 – Source Code 3

- Our script file is now at the bottom of the body.

```
(function(window, $, undefined) {
    $(function() {
    $("#jquery").html( jQuery.fn.jquery );
    $("#underscore").html( _.VERSION );
    $("#modernizr").html( Modernizr._version );
    $("#backbone").html( Backbone.VERSION );
    $("#handlebars").html( Handlebars.VERSION );
    $("#jasmine").html( jasmine.getEnv().versionString() );
    $("#index").html( site1.VERSION );
    });
}(window, jQuery));

</script>
</body>
```

Wait, What, How?

- The plugin took the .css files configured in ApplicationResources, combined them into one file, and made it available at uri:

  /site1b/static/bundle-bundle_site1b_head.css

- It did the same thing with the javascript and made it available at

  /site1b/static/bundle-bundle_site1b_defer.js

But, it moved my javascript

- Yep. Today's practices recommend that javascript be moved to the bottom of the body tag. This allows all of the HTML to be loaded into the DOM before the javascript runs.
- Most javascript runs just fine this way, but if you need, you can configure files to be in the head.

Definitions

- Lets get a few of the definitions out of the way…

Definitions – module

- A module is a group of resources that will be included together on the page. They are defined in ApplicationResources.groovy.

```
jQuery {
    resource url:[dir:"js", file:"jquery-ui-1.8.20.custom.min.js"]
    resource url:[dir:"css", file:"jquery-ui-1.8.20.custom.css"]
}
```

Definitions – disposition

- The disposition tells the plugin where (head or body) to place the resource. Use values head or defer. This example puts modernizr in the head and jQuery in the body

```
modules = {
    core {
        resource url:'js/modernizr-2.5.2.js', disposition: 'head'
        resource url:'js/jquery-1.7.2.js'
    }
}
```

Definitions - bundle

- The bundle is the name given to the files in a module, and it is used in the url used to download the files. By default, it's the same as the module, but you can set it using "defaultBundle"

```
jQuery {
    defaultBundle "ui"
    resource url:[dir:"js", file:"jquery-ui-
    1.8.20.custom.min.js"]
    resource url:[dir:"css", file:"jquery-ui-1.8.20.custom.css"]
}
```

Definitions – r:require

- This tag is placed in the head of your page. It tells the resource plugin which module(s) to use on this page.

```
<head>
    <meta name="layout" content="main"/>
    <r:require module="jQuery"/>
    …
```

Definitions – r:layoutResources

- This tag denotes the location where the bundled file links will be placed. It must appear twice on your page – once in the head and once in the body. Normally put this tag at the bottom of the head and the bottom of the body.

```
<head>
        …
        <r:layoutResources/>
</head>
<body>
        …
        <r:layoutResources/>
</body>
```

Definitions – r:script

- The r:script tag is similar to the <script> tag, except the plugin may move the javascript to either the head or the body of the page.

```
<r:script>
    window.alert('This is the end of the page!');
</r:script>
<r:script disposition='head'>
    window.alert('This is the head of the page!');
</r:script>
```

Demo 3

- In our previous demo, we included Modernizr, but it was incorrect - it needs to in the head, not the body. Lets update our ApplicationResources.groovy

Demo 3

```
modules = {

    site3 {
            resource url:'css/widget1.css'
            resource url:'css/widget2.css'
            resource url:'css/widget3.css'
            resource url:'css/widget4.css'
            resource url:'js/jquery-1.7.2.js'
            resource url:'js/underscore-1.3.3.js'
            resource url:'js/modernizr-2.5.2.js', disposition:"head"
            resource url:'js/backbone-0.9.2.js'
            resource url:'js/handlebars-1.0.0.beta.6.js'
            resource url:'js/jasmine-1.2.0.js'
            resource url:'js/index-1.0.0.js'
    }
}
```

Demo 3 - head

```
<head>
    ...
    <link href="/site3/static/bundle-bundle_site3_head.css"
    type="text/css" rel="stylesheet" media="screen, projection"
    />

    <script src="/site3/static/bundle-bundle_site3_head.js"
    type="text/javascript" ></script>

</head>
```

We now have a javascript file in the header.
Click on the link to see the contents.
(spoiler alert: it is the modernizr file).

Multiple Modules

- Lets move the modernizr code into its own module, and the backbone files into their own module.

Demo 4 – multi modules

- Update ApplicationResources.groovy to use 3 modules

```
modules = {
        modernizr {
                resource url:'js/modernizr-2.5.2.js', disposition: 'head'
        }
        backbone {
                resource url:'js/underscore-1.3.3.js'
                resource url:'js/backbone-0.9.2.js'
                resource url:'js/handlebars-1.0.0.beta.6.js'
        }
        site3 {
                resource url:'css/widget1.css'
                resource url:'css/widget2.css'
                resource url:'css/widget3.css'
                resource url:'css/widget4.css'
                resource url:'js/jquery-1.7.2.js'
                resource url:'js/jasmine-1.2.0.js'
                resource url:'js/index-1.0.0.js'
        }
}
```

Demo 4 – r:requre

- Add the three modules (comma separated) to the r:require tag:

<r:require modules="modernizr, backbone, site4" />

Note: ‘module=‘ was changed to ‘modules=‘

Demo 4 – Source

head:

<link href="/site4/static/bundle-bundle_site4_head.css" …

<script src="/site4/static/js/modernizr-2.5.2.js“ ...

body:

<script src="/site4/static/bundle-bundle_site4_defer.js“…

<script src="/site4/static/bundle-bundle_backbone_defer.js" …

Each module is placed in its own bundle. Notice there are 2 javascript bundles in the body.

Merging Bundles

- Since there are 2 bundles in the body, we can combine them into one bundle by using defaultBundle.
- By giving the modules the same defaultBundle name, they will be placed in the same bundle.

Demo 5

Add defaultBundle to the modules.

```
modules = {
        modernizr {
                defaultBundle “ui”
                resource url:'js/modernizr-2.5.2.js‘, disposition: ’head’
        }
        backbone {
                defaultBundle “ui”
                resource url:'js/underscore-1.3.3.js'
                resource url:'js/backbone-0.9.2.js'
                resource url:'js/handlebars-1.0.0.beta.6.js'
        }
        site3 {
                defaultBundle “ui”
                resource url:'css/widget1.css'
                resource url:'css/widget2.css'
                resource url:'css/widget3.css'
                resource url:'css/widget4.css'
                resource url:'js/jquery-1.7.2.js'
                resource url:'js/jasmine-1.2.0.js'
                resource url:'js/index-1.0.0.js'
        }
}
```

Demo 5 - Source

head:

<link href="/site5/static/bundle-ui_head.css" …

<script src="/site5/static/bundle-ui_head.js " ...

body:

<script src="/site5/static/bundle-ui_defer.js "…

Now, The body only has only one bundle

Ordering Dependencies

- If one module depends on another (for instance if site5 depends on jquery) use dependsOn to specify the dependency

Demo 6 - ApplicationResources

```
modules = {
        modernizr {
                defaultBundle 'ui'
                resource url:'js/modernizr-2.5.2.js', disposition: 'head'
        }
        backbone {
                defaultBundle 'ui'
                resource url:'js/underscore-1.3.3.js'
                resource url:'js/backbone-0.9.2.js'
                resource url:'js/handlebars-1.0.0.beta.6.js'
        }
        site6 {
                dependsOn "modernizr, backbone"
                defaultBundle 'ui'
                resource url:'css/widget1.css'
                resource url:'css/widget2.css'
                resource url:'css/widget3.css'
                resource url:'css/widget4.css'
                resource url:'js/jquery-1.7.2.js'
                resource url:'js/jasmine-1.2.0.js'
                resource url:'js/index-1.0.0.js'
        }
}
```

Demo 6 – index.gsp

- Since site6 pulls in modernizr and backbone, update r:modules to only require site6

```
<r:require module="site6"/>
```

Demo 6 – source

- Since site6 pulls in modernizr and backbone, update r:modules to only require site6

head:

<link href="/site6/static/bundle-ui_head.css" …

<script src="/site6/static/bundle-ui_head.js " ...

body:

<script src="/site6/static/bundle-ui_defer.js "…

Other Plugins

- cache-resources
  - Tells the browser to cache the bundled files long-term
- zipped-resources
  - Tells the server to send g-zipped files when possible.
- yui-minify-resources
  - Minifies the css and javascript files

Troubleshooting

- If you have missing data in your bundled files, you can enable debug mode.
  - add _debugResources=y to your url
    - Turns off bundling and processing for the url
  - add grails.resources.debug = true to config.groovy
    - Turns off bundling and processing for the app
- Enable logging for the plugin
  - debug "org.grails.plugin.resource"