Krinkle chore(refactor): simplify data.files creation ... History

20 contributors +8

144 lines (126 sloc) | 4.02 KB ...

```
1  /*
2   * grunt-karma
3   * https://github.com/karma-runner/grunt-karma
4   *
5   * Copyright (c) 2013 Dave Geddes
6   * Licensed under the MIT license.
7   */
8
9  var runner = require('karma').runner
10 var Server = require('karma').Server
11 var path = require('path')
12 var _ = require('lodash')
13
14 function finished (code) {
15   return this(code === 0)
16 }
17
18 // Parse out all cli arguments in the form of `--arg=something` or
19 // `-c=otherthing` and return the array.
20 function parseArgs (args) {
21   return _.filter(args, function (arg) {
22     return arg.match(/^--?/)
23   })
24 }
25
26 module.exports = function (grunt) {
27   grunt.registerMultiTask('karma', 'run karma.', function () {
28     var done = this.async()
29     var options = this.options({
```

```
      background: false,
      client: {}
    })

    // Allow for passing cli arguments to `client.args` using  `--grep=x`
    var args = parseArgs(process.argv.slice(2))
    if (options.client && _.isArray(options.client.args)) {
      args = options.client.args.concat(args)
    }

    // If arguments are provided we pass them to karma
    if (args.length > 0) {
      if (!options.client) {
        options.client = {}
      }
      options.client.args = args
    }

    // Only create client info if data is provided
    if (options.client) {
      // Merge karma default options
      _.defaults(options.client, {
        args: []
      })
    }

    var opts = _.cloneDeep(options)
    // Merge options onto data, with data taking precedence.
    var data = _.merge(opts, this.data)

    // But override the browsers array.
    if (data.browsers && this.data.browsers) {
      data.browsers = this.data.browsers
    }

    // Merge client.args
    if (this.data.client && _.isArray(this.data.client.args)) {
      data.client.args = this.data.client.args.concat(options.client.args)
    }

    if (data.configFile) {
      data.configFile = path.resolve(data.configFile)
    }

    // Combines both sets of files. The order should be:
    // - first, values from options.files,
    // - then, values from this.files.
    if (options.files || this.files.length) {
      // For our 'files' option, we support arbitrarily nested arrays,
```

```
79        // as a convenient way to specify files without the user needing to
80        // concat or flatten anything within their Gruntfile.
81        data.files = _.flattenDeep(options.files || [])
82        // The 'files' task data expanded by Grunt internally produces
83        // a structure that is exactly 2 levels deep.
84        this.files.forEach((file) => {
85          file.src.forEach((src) => {
86            let obj = {
87              pattern: src
88            };
89            ['watched', 'served', 'included'].forEach((opt) => {
90              if (opt in file) {
91                obj[opt] = file[opt]
92              }
93            })
94            data.files.push(obj)
95          })
96        })
97      }
98
99      // Allow the use of templates in preprocessors
100     if (_.isPlainObject(data.preprocessors)) {
101       var preprocessors = {}
102       Object.keys(data.preprocessors).forEach(function (key) {
103         var value = data.preprocessors[key]
104         if (options.basePath) {
105           key = path.join(options.basePath, key)
106         }
107         key = path.resolve(key)
108         key = grunt.template.process(key)
109         preprocessors[key] = value
110       })
111       data.preprocessors = preprocessors
112     }
113
114     // support `karma run`, useful for grunt watch
115     if (this.flags.run) {
116       runner.run(data, finished.bind(done))
117       return
118     }
119
120     // allow karma to be run in the background so it doesn't block grunt
121     if (data.background) {
122       var backgroundProcess = require('child_process').fork(
123         path.join(__dirname, '..', 'lib', 'background.js')
124       )
125
126       backgroundProcess.on('close', function (code) {
127         var error = code
```

```
128         if (error) {
129           grunt.log.error('background karma process exited with error (code: ' + code + ')')
130         }
131       })
132
133       process.on('exit', function () {
134         backgroundProcess.kill()
135       })
136
137       backgroundProcess.send({ config: data })
138       done()
139     } else {
140       var server = new Server(data, finished.bind(done))
141       server.start()
142     }
143   })
144 }
```