Extending Build to the Client: A Maven User's Guide to Grunt.js

CREATE THE FUTURE

Petr Jiricka
Software Development Manager
NetBeans IDE team, Oracle
September 29, 2014

JavaOne ORACLE

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Program Agenda

1. Evolution of Java Web Applications
2. Introducing Grunt.js
3. Moving from Maven to Maven + Grunt.js
4. Compilation, Minification, Unit testing, …
5. Development Cycle with Grunt.js and NetBeans IDE

Evolution of Java Web Applications

Traditional Java Web Application

- Server-side
  - Server-side business logic and data access code
  - **Presentation logic and view controllers, template processing**
- Client-side
  - Views using a template-based framework (JSF, Wicket, Spring MVC, …)
- Packaged together into a `.war` file, typically using Maven

Modern Java Web Application

- Server-side
  - Server-side business logic and data access code
  - Exposed as RESTful services (typically JSON) — “thin server”
- Client-side
  - MVVM JavaScript framework (Model-View-Viewmodel): AngularJS, Ember, Backbone, Knockout, …
  - Client-side navigation logic, “single-page application”
  - Consumes JSON data from the server
- Typically still packaged together into a `.war` file using Maven

Client-side build tasks using Maven?

- How do we do:
  - Client-side library management analogous to Maven repositories?
  - Compilation of preprocessor languages like SASS, LESS
  - Compilation of languages like CoffeeScript, TypeScript?
  - Optimization of static files (concatenation, minification, compression, image sprites)?
  - Versioning of static resources to enable long-term caching by browsers?
  - Unit testing and integration testing of JavaScript files?
- And what is the overall development cycle?

Introducing Grunt.js and Related Tools

gruntjs.com

First, let's talk about Node.js

- JavaScript runtime for server applications, but also for **command-line apps**
- Includes **node package manager** (npm), online repository of packages
- Packages installed globally or per-project
  - Per-project packages specified in `package.json` file
- Platform-specific installers available at nodejs.org
  - Installer adds `node` and `npm` commands on your system PATH
- Tip: Use Node Version Manager (nvm) to manage Node.js distribution(s)
  - See https://github.com/creationix/nvm

Next, we need to introduce Bower

- Package manager for the client-side
  - Manages frameworks, libraries, assets, …
  - Supports various sources: `git` or `svn` repositories, download from URLs, archives, …
  - Controlled by `bower.json` (and `.bowerrc`) files
- Install using npm, then use bower command:
  - `npm install -g bower` : installs bower as a global Node.js module
  - `bower install --save jquery` : installs jQuery (latest version) to this project and adds an entry to `bower.json`
  - `bower install` : installs everything specified in `bower.json`

And finally, Grunt.js

- JavaScript task runner — or simply a build tool
- Install using npm:
  - `npm install --save-dev grunt` : installs Grunt in this project and saves it to the project's `package.json`
  - `npm install -g grunt-cli` : installs the Grunt command as a global Node.js module (and adds it to your PATH)
- Gruntfile.js defines and configures tasks

Grunt plugins and tasks

- **Plugins** provide predefined functionality and make it available as **tasks**
- Installing a plugin:
  - `npm install --save-dev grunt-usemin` : installs/saves **grunt-usemin** *plugin* (which provides **useminPrepare** and **usemin** *tasks*) in this project

Configuring tasks and loading plugins in Gruntfile.js

```
grunt.initConfig({
  usemin : {
    options: {
      dirs: ['dist']
    },
    html: ['dist/{,*/}*.html']
  }
});

// Load the plugin that provides the "usemin" task.
grunt.loadNpmTasks('grunt-usemin');
```

Custom tasks, default task

```
grunt.registerTask('clean-build',
  'Main clean/build task with SASS preprocessing.',
  ['clean:all', 'build', 'sass']);

// Task to run when running ‘grunt’ without parameters
grunt.registerTask('default', ['clean-build']);
```

Grunt Tips

```
// Time how long tasks take. Can help optimize build times.
require('time-grunt')(grunt);

// Load grunt tasks automatically
require('load-grunt-tasks')(grunt);
```

- You also need this in `package.json`:

```
  "devDependencies": {
    "time-grunt": "^0.4.0",
    "load-grunt-tasks": "^0.4.0",
  }
```

Moving from Maven to Maven + Grunt.js

Changing the project's source structure

Before (Maven)

```
project
├── src
├── target
└── pom.xml
```

- Sources moved from `src/main/webapp` to `client/public_html`
- Need CORS (cross-origin resource sharing) filter for development-time

After (Maven + Grunt.js)

```
project
├─┬ client
│ ├── bower_components
│ ├── node_modules
│ ├── public_html
│ ├── dist
│ ├── bower.json
│ ├── Gruntfile.js
│ └── package.json
└─┬ server
  ├── src
  ├── target
  └── pom.xml
```

From Source to Production

- Bower libraries downloaded to a separate staging area
- Bower libraries copied to source tree
  - NetBeans recommended approach
- SASS compilation
- Unit testing

- Concatenation
- Minification
- Static resource revisions

Manage libraries using Bower

- Declare libraries in `bower.json`, instead of placing them directly under the web root
- Copy them under e.g. `public_html/libs/vendor` using the **bowercopy** grunt task (from the **grunt-bowercopy** node plugin)
- Many Bower components contain “junk” other than the binary itself, so need to be careful what to copy

Bundle client and server into a .war file

- Top-level maven project called `bundle` (next to `client` and `server`)
- `pom.xml` uses `war` packaging type
- uses `grunt-maven-plugin` to call Grunt before packaging the result
  - https://github.com/allegro/grunt-maven-plugin

Compilation, Minification, Unit testing, …

Compile SASS to CSS

- Uses `sass` task (from the `grunt-contrib-sass` node plugin)

```
sass: {
  dist: {
    files: {
      'main.css': 'main.scss',
      'widgets.css': 'widgets.scss'
    }
  }
}
```

Minify and concatenate JavaScript and CSS

- Several tasks orchestrated by the `grunt-usemin` node plugin
  - `useminPrepare` prepares the configuration
  - `concat` concatenates files (usually JS or CSS).
  - `uglify` minifies JS files.
  - `cssmin` minifies CSS files.
  - `filerev` revisions static assets through a file content hash.
  - `usemin` replaces references in HTML files
- Output should be written to a separate `dist` directory
- `copy` task should copy all the other files from `public_html` to `dist`

Unit testing using Karma

- place test sources under `client/test` (sibling of `public_html`)
- `karma.conf.js` file controls the configuration of Karma
- `grunt-karma` plugin provides `karma` task with several subtasks
  - `grunt.registerTask('test', ['karma:run']); // PhantomJS`
  - `grunt.registerTask('test-chrome', ['karma:run-chrome']);`
  - `grunt.registerTask('test-coverage', ['karma:coverage']);`
  - `grunt.registerTask('test-debug', ['karma:debug']);`

Development Cycle with Grunt.js and NetBeans IDE

Development workflow from the command line

- `watch` task (`grunt-contrib-watch` plugin) watches for changes in compiled files and recompiles them
- Serve files using a lightweight web server
- Reload files in the browser

NetBeans IDE

- Great editor for JavaScript, HTML, CSS, SASS, LESS, …
  - Supports frameworks and libraries like require.js, AngularJS, Knockout, …
- Running, debugging and Visual CSS editing not only on Chrome — also on iOS and Android devices and simulators
  - Including support for packaging applications using the Cordova framework
- Built-in lightweight web server, recompile and refresh browser on save
- Built-in test runner
- … and many more features

Resources

- Grunt.js: gruntjs.com
- Node.js: nodejs.org
- Bower: bower.io
- Karma: karma-runner.github.io
- SASS: sass-lang.com
- NetBeans IDE: netbeans.org
- Project sources: https://github.com/pjiricka/affablebean-maven-grunt
- Slides: http://www.slideshare.net/PetrJiricka

JavaOne™

ORACLE


ORACLE