1.– 4. September 2014
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Grunt.js

Im Schweinsgalopp zum Ziel

Frank Goraus

MATHEMA Software GmbH

Steckbrief

- „Task Runner“
- Automatisierung von kleinen Tasks
- in JavaScript geschrieben
- von Ben Alman („Cowboy“, jQuery-Developer)
- veröffentlicht Anfang 2012
- aktuelle Version: v.0.4.5
- Plugin-Support (3390+ Einträge)

Voraussetzungen

- Node.js
  - npm (Node Package Manager)
  - grunt-cli (Grunt Command Line Interface)

Grunt-CLI

- einfache Installation
  - „npm install -g grunt-cli“
- stellt „grunt“ Kommando zur Verfügung
- nicht Grunt selbst!
  - jedes Projekt kann eigene Grunt-Version haben
  - CLI kümmert sich nur um Aufruf dieser

Grunt-Projekt

- besteht aus
  - package.json
  - Gruntfile(.js oder .coffee)
  - Projekt-Sourcen und -Resourcen

package.json (1/2)

- Informationen über das Projekt
- Definition von Abhängigkeiten

- Kommandos:
  - „npm init“ - Erzeugt package.json
  - „npm install“ - Installiert alle Dependencies
  - „npm install <module> --save-dev“ - Installiert <module>, persistent (=Eintrag wird zu package.json hinzugefügt)

package.json (2/2)

- Beispiel:

```
{
  "name": "grunt-demo",
  "version": "0.1.0",
  "devDependencies": {
    "grunt": "~0.4.5",
    "grunt-contrib-jshint": "~0.10.0",
    "grunt-contrib-nodeunit": "~0.4.1",
    "grunt-contrib-uglify": "~0.5.0"
  }
}
```

Gruntfile (1/3)

- Definition der Tasks

- besteht aus:
  - „Wrapper“-Funktion
  - Projekt- und Task-Konfigurationen
  - Aufruf von Grunt-Plugins und -Tasks
  - eigene Tasks

Gruntfile (2/3)

- Beispiel:

```
module.exports = function(grunt) {

  grunt.initConfig({
    pkg: grunt.file.readJSON('package.json'),
    uglify: {
      options: {
          banner: '/*! <%= pkg.name %>
          <%= grunt.template.today("yyyy-mm-dd") %> */\n'
      },
```

Gruntfile (3/3)

```
        build: {
            src: 'src/<%= pkg.name %>.js',
            dest: 'build/<%= pkg.name %>.min.js'
        }
    }
});

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

// Default task(s).
grunt.registerTask('default', ['uglify']);
};
```

Task Konfiguration (1/5)

- selber Name wie Task
  - Task: „**concat**“

    → grunt.initConfig({ **concat: { … }** });

- Targets
  - unterschiedliche Konfigurationen für z.B. verschiedene Zielplattformen
  - grunt.initConfig({

    concat: { **dev: { … }, prod: { … },** }

    });

    → „grunt concat:dev“, „grunt concat:prod“
  - „grunt concat“ führt alle aus → dev & prod

Task Konfiguration (2/5)

- Options
  - Konfigurationsparameter für den Task
  - überschreibt Defaults des Tasks
  - task level & target level

```
grunt.initConfig({
  concat: {
    options: { … },
    dev: {
      options: { … }
    }
  }
});
```

Task Konfiguration (3/5)

- Files
  - Angabe per „src“ und „dest“
  - Arrays und weitere Parameter möglich
    - filter, nonull, dot, matchBase, expand, …

```
dev1: {
  src: ['src/datei1.js', 'src/datei2.js'],
  dest: 'build/combined.js'
},
dev2: { 'build/combined.js': ['src/datei1.js', 'src/datei2.js'] },
prod: { files: [
  { src: ['src/datei1.js', 'src/datei2.js'], dest: 'res/' , nonull : true }
 ] }
```

Task Konfiguration (4/5)

- Templates
  - Angabe innerhalb <% %>
  - Scope: alles innerhlab des Konfig-Objects und „grunt“

```
dev1: {
  src: ['<%= srcFiles %>', 'src2/widget2.js'],
  dest: '<%= targetFolder %>/combined.<%=
        grunt.template.today('yyyy-mm-dd') %>.js'
},
targetFolder : 'res',
srcFiles : ['src1/lib.js', 'src2/widget.js']
```

Task Konfiguration (5/5)

- Import externer Daten
  - „grunt.file.readJSON“ und „grunt.file.readYAML“

```
pkg: grunt.file.readJSON('package.json'),
dev2: {
  src: 'src/<%= pkg.name %>.js',
  dest: 'res/<%= pkg.name %>.combined.js'
}
```

Tasks (1/5)

- Aufruf über CLI („grunt <task>“) oder per default
- Definition oder Aliasing möglich
  - grunt.registerTask(taskName, [description,] taskList)
    → Liste von anderen Tasks, aufrufbar über den Alias taskName
  - grunt.registerTask(taskName, [description,] taskFunction)
    → Definition einer Funktion, aufrufba über taskName
- „Multitasks“
  - Tasks mit verschiedenen Targets
  - grunt.registerMultiTask(taskName, [description,] taskFunction)

Tasks (2/5)

- Parameter für Tasks

```
grunt.registerTask('log', 'logs stuff.', function(arg1, arg2) {
  if (arguments.length === 0) {
    grunt.log.writeln(this.name + ", no args");
  } else {
    grunt.log.writeln(this.name + ", " + arg1 + " " + arg2);
  }
});
```

→ Aufruf „grunt log:Hallo:Welt“ = „log, Hallo Welt“

Tasks (3/5)

- Aufruf anderer Tasks

```
grunt.registerTask('default', function() {
  grunt.task.run( 'otherTask', 'log' );
});
```

- Asynchrone Tasks

```
grunt.registerTask('asynchronerTask', function() {
  var done = this.async();
  setTimeout(function() { …; done(); }, 1000);
});
```

Tasks (4/5)

- Fail on Error
  - bricht auch Abarbeitung folgender Tasks ab! (ohne --force)

```
grunt.registerTask('default', function() {
  if (1 === 2) {
    grunt.log.error('Es ist ein Fehler aufgetreten.');
  }
  if (ifErrors { return false; }
  grunt.log.writeln('Alles gut!');
});
```

  - bei asynchronen Tasks
    → done(false);

Tasks (5/5)

- bedingte Ausführung
  - anderer Task muss erfolgreich ausgeführt worden sein

    grunt.task.requires('concat');

  - Konfigparameter muss gesetzt sein

    grunt.config.requires('pkg.name');

    grunt.config.requires(['pkg', 'name']);

    → Zugriff auf Konfigparamter

    grunt.config('pkg.name');

    grunt.config(['pkg', 'name']);

Live Demo

- Beispiel
  - jshint
  - concat
  - uglify
- Beispiel 2
  - Integration in Maven

Vielen Dank!

Frank Goraus

MATHEMA Software GmbH