Industrialisation Web & Javascript

zenika
ARCHITECTURE INFORMATIQUE

Plan

- Build et génération du livrable
- Gestion des dépendances
- Tests et qualimétrie
- Productivité
- Intégration continue
- Debugging et optimisation

Build
Génération du livrable

zenika
ARCHITECTURE INFORMATIQUE

Plan

- *Build et génération du livrable*
- Gestion des dépendances
- Tests et qualimétrie
- Productivité
- Intégration continue
- Debugging et optimisation

Pourquoi un build automatisé ?

- Plus rapide
- Peut être reproduit à l'identique facilement
- Soulage les développeurs de tâches rébarbatives

⇒ Gagne du temps et de l'argent

- Les outils de build sont omniscients dans le monde Java avec Ant, Maven, et Gradle
- De nombreuses plate-formes dispose de leur propre outil : make pour C et C++, MSBuild pour .net, Rake pour Ruby...
- Et Javascript ?

Pourquoi un build automatisé côté client ?

- Concaténation
- Minification
- Compilation
- Optimisation des images
- Documentation
- Tests
- Déploiement
- ...

Grunt

The Javascript Task Runner

Principes de fonctionnement

- Description des tâches dans un fichier `Gruntfile.js`
  - Pur Javascript
- Un exécutable en ligne de commande : `grunt tâches...`
- De nombreux plugins apportent des tâches pré-définies
  - Il n'y a plus qu'à configurer !
- Possibilité de définir à volonté de nouvelles tâches en Javascript
  - APIs pour manipuler les fichiers, le logging, le reporting...

Installation

- Pré-requis : NodeJS et NPM
- Installer la ligne de commande : `npm install -g grunt-cli`
  - La commande `grunt` devient alors accessible partout
- Grunt lui-même s'installe ensuite une fois par projet
  - Permet de travailler avec des versions différentes
  - `npm install grunt`
  - La commande `grunt` échoue s'il n'y pas de version de Grunt installée pour le projet

Concepts

- Une tâche (task) est une opération automatisable paramétrée abstraite
  - Exemple : minifier
- Une cible (target) est une instanciation d'une tâche sur un ensemble de fichier particulier
  - Exemple : minifier le fichier `app.js` en `app.min.js`
  - Une cible peut surcharger les paramètres de sa tâche
- Pour une tâche donnée, on peut créer autant de cibles que l'on veut

Syntaxe du Gruntfile

```
module.exports = function(grunt) {
  grunt.initConfig({
    // Configurer des tâches et des cibles
  });

  // Importer un plugin installée avec NPM
  grunt.loadNpmTasks('a-task-plugin');

  // Importer une tâche à partir d'un fichier
  grunt.loadTasks('a-task-file.js');

  // Déclarer une tâche composite
  grunt.registerTask('task', [/* une liste de tâche */]);

  // Déclarer une tâche personnalisée
  grunt.registerTask('custom', function() {

  });
};
```

Configurer une tâche

- Une tâche est un objet contenant :
  - 1 ou plusieurs cibles
  - 0 ou 1 objet `options` qui valorise les options par défaut pour toutes les cibles

```
grunt.initConfig({
  task: {
    options: {
      option1: 'valeur',
    },
    // Les cibles sont configurées ici
  },
});
```

Configurer des cibles

- Une cible est un objet contenant :
  - 0 ou 1 objet `options` qui surcharge les options de la tâche pour cette cible seulement
  - des paramètres (en général des chemins source et destination)

```
task: {
  target1: {
    options: {
      option1: 'surcharge',
    },
    parameter1: 'valeur',
  },
},
```

Exemple réel

```
grunt.initConfig({
  copy: {  // tâche
    assets: {  // cible
      src: 'assets/**',  // paramètre
      dest: 'target/',  // paramètre
    },
  },
});

grunt.loadNpmTasks('grunt-contrib-copy');
```

```
$ ls
assets    Gruntfile.js
$ grunt copy
Running "copy:assets" (copy) task
Created 1 directories, copied 1 file
Done, without errors.
$ ls target
assets
```

Plugins

- Grunt n'embarque aucune tâche prédéfinie
- Ces tâches sont disponibles sous forme de plugins
  - 25 sont maintenus par l'équipe Grunt : `grunt-contrib-*`
  - Des dizaines d'autres par la communauté
- Installation :
  - `npm install <plugin>`
  - `grunt.loadNpmTasks('<plugin>')`

Plugins Exemples

- Concaténer : `grunt-contrib-concat`
- Minifier : `grunt-contrib-uglify`
- Compiler du Sass : `grunt-contrib-sass`
- Optimiser des images : `grunt-contrib-imagemin`
- Extraire de la documentation JSDoc : `grunt-jsdoc`
- Lancer des tests Jasmine : `grunt-contrib-jasmine`
- Déployer en FTP : `grunt-ftp-deploy`
- Déployer sur Artifactory : `grunt-artifactory-artifact`
- ...

Plugins

- Charger tous les plugins peut devenir fastidieux

```
grunt.loadNpmTask('grunt-contrib-concat');
grunt.loadNpmTask('grunt-contrib-uglify');
grunt.loadNpmTask('grunt-contrib-sass');
grunt.loadNpmTask('grunt-contrib-imagemin');
grunt.loadNpmTask('grunt-contrib-jsdoc');
...
```

- Solution : utiliser le plugin `load-grunt-tasks`

```
require('load-grunt-tasks')(grunt);
```

  - Charge tous les plugins listés dans le `package.json` (voir chapitre suivant)


TP1

Gestion des dépendances

zenika
ARCHITECTURE INFORMATIQUE

Plan

- Build et génération du livrable
- *Gestion des dépendances*
- Tests et qualimétrie
- Productivité
- Intégration continue
- Debugging et optimisation

Motivation

- Facilité d'installation des dépendances
- Pas de dépendances dans le gestionnaire de sources
- Gestion des dépendances transitives
  - Si l'application dépend de `A` et que `A` dépend de `B` alors `B` est localisée et téléchargée automatiquement

NPM

The Node Package Manager

NPM Node Package Manager

- http://npmjs.org
- ~64k packages pour la plate-forme Node
  - pas pour le Javascript côté client
  - mais utile pour les outils de développement (Grunt...)
- Description des dépendances dans un fichier `package.json`
  - Différenciation des dépendances de développement (build, test) et des dépendances de production
- Stockage dans un dossier `node_modules`, un par projet

Syntaxe du package.json

```
{
  "name": "formation-zenika-industrialisation-web-et-javascript",
  "version": "1.0.0",
  "author": "Zenika",
  "devDependencies": {
    "grunt": "~0.4.4"
  },
  "dependencies": {
    "async": "0.2.x",
    "underscore": "^1.6.0"
  }
}
```

Versionning

- Les versions doivent suivre le Semantic Versioning 2.0
  - http://semver.org/
  - version = major.minor.patch-anything
  - `~` : "raisonnablement proche de" (mineur fixe)
  - `^` : "compatible avec" (majeur fixe)
  - `x` : "n'importe quel nombre à cet emplacement"
  - sans préfixe ou `=` ou `v` : exactement cette version
  - `<`, `<=`, `>`, `>=` sont également supportés

Commandes

- `npm install` peut être abbrévié `npm i`
- `npm init` crée un `package.json` interactivement
- `npm install` installe toutes les dépendances si un `package.json` est présent
- L'option `--save` de `npm install` enregistre la dépendance
  - Idem pour les dépendances de développement avec `--save-dev`
  - La version peut être spécifiée : `npm install <package>@<version>`
- Recherche : `npm search <term>`

Commandes

- L'option `--global` ou `-g` de `npm install <package>` installe la dépendance globalement. Ce n'est utile que pour les outils qui fournissent une interface en ligne de commande que l'ont veut disponible partout.
- L'option `--production` installe seulement les dépendances de production
- `npm prune` désinstalle tous les packages qui ne sont pas mentionnées dans `package.json`
- `npm shrinkwrap` gèle récursivement toutes les dépendances et écrit le résultat dans un fichier utilisable au déploiement
- `npm outdated` permet de savoir quels packages disposent d'une version plus récente que celle installée

Côté client ?

- Rappel : NPM installe des packages pour NodeJS, pas des librairies pour le développement dans le navigateur
- Il faut donc un autre programme pour cela...

Bower

A package manager for the web

Bower

- http://bower.io
- Basé sur Git, développé par Twitter
- Tourne sur NodeJS : `npm install -g bower`
- ~10k packages dans l'index
  - mais possible d'utiliser n'importe quel dépôt Git ou SVN comme source pour d'autres packages
- Description des dépendances dans un fichier `bower.json`
  - Différenciation des dépendances de développement (build, test) et des dépendances de production
- Stockage dans un dossier `bower_components`, un par projet

Syntaxe du bower.json

```
{
  "name": "formation-zenika-industrialisation-web-et-javascript",
  "version": "1.0.0",
  "devDependencies": {
    "underscore": "^1.6.0"
  },
  "dependencies": {
    "angular": "~1.2.10"
  }
}
```

- Les versions doivent suivre le Semantic Versioning 2.0
  - version = major.minor.patch-anything
- ou être une référence Git : commit, tag, branche

Commandes

- `bower init` crée un `bower.json` interactivement
- `bower install` installe toutes les dépendances si un fichier `bower.json` est présent
- L'option `--save` de `bower install <package>` enregistre la dépendance dans le `bower.json`
  - Idem pour les dépendances de développement avec `--save-dev`
  - La version peut être spécifiée : `bower install <package>#<version/commit>`
- Recherche : `bower search <term>`

Intégration Bower / Grunt

- On peut consommer les packages Bower simplement en référençant leur contenu dans une page HTML

```
<link href="bower_components/bootstrap/dist/bootstrap.css">
```

  ⇒ Contraint d'embarquer `bower_components` en production alors que tous les fichiers ne sont pas forcément utilisés

- Solution : utiliser Grunt pour copier les fichiers dans le répertoire de build
  - Plusieurs plugins de la communauté sont disponibles (fonctionnalités et maturité diverses) : `grunt-bower`, `grunt-bowercopy`, `grunt-bower-task`
  - On peut aussi le faire avec `grunt-contrib-copy`


TP2

Tests
Qualimétrie

zenika
ARCHITECTURE INFORMATIQUE

Plan

- Build et génération du livrable
- Gestion des dépendances
- *Tests et qualimétrie*
- Productivité
- Intégration continue
- Debugging et optimisation

Objectifs

- Feedback rapide sur la fiabilité du code à tous les niveaux
  - Analyse statique
  - Tests unitaires
  - Tests d'intégration
  - Tests de bout-en-bout
  - Calcul de la couverture

Analyse statique

JSHint

- http://jshint.com
  - Analyse possible directement sur le site
- Fork de JSLint de Douglas Crockford
- Recherche les erreurs
- Fait des remarques de styles
- Complètement configurable
- Installable avec NPM
  - `npm install -g jshint`
  - `jshint <fichier.js>`

Exemple

```
function une_accolade_par_là() {
  une_autre_variable = 2 * variable
  var variable = 0
  la_même = '' || [];
  if (variable == la_même) return undefined
  consle.log("que dit jshint ?")
  return 1
}

function une_accolade_par_ci()
{
  return
  {
    un: 'objet,'
    bien: 'rempli'
  }
}

console.log(une_accolade_par_ci())
console.log(une_accolade_par_là())
```

Règles

- Liste des règles : http://jshint.com/docs/options
- Toutes (dés)activables
- Configuration pour un projet à écrire dans un fichier JSON nommé `.jshintrc`
  - Il est possible de mettre un de ces fichiers dans un sous-dossier, et il surchargera les options du fichier parent

```
{
  "browser": true,
  "asi": false,
  "laxbreak": false,
  ...
}
```

Intégration avec Grunt

- Plugin `grunt-contrib-jshint`

```
jshint: {

  options: {
    curly: true, // forcer les accolades
    eqeqeq: true, // forcer ===
    /* ... */
  },

  src: ['src/**/*.js'],
  test: ['test/**/*.js'],
},
```

Intégration aux éditeurs de texte

- Plugin pour de nombreux éditeurs
  - Notepad++, Gedit
  - Sublime Text, TextMate
  - Vim, Emacs
  - Visual Studio, Eclipse
  - IntelliJ, WebStorm (intégré en standard)

Tests

Jasmine

Behavior-Driven Javascript

Jasmine

- http://jasmine.github.io/2.0

```
describe('a parrot', function() {

  var sut = parrot();
  var message = 'hello!';

  it("repeats what it's told", function () {
    sut.onTold(message);
    expect(sut.repeat()).toBe(message);
  });
});
```

- Divers matchers : `toBeEqual`, `toContain`, `toBeLessThan`, `toBeTruthy`... + matchers custom

Setup & Teardown

```
describe('a parrot', function() {

  var sut = parrot();
  var message = 'hello!';

  beforeEach(function() {
    sut.onPet();
  });

  afterEach(function() {
    sut.onFed();
  });

  it("repeats what it's told", function () {
    sut.onTold(message);
    expect(sut.repeat()).toBe(message);
  });

});
```

Lancer les tests

- Dans un navigateur
  - Ecrire une page HTML qui importe Jasmine, le code à tester, les tests
  - Ouvrir la page dans le navigateur de référence
  - Une telle page est fourni avec Jasmine, il faut simplement modifier les `script[src]`
- Dans Node, à l'aide du projet `jasmine-node` (Jasmine 1.3)
  - `npm install -g jasmine-node`
  - `jasmine-node <fichiers/dossiers de tests>`
  - Intégrable avec Grunt

Alternatives

- Mocha
  - API très proche de Jasmine
  - Conçu pour Node mais supporte les navigateurs
  - Plus flexible mais plus difficile à appréhender (pas d'API d'assert ni de mock embarquées)
- QUnit
  - API standard xUnit
  - Conçu pour les navigateurs, peut fonctionner sous Node avec à l'aide de projets tierce-partie

Automatisation des tests unitaires

Spectacular Test Runner for Javascript

Karma

- Module Node créé par l'équipe AngularJS
- Il exécute atuomatiquement les tests
  - dans plusieurs navigateurs
  - à chaque modification du code
- Indépendant du framework de test
  - Compatible Jasmine, Mocha, QUnit et autres

Installation

- `npm install -g karma-cli`
- `npm install karma` + les plugins voulus
  - `karma-jasmine`
  - `karma-firefox-launcher`
  - ...
- `karma init` crée un fichier de configuration `karma.conf.js` interactivement
- `karma start` lance Karma en continue
  - ajouter l'option `--single-run` pour passer les tests une fois

Exemple de karma.conf.js

```
module.exports = function(config) {
  config.set({
    frameworks: ['jasmine'],

    files: [ // Inclus le code à tester
      'src/*.js',
      'test/*.js',
    ],

    browsers: ['Chrome', 'Firefox'],

    // Relancer les tests à chaque modification d'un fichier
    autoWatch: true,

    // Une seule passe de test
    singleRun: false,
  });
};
```

Couverture de test

- `npm install karma-coverage`

```
module.exports = function(config) {
  config.set({

    /* ... */

    preprocessors: {
      "*.js": ['coverage'],
    },

    reporters: ['coverage'],

  });
};
```

Couverture de test

Code coverage report for **./simple-spec.js**

Statements: **100%** (14 / 14) Branches: **100%** (0 / 0) Functions: **100%** (6 / 6) Lines: **100%** (14 / 14) Ignored: none

All files » .\ » simple-spec.js

```
 1   1 function parrot() {
 2   1   var latestMessage = "";
 3   1   return {
 4       onTold: function(message) {
 5   1     latestMessage = message;
 6       },
 7       repeat: function() {
 8   1     return latestMessage;
 9       }
10     }
11   }
12
13
14   1 describe('a parrot', function() {
15
16   1   var sut = parrot();
17   1   var message = 'hello!';
18
19   1   beforeEach(function() {
20   1     spyOn(sut, 'onTold').and.callThrough();
21     });
22
23   1   it("repeats what it's told", function () {
24   1     sut.onTold(message);
25   1     expect(sut.onTold).toHaveBeenCalledWith(message);
26   1     expect(sut.repeat()).toBe(message);
27     });
28
29   });
```

Generated by istanbul at Thu Mar 20 2014 11:10:59 GMT+0100 (Paris, Madrid)

Tests bout-en-bout

Protractor: E2E test framework for Angular apps

Protractor

- Module Node créé par l'équipe AngularJS
- Basé sur Selenium
  - Tests par automation du navigateur
  - Nécessite un serveur Selenium
  - Reprend le style de l'API Selenium en ajoutant des spécificités Angular
- API pour les tests : Jasmine ou Mocha

Mise en route

- `npm install -g protractor`
- `webdriver-manager update` + `webdriver-manager start` pour installer et lancer un serveur Selenium
- `protractor protractor.conf.js`

```
exports.config = {
  seleniumAddress: 'http://localhost:4444/wd/hub',
  capabilities: {
    'browserName': 'firefox'
  },
  specs: ['spec.js'],
};
```

Exemple de test

```
describe('angularjs homepage', function() {

  it('should greet the named user', function() {
    browser.get('http://www.angularjs.org');

    // Cherche les input avec ng-model=yourName
    element(by.model('yourName')).sendKeys('Zenika');

    // Cherche les éléments bindés à yourName
    // Exemple : <h1>Hello \{{yourName}}</h1>
    var greeting = element(by.binding('yourName'));

    expect(greeting.getText()).toEqual('Hello Zenika!');
  });

});
```


TP3

Productivité

zenika

ARCHITECTURE INFORMATIQUE

Plan

- Build et génération du livrable
- Gestion des dépendances
- Tests et qualimétrie
- *Productivité*
- Intégration continue
- Debugging et optimisation

LiveReload

A happy land where browsers don't need a Refresh button.

Fonctionnement

- Un serveur surveille des fichiers et envoie des notifications à tous les clients connectés quand l'un d'eux est modifié.
- Le serveur peut être :
  - Une application avec GUI : http://livereload.com/
  - Une application en ligne de commande : `guard-livereload`
  - Un éditeur de texte
  - Un outil de build : `grunt-contrib-watch`
- Le client prend la forme d'un plugin pour navigateur qui rafraichit la page à chaque notification reçue

Configuration de Grunt

- `npm install grunt-contrib-watch`

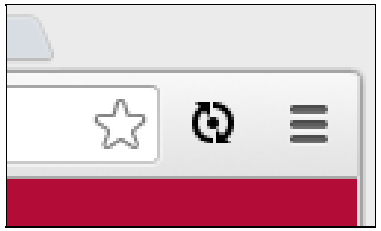
```
watch: {
  options: {
    livereload: true,
  },
  assets: {
    files: ['assets/**'],
    tasks: ['copy:assets'],
  },
}
```

```
grunt watch
```

- Attention : `watch` est une tâche bloquante, elle doit donc être la dernière tâche dans une suite de tâche

Côté navigateur

- Extensions Firefox et Safari disponibles sur livereload.com
  - Pas d'accès aux fichiers locaux pour Safari
- Extension Chrome LiveReload sut le Chrome Web Store
  - Cocher Autoriser l'accès aux URL de fichier dans les paramètres de l'extension pour pouvoir l'utiliser avec des fichiers locaux
  - Il faut activer l'extension en cliquant dessus pour chaque page que l'on veut voir rechargée



Les méta-langages

- Méta-langages Javascript
  - CoffeeScript
  - TypeScript
  - Dart
- Méta-langages CSS
  - Sass
  - Less

CoffeeScript

- http://coffeescript.org/
- Amélioration syntaxique du Javascript
- Accès à tout l'écosystème JS
- Se compile en du Javascript lisible

CoffeeScript

- `var` ainsi que certaines parenthèses, accolades et virgules sont optionnels

```
newObj = Object.create 'CoffeeScript'
```

```
computer =
    vendor: 'Zenika'
    cpu: 3200
    ram: 8192
```

- Tout est expression

```
hour24 = if ampm is 'AM' then hour12 else hour12 + 12
```

CoffeeScript

- Opérateur d'existence

```
if WebSockets?
```

- Notation raccourci pour les fonctions

```
square = (x) -> x * x
```

- Interpolation (double quotes seulement)

```
fruits = 'fraises'
print "les #{fruits} sont de saison !"
```

CoffeeScript

- Classes et héritage

```
class Animal
  constructor: (@name) ->

  move: (meters) ->
    alert @name + " moved #{meters}m."

class Snake extends Animal
  move: ->
    alert "Slithering..."
    super 5
```

CoffeeScript

- List comprehensions et intervalles

```
x * x for x in [0..10]
```

- Beaucoup d'autres facilités de syntaxe
  - `yes`/`no`, `on`/`off` équivalents à `true`/`false`
  - `and`, `or` et `not` au lieu de `&&`, `||` et `!`
  - `unless` plutôt que `if not`
  - `==` et `!=` sont traduit en `===` et `!==`
  - Possible d'intégrer du Javascript entre backticks

TypeScript

- http://www.typescriptlang.org
- Un Javascript avec typage optionnel
- Sur-ensemble de Javascript
  - Tout code Javscript est un code TypeScript

TypeScript

- Typage des paramètres

```
function greeter(person: string) {
    return "Hello, " + person;
}
```

- Interface

```
interface Person {
    firstname: string;
    lastname: string;
}
```

TypeScript

- Classes

```
class Greeter {
  greeting: string;
  constructor(message: string) {
    this.greeting = message;
  }
  greet() {
    return "Hello, " + this.greeting;
  }
}

var greeter = new Greeter("world");
```

Sass

- http://sass-lang.com
- Syntactically Awesome Style Sheets

Sass

- Variales

```
$font-stack:    Helvetica, sans-serif
$primary-color: #333

body
  font: 100% $font-stack
  color: $primary-color
```

Sass

- Mixins

```
@mixin border-radius($radius) {
  -webkit-border-radius: $radius;
     -moz-border-radius: $radius;
      -ms-border-radius: $radius;
          border-radius: $radius;
}

.box { @include border-radius(10px); }
```

Sass

- Nesting

```
nav
  ul
    margin: 0
    padding: 0
    list-style: none

  li
    display: inline-block

  a
    display: block
    padding: 6px 12px
    text-decoration: none
```

Sass

- Héritage

```
.message
  border: 1px solid #ccc
  padding: 10px
  color: #333

.success
  @extend .message
  border-color: green
```


TP4