

Documentation des API Rest (docs, tests, mocks)

Différents outils de génération de doc

Classement par préférence :

1. [RAML](#) : v1.0 (2015-11-03) [Licence MIT]
2. [API-Blueprint](#) : v1A9 (2015-06-08) [Licence MIT]
3. [Swagger](#) : v2.0 (2014-09-08) [Licence Apache, v2.0]
4. [ApiDocJs](#) : v0.13.0 (2015-05-08) Permet de générer une doc à partir d'une syntaxe PHPdoc. Simple mais avec peu d'outils liés. [Licence MIT]

RAML

Site principal : [RAML](#) : v1.0 (2015-11-03) [Licence MIT]

Permet de rédiger la doc en Markdown. Très similaire à Apiblueprint. Écosystème très riche (plus qu'Apiblueprint). De très nombreux outils (dans différents langage) à disposition : doc, test, designer, mock server...

Apiblueprint

Site principal : [API-Blueprint](#) : v1A9 (2015-06-08) [Licence MIT]

Permet de rédiger la doc en Markdown. Relativement complexe pour se retrouver dans la doc.

Ressources

- [Site officiel](#)
 - [Exemple](#) : exemple de fichier markdown.
 - [Agglio](#) : interface de consultation de la doc API Blueprint (PHP). [Licence MIT]
 - [Rendu](#) : Exemple de rendu avec Aglio.
 - [Dredd](#) : outils vérifiant la synchro entre l'implémentation des services web et la doc API Blueprint. [Licence MIT]
 - [Retour sur les API REST et les outils de tests](#), dont Dredd & API Blueprint
 - [Cucumber-4-API-blueprint](#) : jonction entre Cucumber (framework de test JS) et Api-blueprint. [Licence MIT]
 - [Api-Mock](#) : créer un serveur qui renvoie les données fournies en exemple pour chaque service web. [Licence MIT]
 - [Apib2Swagger](#) : convertisseur de API Blueprint 1A8 vers Swagger 2.0. [Licence MIT]
- Pour les tutoriels, voir le site Apiary :
 - [Tutoriel](#) : tutoriel pour la création d'une doc API Blueprint.
 - [API Blueprint](#)
 - [MSON](#)

- [URI Templates](#)
- [JSON Schema](#)
- Les spécifications :
 - [API Blueprint](#)
 - [MSON](#)

Notes

- Pour pouvoir gérer différente version de *nodejs* (et *npm*), il faut utiliser : ***nvm***
- Installer *npm* globalement avec *nvm* : `## n=$(which node);n=${n%/bin/node}; chmod -R 755 $n/bin/*; sudo cp -r $n/{bin,lib,share} /usr/local ##`
- Pour que *Protagonist* fonctionne, il faut bien compiler tous les paquets avec la même version de *Node*.
- Comma separated URI parameter : <https://github.com/apiaryio/api-blueprint/issues/120>

Swagger

Site principal : [Swagger](#) : v2.0 (2014-09-08) [Licence Apache, v2.0] Ancien, mais toujours actif. Syntaxe YAML. Editeur et doc utilise *nodejs*. L'écosystème semble moins développer que les précédents. Le générateur de SDK est en Java...

Ressources

- [Swagger-Editor](#) : interface web (utilise Node.js)
- [Swagger-ui](#) : interface de consultation de la doc Swagger (HTML 5). [Licence Apache, v2.0]

Initialisation d'un projet de doc avec Apiblueprint

1. Installer le gestionnaire de version de node : `## nvm ##`
2. Installer la version 0.12.8 : `##nvm install 0.12.8##`
3. Basculer sur la version de node 0.12.8 : `##nvm use 0.12.8##`
4. Avec *npm* initialiser un projet de doc : `##npm init##`
5. Puis installer les paquets :
 - *aglio* : pour générer la doc au format HTML.
 - *drakov* : serveur qui 'mocke' une api en fonction de la doc APIblueprint.
 - *dredd* : pour tester la doc vis à vis de l'api développée. Permet de vérifier la consistance de la doc vis à vis de l'API réelle.
 - *hercule* : permet de réaliser la transclusion de contenu présent dans différents fichiers.

Aglio (node : v0.12.8)

Pour simplifier, on peut rajouter ces commandes dans la partie *scripts* du fichier *package.json*, avec les entrées :

```
"aglio-server": "./node_modules/.bin/aglio -i mon-api.apib --theme-full-width --theme-variables flatly -s",  
"aglio-convert": "./node_modules/.bin/aglio -i mon-api.apib --theme-full-width --theme-variables flatly -o mon-api.html"
```

Drakov (node : v0.12.8)

Serveur de mock de l'API. Ajouter dans la partie *scripts* du fichier *package.json* :

```
"drakov": "./node_modules/.bin/drakov -f mon-api.apib"
```

Dredd (node : v0.12.8)

- Pour initialiser *Dredd* et créer le fichier de config *dredd.yml*, lancer la commande :
##./node_modules/.bin/dredd init##
- Pour lancer les tests avec Dredd : ##./node_modules/.bin/dredd##
- Pour simplifier, on peut rajouter ces commandes dans la partie *scripts* de *package.json* :

```
"dredd": "./node_modules/.bin/dredd"
```

On pourra ensuite utiliser les commandes :

- ## npm run dredd init ##
- ## npm run dredd ##

Hercule (node v0.12.8)

Permet de découper une document Markdown en plusieurs sous partie.

- Pour simplifier, on peut rajouter une commande dans la partie *scripts* de *package.json* :

```
"hercule": "./node_modules/.bin/hercule mon-api.src.apib -o mon-api.apib"
```

Gulp (node v0.12.8)

Permet d'automatiser le fonctionnement des modules ci-dessus : aglio, dredd et hercule.

```
var gulp = require('gulp');  
var plumber = require('gulp-plumber');  
var rename = require("gulp-rename");  
var es = require('event-stream');  
var hercule = require('hercule');  
var dredd = require('dredd');  
var exec = require('child_process').exec;  
  
gulp.task('default', ['startWatcher', 'startAglioServer']);
```

```
gulp.task('startAgliaServer', function() {
  exec('./node_modules/.bin/aglio -i mon-api.apib --theme-full-width --
theme-variables flatly -s', function (error, stdout, stderr) {
    console.log('Stdout: ' + stdout);
    console.log('Stderr: ' + stderr);
    if (error !== null) {
      console.log('Exec error: ' + error);
    }
  });
});

gulp.task('startWatcher', function() {
  var watcher = gulp.watch('mon-api.src.apib', ['doHercule']);
  var watcherDest = gulp.watch('mon-api.apib', ['doDredd']);

  watcher.on('change', function(event) {
    console.log('File ' + event.path + ' was ' + event.type + ', running
tasks...');
  });
});

gulp.task('doHercule', function() {
  function transclude() {
    // you're going to receive Vinyl files as chunks
    function transform(file, cb) {
      hercule.transcludeString(file.contents.toString(), function(output) {
        file.contents = new Buffer(output);
      });
      cb(null, file);
    }
    return es.map(transform);
  };

  gulp.src('mon-api.src.apib')
    .pipe(plumber())
    .pipe(transclude())
    .pipe(rename('mon-api.apib'))
    .pipe(gulp.dest('./'));
});

gulp.task('doDredd', function() {
  var dreddConfig = {
    server: 'http://api.mon-site.org/mon-api/v1', // your URL to API
    endpoint the tests will run against
    options: {
      'path': 'mon-api.apib', // Required Array if Strings; filepaths
to API Blueprint files, can use glob wildcards
      'dry-run': false, // Boolean, do not run any real HTTP transaction
      'names': false, // Boolean, Print Transaction names and finish,
```

```
similar to dry-run
  'level': 'verbose', // String, log-level (info, silly, debug, verbose,
  ...)
  'silent': false, // Boolean, Silences all logging output
  'only': [], // Array of Strings, run only transaction that match
these names
  'header': [], // Array of Strings, these strings are then added as
headers (key:value) to every transaction
  'user': null, // String, Basic Auth credentials in the form
username:password
  'hookfiles': [], // Array of Strings, filepaths to files containing
hooks (can use glob wildcards)
  //'reporter': ['dot', 'html'], // Array of possible reporters, see
folder src/reporters
  //'output': [], // Array of Strings, filepaths to files used for
output of file-based reporters
  'inline-errors': false, // Boolean, If failures/errors are display
immediately in Dredd run
  'color': true,
  'timestamp': false
}
  //'emitter': EventEmitterInstance, // optional - listen to test
progress, your own instance of EventEmitter
  //'hooksData': {
  // 'pathToHook' : ''
  //}
};
var dredd = new dredd(dreddConfig);
dredd.run(function (err, stats) {
  if (err) {
    console.log(err);
  }
});
});
```

From:

<https://memos.clapas.org/> - Memos

Permanent link:

<https://memos.clapas.org/informatique/developpement/api-rest/doc-test-mock?rev=1576335346>

Last update: 2019/12/14 14:55

