

Table des matières

<Base>	2
Qu'est-ce que SailsJS ?	2
Que permet-il ?	2
Installation	3
Créer un projet:	3
Structure par défaut du nouveau projet	4
Générer une API	4
Particularité du moteur de template EJS	4

<Base>

Qu'est-ce que SailsJS ?

Sails est, bien sûr, un framework web. Mais prenez un peu de recul. Qu'est-ce que cela veut dire? Parfois, quand nous nous référons à la «toile», nous entendons le «front-end Web."

Nous pensons à des concepts tels que les standards du Web, ou l'HTML 5, le CSS3; et les frameworks comme Backbone, ou angularJS, ou jQuery.

Sails n'est pas "ce genre" de framework web.

Sails fonctionne très bien avec angularJS et Backbone, mais vous n'utiliserez jamais sailsJS à la place de ces bibliothèques.

D'autre part, parfois, lorsque nous parlons de «frameworks», nous entendons le «back-end Web." Cela évoque des concepts tels que REST, ou HTTP, ou WebSockets; et les technologies comme Java, ou Ruby, ou Node.js.

Un cadre "web back-end" vous aide à faire des choses comme construire une API, servir des fichiers HTML, et de gérer des centaines de milliers d'utilisateurs simultanés.

Sails est "ce genre" d'framework web.

Que permet-il ?

Le framework **Sails.js** permet de créer rapidement une application Web avec une architecture MVC reprenant le base Ruby On Rails pour l'adapter à un serveur NodeJS.

Les points mise en avant dans Sails.js :

- Un ORM simple pouvant être utiliser avec n'importe quelle base de données, il suffit d'employer l'adaptateur qui correspond à la base de données ciblée
- L'API RESTful JSON est automatiquement générée par Sails.js
- Un système de sécurité comprenant authentification, rôles et stratégies de permissions flexibles, personnalisables et utilisable dans les cas d'applications temps réel
- Les requêtes Socket.IO sont directement mappées vers les contrôleurs associés de façon transparente
- Optimisation automatique des CSS et Javascript lors de la mise en production via la minification des ressources

Pour ce faire Sails.js se base sur :

- ExpressJS : framework de création d'applications web en NodeJs
- Blueprint : Il permet de générer automatiquement des Api Json des appels CRUD du modèle, ainsi qu'un routing automatique des contrôleurs.
- SocketIO : Module NodeJs pour les communications via les websockets
- Waterline : ORM utilisé par Sails.js
- EJS : Librairie de templating Javascript utilisé par Sails.js pour la création des vues
- Grunt : Pour l'exécution de tâches automatiques javascript

Installation

To install the latest stable release with the command-line tool:

```
sudo npm -g install sails
```

Créer un projet:

Create a new app:

```
sails new testProject
```

Now lift the server:

```
cd testProject
```

```
sails lift
```

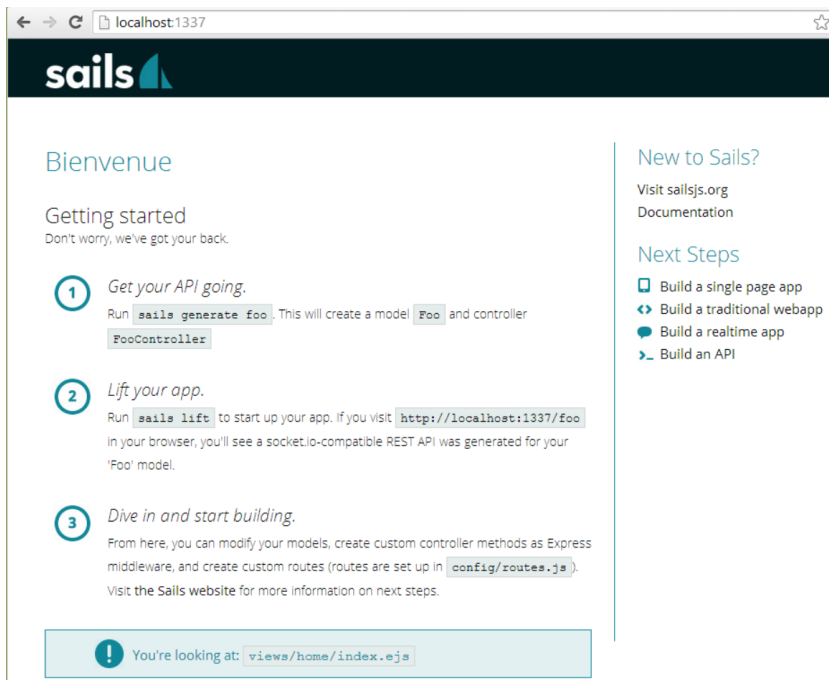
Ceci démarre un serveur NodeJs avec votre application Sails.js.

```











info: Sails.js
info: v0.9.16
info:
info:      /\_/\
info:     /__\/__\
info:    /___\/___\
info:   /____\/____\
info:  /_____\/_____
info: /_____/_____/_____/
info:/_____/_____/_____/_____/
info:/_____/_____/_____/_____/_____/
info:/_____/_____/_____/_____/_____/_____/
info:Server lifted in `D:\NETAPSYS\TB\SAILS\TEST`
info: To see your app, visit http://localhost:1337
info: To shut down Sails, press <CTRL> + C at any time.
debug: -----
debug: :: Wed Jun 25 2014 11:11:57 GMT+0200 (Paris, Madrid (heure d'été))
debug: Environment : development
debug: Port : 1337
debug: -----
info: handshake authorized dUgD8Eb7T0DIBbRBS70C

```

At this point, if you visit (<http://localhost:1337/>) you will see the default home page.



Structure par défaut du nouveau projet

 api	api : toutes les couches de votre logique métier (le modèle, les contrôleurs, les services ...)
 assets	assets : le dossier pour les fichiers JS et les CSS
 config	config : contient tous les fichiers de configuration liés à l'application (les adaptateurs pour les
 node_modules	bases de données, les configurations globales pour : les rôles utilisateurs, les contrôleurs, les vues,...)
 views	node_modules : modules NodeJs utilisés au sein du projet
 .gitignore	views : templates Javascript au format EJS
 app.js	app.js : fichier exécuter au lancement de votre application
 Gruntfile.js	Gruntfile.js : définitions des tâches Grunt par défaut (minification, injection automatique des CSS et JS dans les vues...)
 package.json	package.json : déclare les dépendances Node de l'application, les informations propres à l'application (version, description, fichier principal...)
 README.md	

Générer une API

Sails.js a prévu une commande afin de générer une nouvelle API. Retournez donc sur votre console et tapez :

```
sails generate api article
```

Un modèle est un objet représentant une entité de notre base de données.

Cet objet est issu de l'ORM de Sails.js : Waterline.

Waterline va nous permettre de manipuler nos données via des Query Methods. Ce sont des méthodes déjà prêtes qui vont nous permettre d'ajouter des données dans notre base, d'en modifier, d'en récupérer, etc. Nous verrons tout cela dans nos controllers.

Enfin, Waterline utilise par défaut une base de données local (.db).

Particularité du moteur de template EJS

Voici la particularité du moteur de template EJS : Les balises pour écrire du Javascript.

<% : Permet d'écrire du Javascript.

<%= : Permet d'afficher le contenu d'une variable Javascript

<%- : Permet d'afficher le contenu d'une variable Javascript sans encodage. Utile pour interpréter du HTML dans le DOM.