

Useful Rreferances

Official Documentation

<http://sailsjs.com/>

<https://digitaldrummerj.github.io/sails-tutorial/>

Beginers guide

<https://www.codementor.io/@codeforgeek/how-to-setup-sailsjs-tutorial-beginners-du107nl5i#installation-and-configuration>

<https://www.sitepoint.com/an-introduction-to-sails-js/>

File image upload

<https://jumpstartsails.blogspot.com/2015/10/fileimage-uploading-in-sailsjs.html>

How to install sails on managed hosting

<https://www.a2hosting.in/kb/installable-applications/manual-installations/installing-sails-js-on-managed-hosting-accounts#Step-2.3A-Install-Sails.js>

knex js as node js (Data migration tool -web Development)

<https://www.logisticinfotech.com/blog/use-of-knex-js-as-node-js-database-migration-tool/>

send email

https://medium.com/@raj_adroit/sails-js-email-sending-using-sails-hook-email-and-mailgun-service-f6a8ab0e6d77

for sending mail

```
"nodemailer": "^4.7.0",  
"nodemailer-ses-transport": "^1.5.1",  
"nodemailer-smtp-transport": "^2.7.4",
```

job and queue

<https://www.logisticinfotech.com/blog/easiest-way-to-create-job-queue-in-sails-with-sails-hook-job-queue/>

<https://www.npmjs.com/package/kue#queue-maintenance>

open port check

<https://askubuntu.com/questions/410218/how-to-close-an-open-port-in-ubuntu>

kue npm for jobs

<https://www.npmjs.com/package/kue#queue-maintenance>

```
"kue": "^0.11.6",
```

kill specific port

sudo kill \$(sudo lsof -t -i:1337)

show all open port list

sudo netstat -lnp

<https://linuxize.com/post/check-listening-ports-linux/>

show all tcp open port list

sudo netstat -tnlp

check specific port is open or not

sudo netstat -tnlp | grep :1337

kill specific port

sudo kill \$(sudo lsof -t -i:1337)

Installation and Configuration

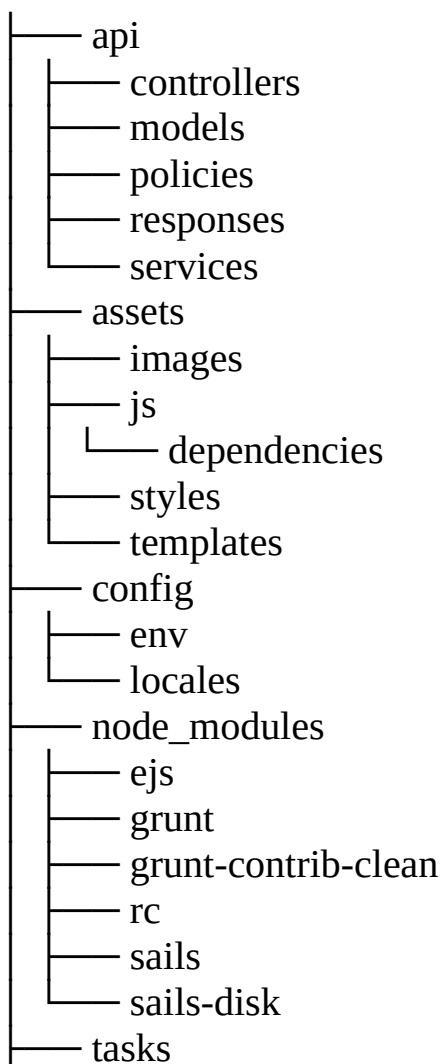
npm install -g sails

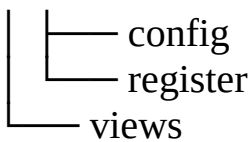
sails create projectName

npm install

run the project - sails lift

Visit localhost:1337 to view the app.





- **api/controllers**: this is the folder where controllers live. Controllers correspond to the *C* part in *MVC*. It's where the business logic for your application exists.
- **api/models**: the folder where models exist. Models correspond to the *M* part of *MVC* architecture. This is where you need to put classes or objects that map to your SQL/NoSQL data.
- **api/policies**: this is the folder where you need to put policies for your application
- **api/responses**: this folder contains server response logic such as functions to handle the 404 and 500 responses, etc.
- **api/services**: this where your app-wide services live. A service is a global class encapsulating common logic that can be used throughout many controllers.
- **./views**: this folder contains templates used for displaying views. By default, this folder contains the *ejs* engine templates, but you can configure any Express-supported engine such as *EJS*, *Jade*, *Handlebars*, *Mustache* and *Underscore* etc.
- **./config**: this folder contains many configuration files that enable you to configure every detail of your application, such as *CORS*, *CSRF* protection, *i18n*, *http*, settings for models, views, logging and policies etc. One important file that you'll frequently use is **config/routes.js**, where you can create your application routes and map them to actual actions in the controllers or to views directly.
- **./assets**: this is the folder where you can place any static files (*CSS*, *JavaScript* and images etc.) for your application.

Configuring Your Database

`npm install --save sails-mysql`

config/connections.js changes related DB access

```
someMysqlServer: {  
  adapter: 'sails-mysql',  
  host: 'YOUR_MYSQL_SERVER_HOSTNAME_OR_IP_ADDRESS',  
  user: 'YOUR_MYSQL_USER',  
  password: 'YOUR_MYSQL_PASSWORD',  
  database: 'YOUR_MYSQL_DB'
```

```
}
```

Now to tell Sails to use it, open `config/models.js` file and change the driver.
`connection: 'someMySQLServer',`
`migrate : 'alter'`

Creating an Auto-generated API

`sails generate api testAPI`. ———> api/controllers folder,
it will create a file called `TestAPIController.js`,
———> In the Model folder, it will create new file called `testAPI.js`.

Run this command :: `sails lift`

//create notification

```
notificationObj = {  
  sender_id: inputs.fromUserId,  
  reciever_id: adminId,  
  notification_type: 'CreatedChatByUser',  
  notification_data: chatsResult.id,  
  sender_type: 'app',  
  receiver_type: 'admin'  
};
```

```
notificationResult = await Notifications.create(notificationObj).fetch();
```

1. `sails new projectname`
create empty application press 2
2. `run sails project`
`sails lift`
3. `create api`
`sails new generate apiname`
so it will create 2 files in controller and modal folder
4. now test our blueprint api (go to browser and type url)
`localhost:1337/apiname`
ite will show blank array `[]`
5. lets add items in it using blueprint api
type”
`localhost:1337/apiname/create?name =abc`

6. go back
localhost:1337/apiName
it will show one record

this data is stored in localdiskDB (tmp/archive.db and tmp/apiName.db)

remove node modules

```
rm -rf node_modules/
```

You can scaffold a new Sails.js project without a front end with this:

```
sails new sailsdemo --no-frontend
```

Configuring Your Database

```
npm install --save sails-mysql
```

install redis servdr

```
brew install redis
```

#follow this link for knex update table

<http://perkframework.com/v1/guides/database-migrations-knex.html>

MIGRATION

```
npm install knex -g  
knex migrate:make migration_file_nameMigration  
knex migrate:latest
```

```
knex migrate:latest --env staging
```

All genereate cmd

action:: sails generate action <action file name>

helper:sails genereate helper <helpername>

create controller in **api/controllers/<namecontroller.js>**

```
sails generate controller product
```

create model in **api/model/<modalname.js>**

```
sails generate model <modalname>
```

Creating an Auto-generated API

sails generate api testAPI. —> api/controllers folder,
it will create a file called TestAPIController.js,

——> In the Model folder, it will create new file called testAPI.js.

Create service and job

manually create folder in api/service or api/jobs

for sending mail npm package

```
"nodemailer": "^4.7.0",  
"nodemailer-ses-transport": "^1.5.1",  
"nodemailer-smtp-transport": "^2.7.4",
```

for making job npm package

```
"kue": "^0.11.6",
```

If got error related to ruby install

```
sudo apt install ruby-full rubygems autogen autoconf libtool make  
sudo gem install sass
```

for grunt error

```
sudo gem install sass
```

if still issue try

```
sudo gem install clean
```

```
npm install grunt-sails-linker --save-dev --save-exact
```

```
grunt buildProd
```

```
grunt buildProd --build=prod --force
```

for install grunt

```
sudo npm install -g grunt
```

```
#for install sass
```

```
sudo npm install -g sass
```

```
sudo npm install -g node-sass
```

```
#for permission
```

```
sudo chown -R $(whoami) $(npm config get  
prefix)/{lib/node_modules,bin,share}
```

```
sudo chown -R $USER:$GROUP ~/.npm
```

```
sudo chown -R $USER:$GROUP ~/.config
```

#remove cache for npm

```
sudo npm cache clean -f
```

#remove node modules

```
rm -rf /usr/local/lib/node_modules
```

```
sudo rm -rf /usr/local/lib/node_modules
```

#follow this link for knex update table

<http://perkframework.com/v1/guides/database-migrations-knex.html>