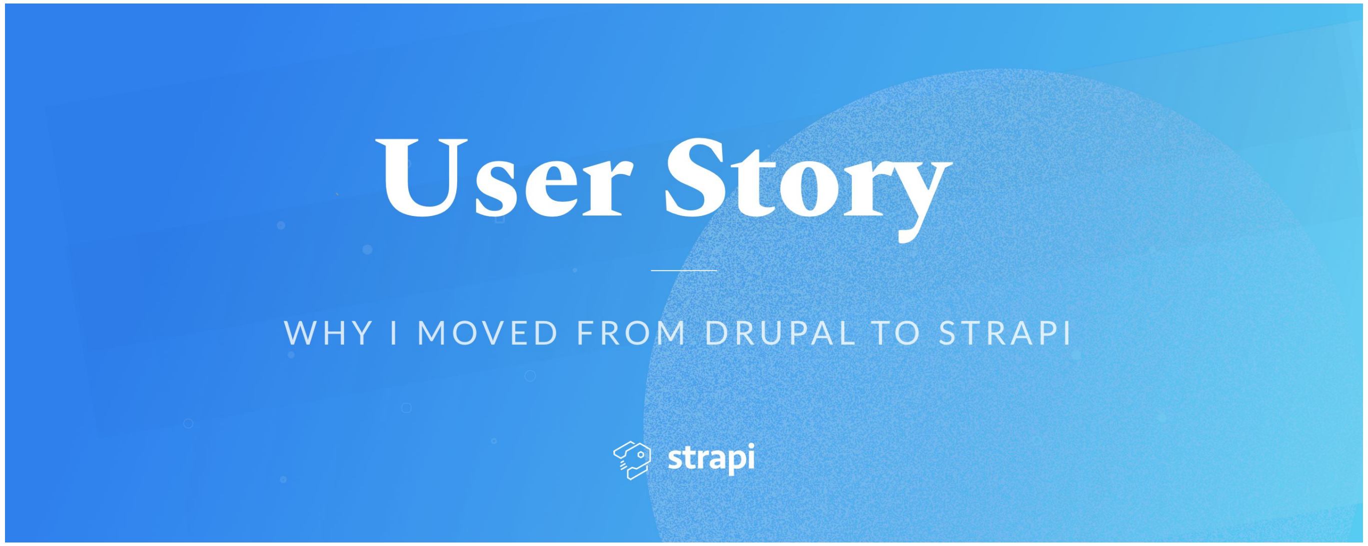
Why I moved from Drupal to Strapi - A user story

This is the story from a Strapi user that used to work on Drupal CMS and now uses Strapi. He's explaining the features available in Strapi.

Yves Do
July 30, 2019

> *Strapi's note: this is a guest post from Baldeep Singh Kwatra, a great Strapi user in Delhi, willing to share his experience on Strapi. He's been using the project for more than a year now.*



The Journey to Strapi.

Gone are the days when server-side rendering of web pages was heavily used and servers were solely responsible to generate the entire web page for the clients. Remember JSP, PHP, asp? Now we have to deal with the rise of new gadgets, the advancement of hardware and technology incorporation. New clients such as smartphones, smartwatches, smartTV, IoT's have bombarded the market, and heavily consume data.

For sure you can expose data (API) using traditional CMS but it could become cumbersome and painful. I am not saying you can't do that. It's just not thought out to do that from the ground up. Currently, there is a heavy demand to decouple the data/logic and presentation layer whereby the data is exposed as JSON response. The client, whether a browser or an app, should be able to render it. This enables to work faster. Front end and back end teams can work individually without much interference.

So what we need is an API driven architecture with the power to manage the content as traditional CMS do. Here comes the new catchy buzz word 'Headless CMS' - A CMS without its head, but surely not without brains 😉 It is a CMS that is front-end agnostic. A CMS that has no view layer. It should just be capable to store and expose the data.

And this is **the real need of the hour** for almost every project. If this is what you need then say goodbye to your traditional CMS and welcome to the era of Strapi!

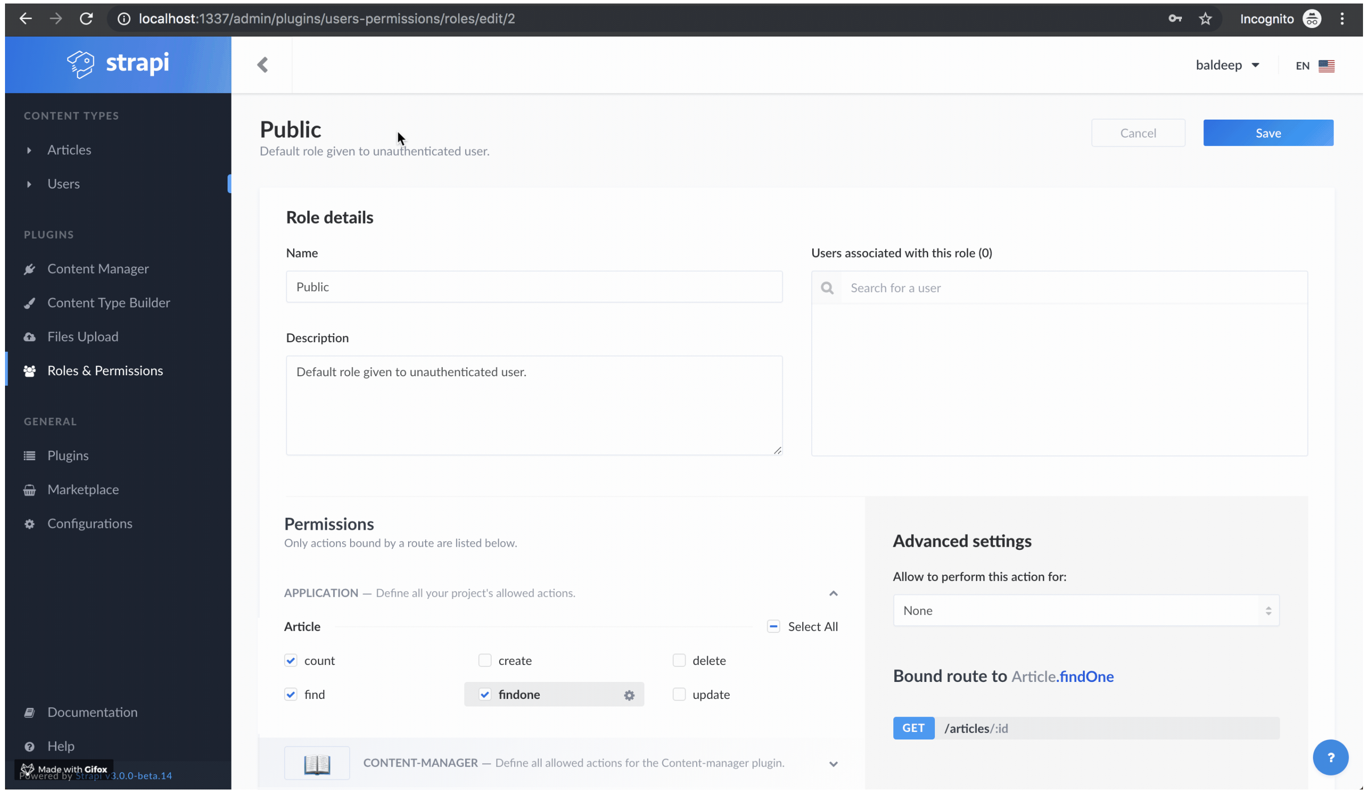
> By the way, did you know that the name Strapi comes from "Boot**strap** your **API**"?

You might think that exposing an API can be done easily using any other stack like Spring Boot or Ruby On Rails, etc. I also have worked on them. But since I started working on Strapi, I observed **time to market has heavily reduced**. All the boilerplate code is reduced and I just focus on the core business logic.

Do you know why? Because **all the items listed below are supported out of the box.**

1.
It supports all the basic CRUD operations.
You don't have to write even a single line of code. All you need is a content-type, i.e.

how your model looks like. In simpler words what you want to save in your database.



localhost:1337/admin/plugins/users-permissions/roles/edit/2

Public
Default role given to unauthenticated user.

Role details

Name: Public

Description: Default role given to unauthenticated user.

Users associated with this role (0)
Search for a user

Permissions
Only actions bound by a route are listed below.

APPLICATION — Define all your project's allowed actions.

Article
- [✓] count
- [ ] create
- [ ] delete
- [✓] find
- [✓] findone
- [ ] update

CONTENT-MANAGER — Define all allowed actions for the Content-manager plugin.

Advanced settings
Allow to perform this action for: None

Bound route to Article.findOne

GET /articles/:id

2.

Every API needs to have **server-side pagination**, which is by default supported by Strapi.

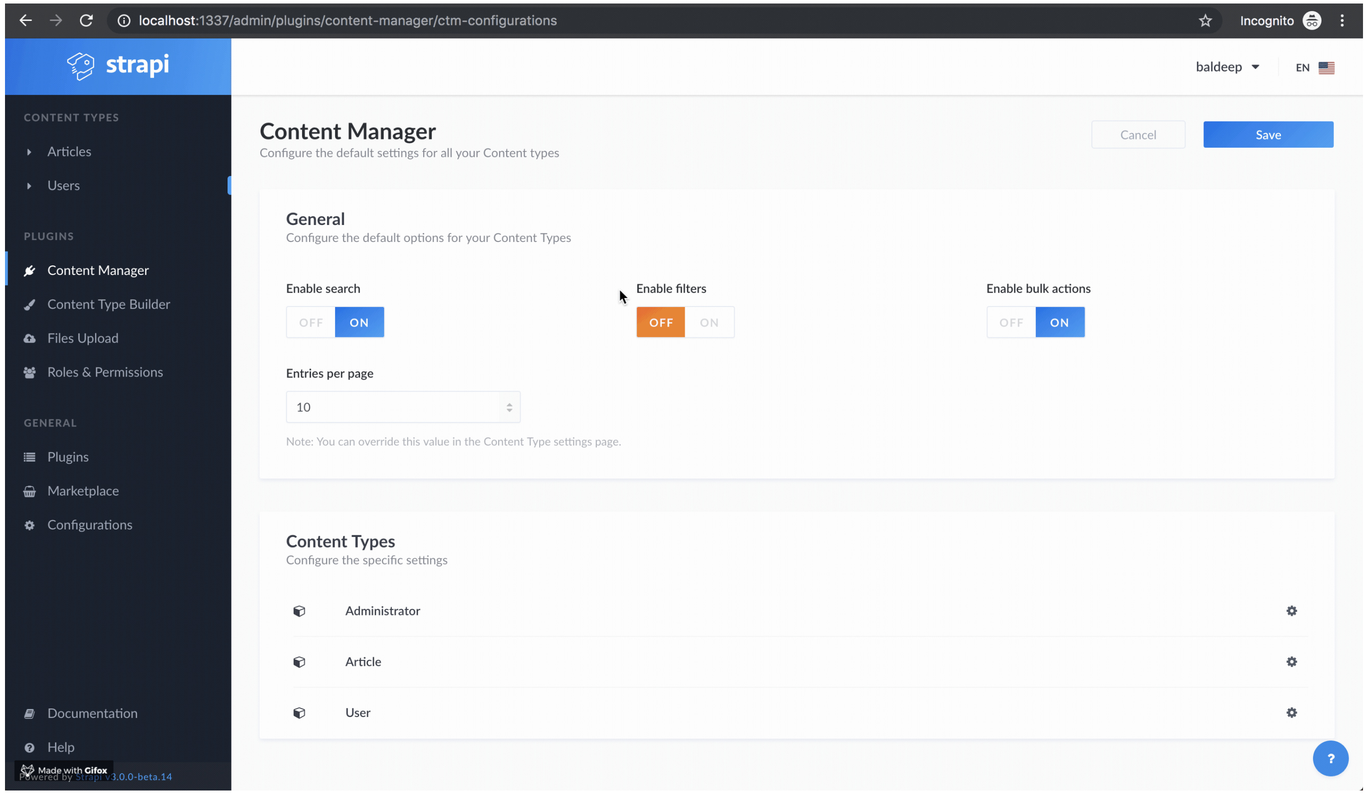
3.

Full-Text Searching.

You need to search anywhere in your database tables/collections.

4.

Filtering capabilities with conditional logics without even writing a single database query.



Content Manager
Configure the default settings for all your Content types

General
Configure the default options for your Content Types

Enable search: OFF [✓ ON]

Enable filters: [✓ OFF] ON

Enable bulk actions: OFF [✓ ON]

Entries per page: 10

Note: You can override this value in the Content Type settings page.

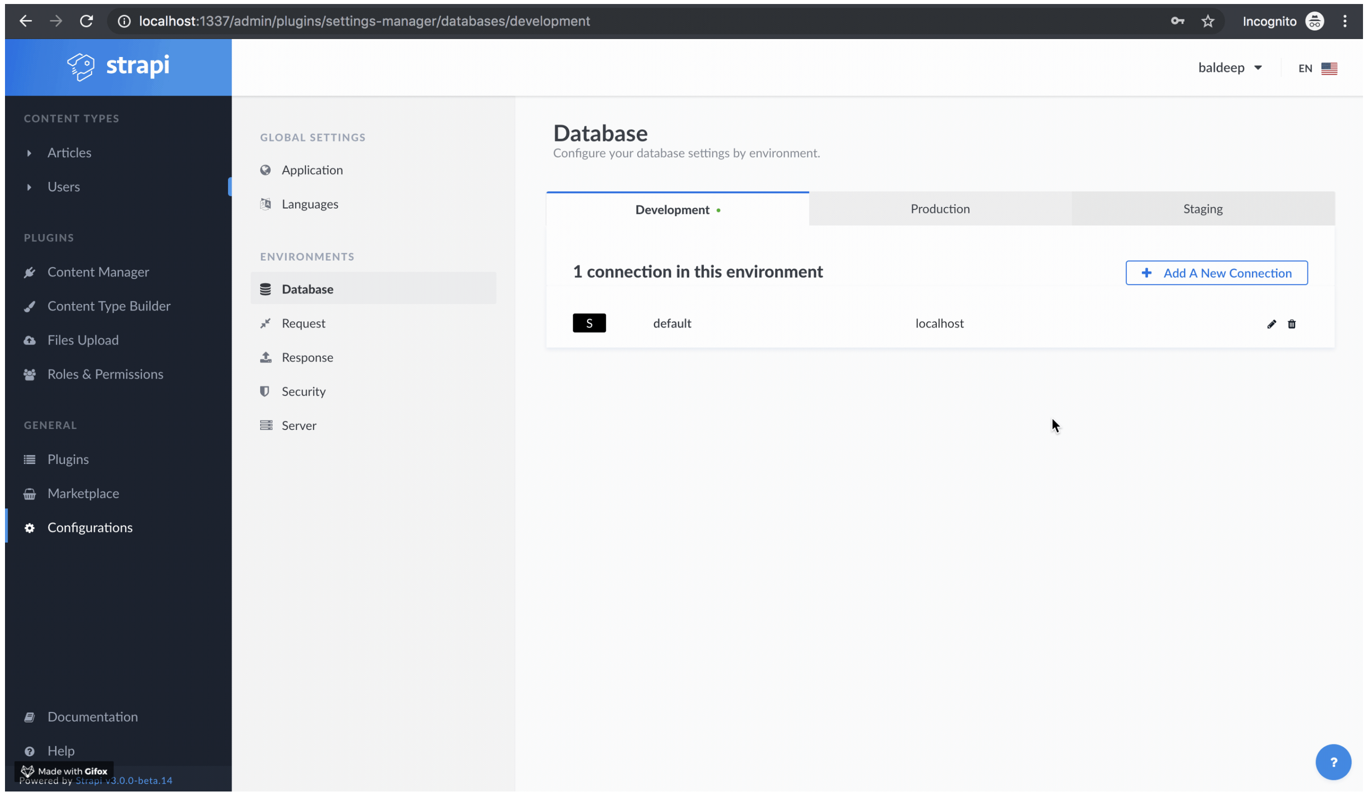
Content Types
Configure the specific settings

Administrator
Article
User

5.

Plug and play from a relational DB to nonrelational DB.

It supports SQLite, MongoDB, MySQL, Postgres.

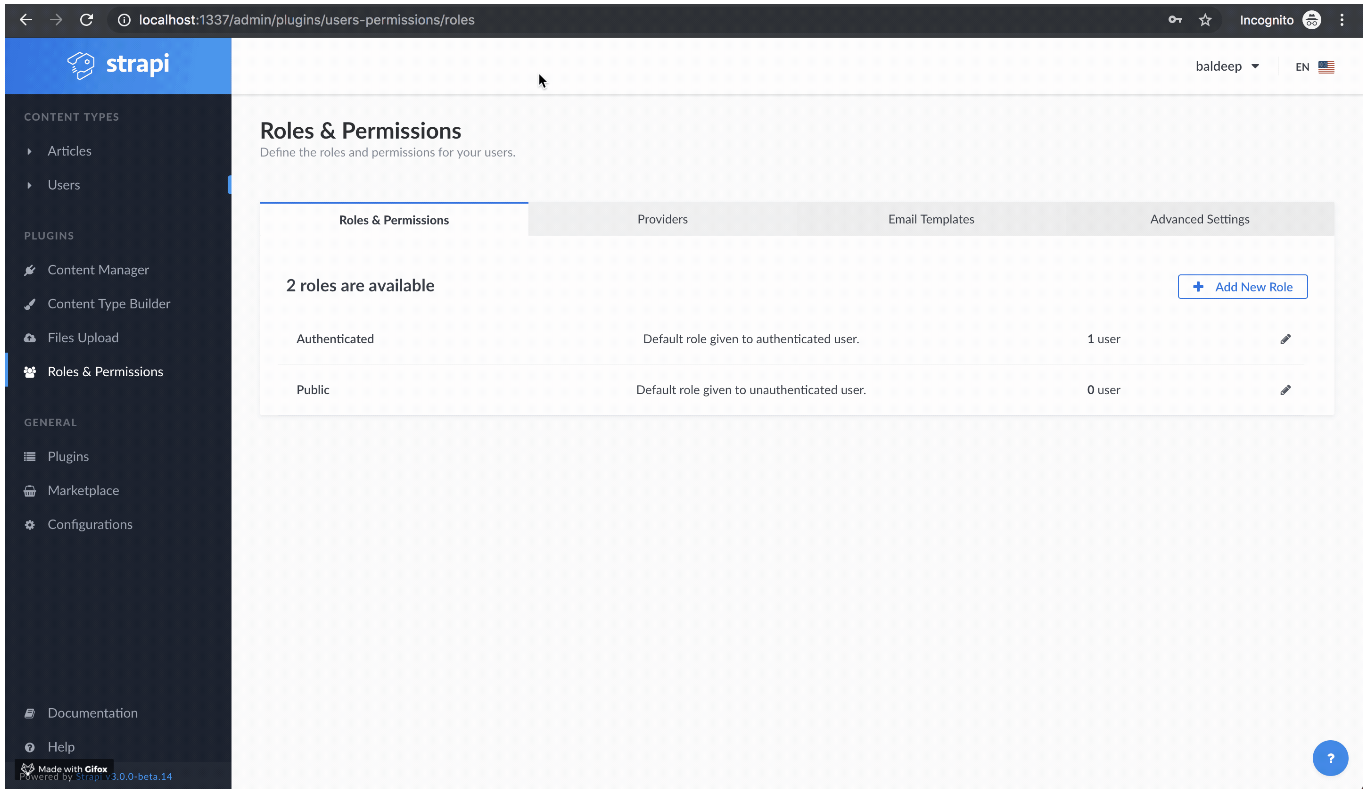


6.

Uploading files to a directory on the server or directly to an AWS S3 Bucket.

7.

Supports OAuth2 authentication for Google, Facebook, Github, Microsoft, Twitter, Discord.



Roles & Permissions

Define the roles and permissions for your users.

2 roles are available

Authenticated	Default role given to authenticated user.	1 user
Public	Default role given to unauthenticated user.	0 user

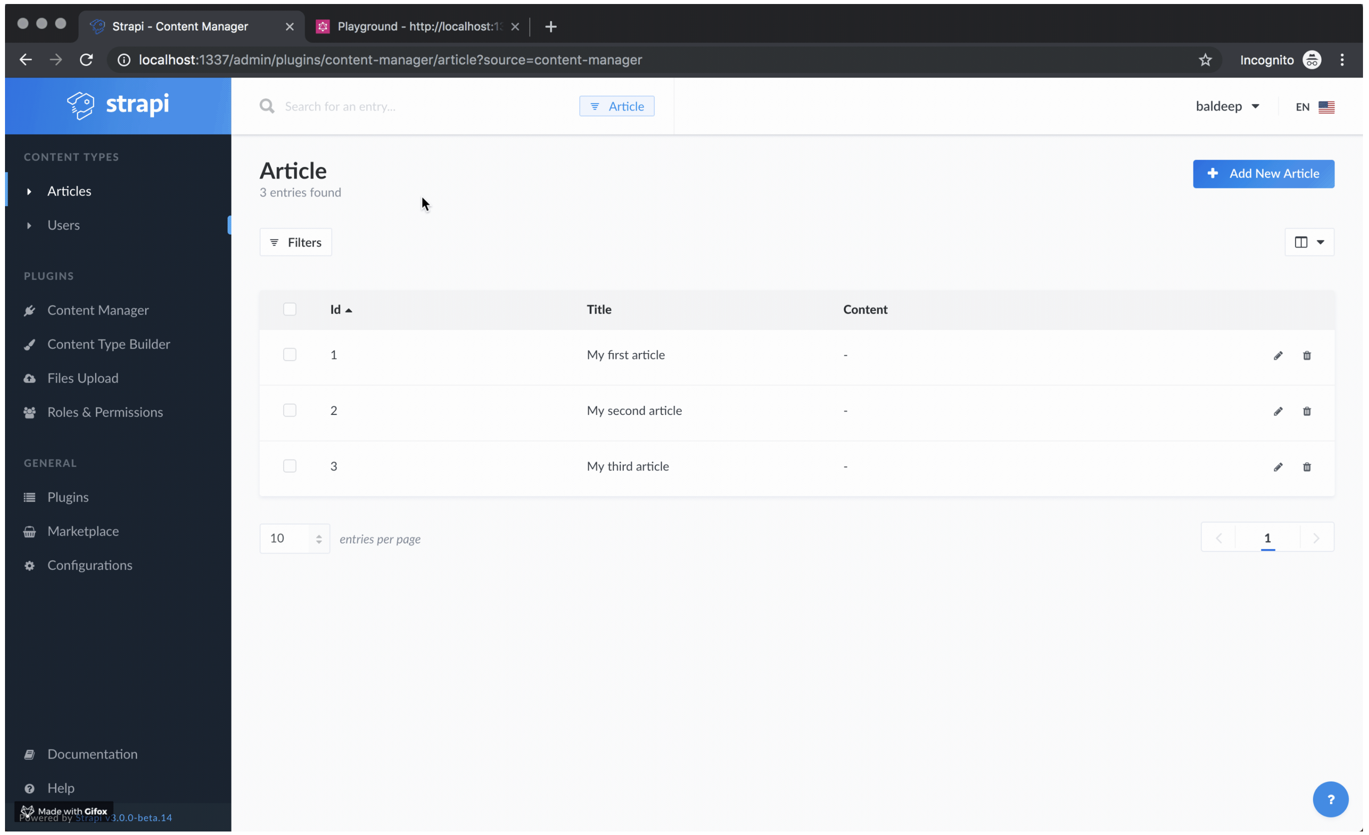
8.

Manage Roles and Users with API level access.

9.

Choose over REST or GraphQL.

It supports REST by default. But you can install a GraphQL plugin to and expose/mutate data.

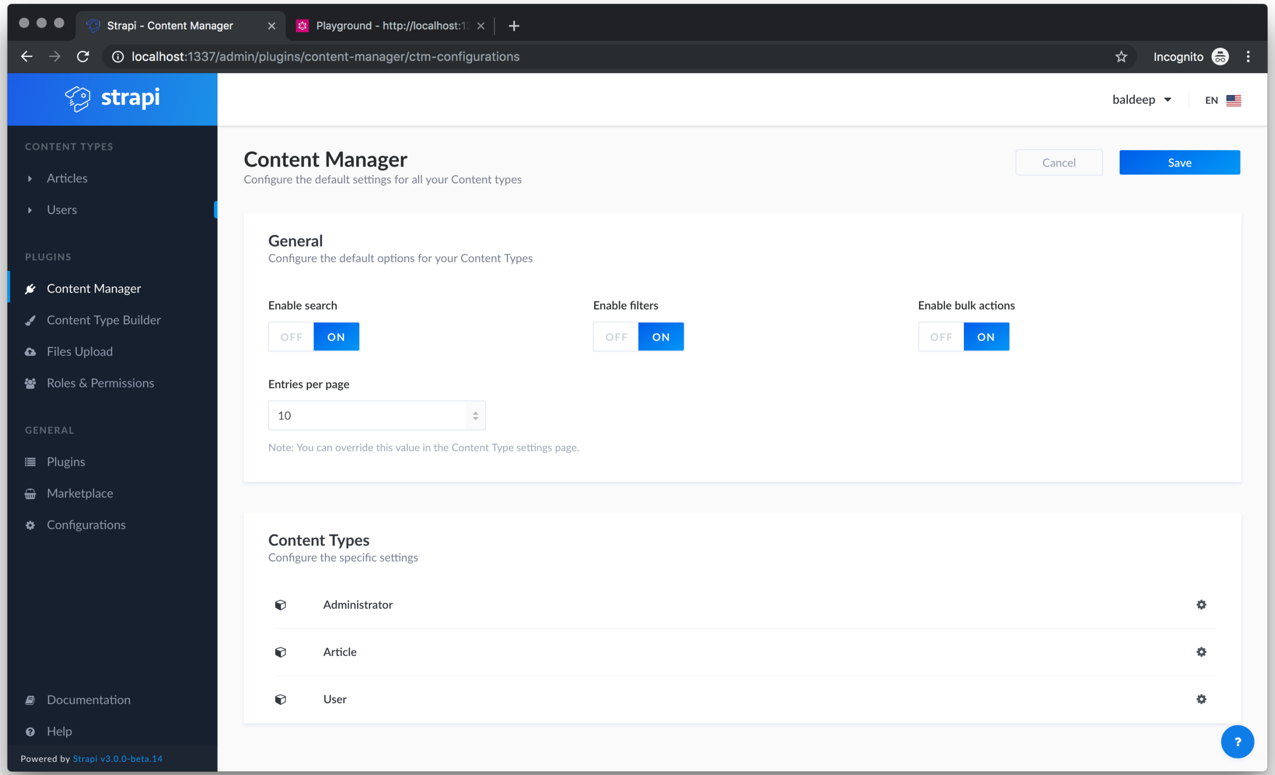


Article
3 entries found

Id	Title	Content
1	My first article	-
2	My second article	-
3	My third article	-

10 entries per page

10. It also gives you an **Admin Dashboard** that is built on **React**, which you can use to create data and perform mundane tasks of roles and user management easily.



Content Manager
Configure the default settings for all your Content types

General
Configure the default options for your Content Types

Enable search: OFF [✓ ON]
Enable filters: OFF [✓ ON]
Enable bulk actions: OFF [✓ ON]

Entries per page: 10
Note: You can override this value in the Content Type settings page.

Content Types
Configure the specific settings

Administrator
Article
User

Strapi's Tech-Stack.

Strapi is built on NodeJS. It internally uses KoaJS as its framework which is developed by the team who created ExpressJS, marketed as 'Next generation web framework'. It works with almost all major databases. For SQL based databases, it uses BookshelfJS internally and for NoSQL DB, it uses Mongoose ORM (strapi-hook-mongoose).

The stack itself is very promising as it is **completely open-source** and the community is also increasing tremendously. All the major common modules like sending emails, uploading files, managing users, are developed in the form of plugins. You can plug and play and tailor them according to your needs.

Cheatsheet

Let me walk you through the below cheat sheet that you should always keep handy 😉

System Requirements

Strapi only requires Node.js and npm. The current recommended version to run strapi is Node v10 (current LTS). This is all that is needed before Strapi can run on your local environment. Download it here

Install Strapi Globally

```
npm install strapi@beta -g
```

This will set up Strapi on your system globally.

Create Project

```
strapi new my-app
```

Three magical words to create your project.

Run Project

```
strapi dev
```

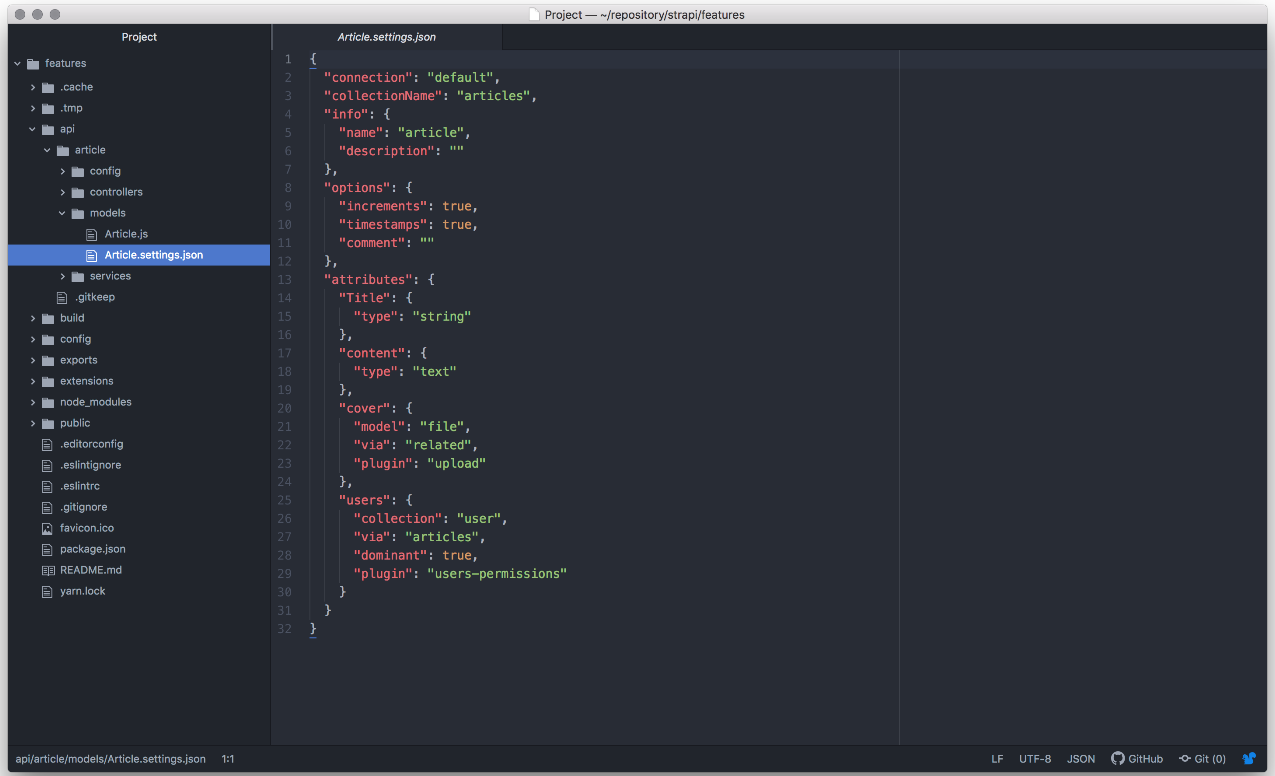
Go to the project folder and run the project in dev mode

Create the Admin User

Once the above step is done, navigate to http://localhost:1337/admin and create your first admin. At least one account is mandatory.

Create Content-Type "Article"

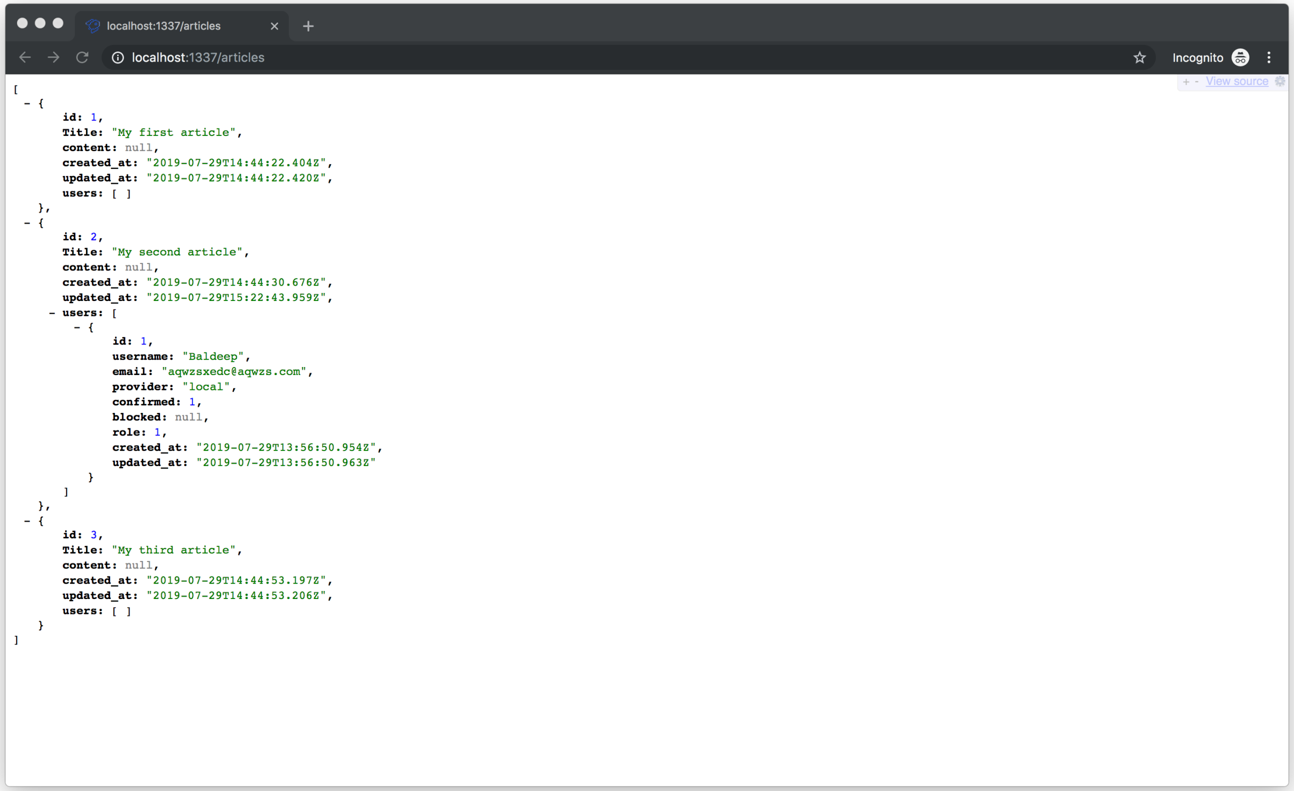
Navigate to Content Type Builder and create your content type. Once done you can add data to it. Strapi will automatically create the file, shown below, for you once you save the content type.



```
{
  "connection": "default",
  "collectionName": "articles",
  "info": {
    "name": "article",
    "description": ""
  },
  "options": {
    "increments": true,
    "timestamps": true,
    "comment": ""
  },
  "attributes": {
    "Title": {
      "type": "string"
    },
    "content": {
      "type": "text"
    },
    "cover": {
      "model": "file",
      "via": "related",
      "plugin": "upload"
    },
    "users": {
      "collection": "user",
      "via": "articles",
      "dominant": true,
      "plugin": "users-permissions"
    }
  }
}
```

Fetch Data for the Content-Type

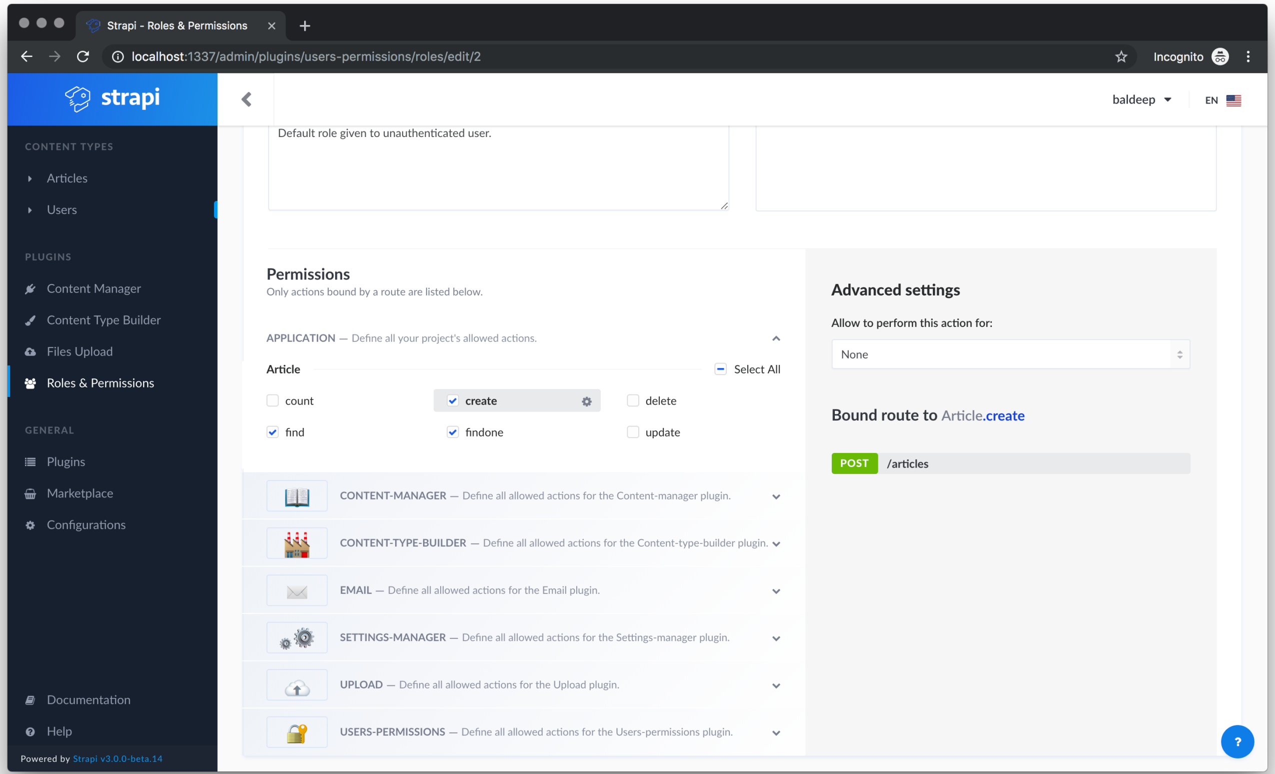
Once the above step is done, navigate to http://localhost:1337/cities



```
[
  - {
        id: 1,
        Title: "My first article",
        content: null,
        created_at: "2019-07-29T14:44:22.404Z",
        updated_at: "2019-07-29T14:44:22.420Z",
        users: [ ]
    },
  - {
        id: 2,
        Title: "My second article",
        content: null,
        created_at: "2019-07-29T14:44:30.676Z",
        updated_at: "2019-07-29T15:22:43.959Z",
      - users: [
          - {
                id: 1,
                username: "Baldeep",
                email: "aqwzsxedc@aqwzs.com",
                provider: "local",
                confirmed: 1,
                blocked: null,
                role: 1,
                created_at: "2019-07-29T13:56:50.954Z",
                updated_at: "2019-07-29T13:56:50.963Z"
            }
        ]
    },
  - {
        id: 3,
        Title: "My third article",
        content: null,
        created_at: "2019-07-29T14:44:53.197Z",
        updated_at: "2019-07-29T14:44:53.206Z",
        users: [ ]
    }
]
```

Protect your API

Go to Roles & Permissions and assign permissions to your models. Permissions includes (count + create + find + findOne + delete + update).



Default role given to unauthenticated user.

Permissions
Only actions bound by a route are listed below.

APPLICATION — Define all your project's allowed actions.

Article [ ] Select All

- [ ] count
- [✓] create
- [ ] delete
- [✓] find
- [✓] findone
- [ ] update

CONTENT-MANAGER — Define all allowed actions for the Content-manager plugin.

CONTENT-TYPE-BUILDER — Define all allowed actions for the Content-type-builder plugin.

EMAIL — Define all allowed actions for the Email plugin.

SETTINGS-MANAGER — Define all allowed actions for the Settings-manager plugin.

UPLOAD — Define all allowed actions for the Upload plugin.

USERS-PERMISSIONS — Define all allowed actions for the Users-permissions plugin.

Advanced settings
Allow to perform this action for: None

Bound route to Article.create
POST /articles

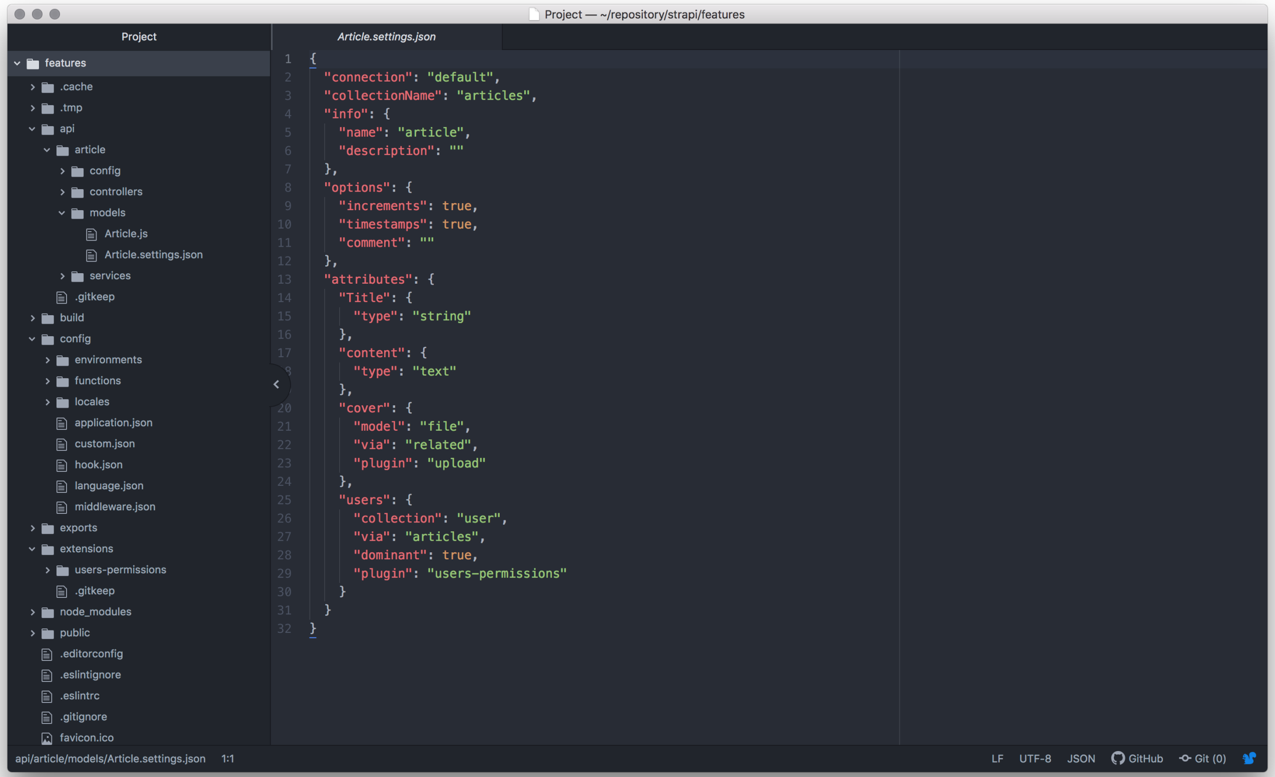
Project Structure

The /api folder contains all the content-types that you create. For example, see the /article folder inside /api.

The /build folder is generated when you run your project. It contains js files for the dashboard which is built on react.

The /config folder contains configuration files for database, server, caching, locales(i18N), cron, etc.

The /extensions folder is where the plugins that you install are.



Project — ~/repository/strapi/features

Project

```
features
  .cache
  .tmp
  api
    article
      config
      controllers
      models
        Article.js
        Article.settings.json
      services
    .gitkeep
  build
  config
    environments
    functions
    locales
    application.json
    custom.json
    hook.json
    language.json
    middleware.json
  exports
  extensions
    users-permissions
    .gitkeep
  node_modules
  public
  .editorconfig
  .eslintignore
  .eslintrc
  .gitignore
  favicon.ico
```

Article.settings.json

```json
{
  "connection": "default",
  "collectionName": "articles",
  "info": {
    "name": "article",
    "description": ""
  },
  "options": {
    "increments": true,
    "timestamps": true,
    "comment": ""
  },
  "attributes": {
    "Title": {
      "type": "string"
    },
    "content": {
      "type": "text"
    },
    "cover": {
      "model": "file",
      "via": "related",
      "plugin": "upload"
    },
    "users": {
      "collection": "user",
      "via": "articles",
      "dominant": true,
      "plugin": "users-permissions"
    }
  }
}
```

api/article/models/Article.settings.json 1:1 LF UTF-8 JSON GitHub Git (0)

Here's hoping you learned a thing or two with this piece. I sure did! :)

> *Do you have a story you would like to share with the community? Let's talk about it here* 🚀

Unleash content.

Learn more

Get Started

strapi

Product

Resources

Strapi is the most popular open-source Headless CMS.

Strapi gives developers the freedom to use their favorite tools and frameworks while allowing editors to easily manage their content and distribute it anywhere.

Why Strapi?

Content Architecture

Features

Enterprise Edition

Partner Program

Roadmap

How to get started

Meet the community

Tutorials

API documentation

GitHub repository

Strapi vs Wordpress

Integrations

Gatsby CMS

React CMS

Vue.js CMS

Nuxt.js CMS

Next.js CMS

Angular CMS

Company

About us

Blog

Careers

Contact

Newsroom