

Strapi.io

—

Headless CMS

# Headless

- Supports APIs without dependency on the presentation layer [1]
- Separates the Model from the View
- Manage content separately from delivery
- Can serve more than one website[2]

# CMS

- Adds scalability to content - changes can be applied globally or granularly by adjusting the model logic
- Simplifies management tasks by exposing them through well defined GUI or CLI processes
- Simplifies access restriction setting and checking through user roles and permissions [3]

# Strapi Features

## Content Management

- CRUD operations through admin panel can then be exposed to API
- API requests can be RESTful or GraphQL
- Endpoints secured by setting user roles
- File uploads locally or through external provider (AWS S3, Cloudinary, etc) [1]

## Content Type Structure

- Content constructed from pre-defined field types, custom fields are on the roadmap [2]
  - String, Text, Number, Boolean, Date, JSON, Email, Password (hashed with bcrypt [3], Media (images, videos, PDFs, etc.), and more [1]
- Validation
  - Private - boolean, attribute removed from response if true
  - Required - boolean, must be present for creation of the model
- Relations
  - Many-to-many, One-to-Many, One-to-one, One-way

# Compared to Other Self-Hosted Headless CMS Options

## Strapi

- Allows full JavaScript stack
- Databases supported [3]
  - PostgreSQL, MongoDB, SQLite, MySQL, MariaDB
- RESTful or GraphQL API
- GUI content manager
- In alpha, frequent bug reports, under active development [8]
- Community forum has 123 members [11]

## Wordpress REST API [1]

- PHP based
- Databases supported
  - MySQL[4]
- RESTful API
- GUI content manager
- Released with WordPress Core in version 4.4 [7]
- Wordpress overall is very popular, and the REST API is well documented [9]
- Violates MVC design pattern [1]

## Cockpit [2]

- PHP based
- Databases supported
  - SQLite, MongoDB [5]
- RESTful API, with GraphQL add-on [6]
- GUI content manager
- Released in 2015, but under active development
- Community forum has 248 users [10]

# Installation

1. Insure Node.js and npm are installed
2. Run “npm install strapi@alpha -g” to install Strapi globally
3. Run “strapi new cms --quickstart” within desired parent folder for the project
4. Navigate to <http://localhost:1337/admin>, and complete the form

**Welcome!**

To finish setup and secure your app, please create the first user (root admin) by entering the necessary information below.

**Username**  
John Doe

**Password**

**Confirmation Password**

**Email**  
@ johndoe@gmail.com

Keep me updated about the new features and upcoming improvements.

**Ready to start**

# Create Content Type (A.K.A. Model)

1. Click “+ Add Content Type” button in Content Type Builder plugin
2. Enter the Name of the New Content Type
3. Enter a brief description
4. Advanced Settings
  - a. Collection Name - Useful when the name of your Content Type and your table name differ

**Add New Content Type** BASE SETTINGS ADVANCED SETTINGS ×

Name

Content Type names should be singular: [Check out our documentation](#)

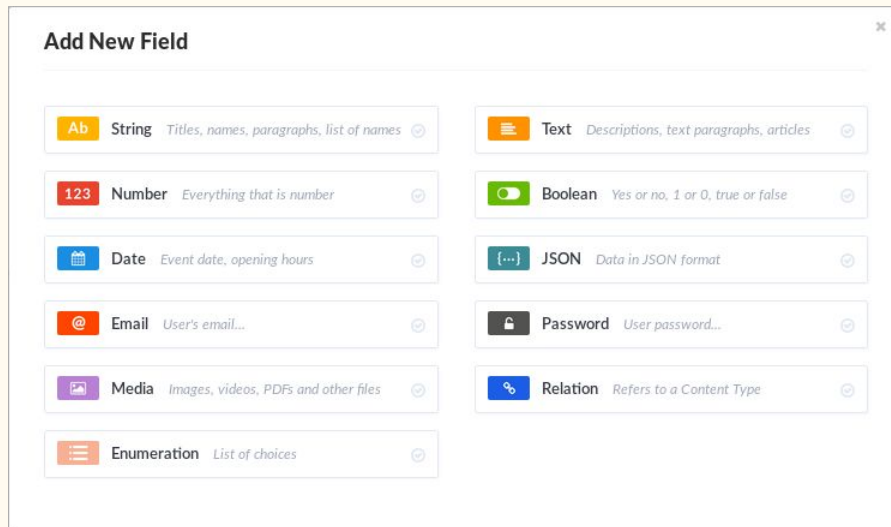
Connection

Description

Cancel Save

# Add Content Type Fields

1. Click “+ Add New Field” button
2. Click desired field type, e.g. “String”



# Add Content Type Fields

1. Click “+ Add New Field” button
2. Click desired field type, e.g. “String”
3. Enter desired name of the field
4. Enter desired Advanced Settings
  - a. Default value
  - b. Required field
  - c. Unique field
  - d. Minimum length
  - e. Maximum length
5. After all fields are added, click “Save”

**Add New *String* Field** BASE SETTINGS ADVANCED SETTINGS ×

Default value

Settings

Required field  
You won't be able to create an entry if this field is empty

Unique field  
You won't be able to create an entry if there is an existing entry with identical content

Minimum length

Maximum length

Cancel + Add New Field Continue



# Create Content Type

## 1. Repeat for Photo Content Type

- Add String field for Name
- Add Media field for Photo
- Add String field for URL
- Add Relation field for Album

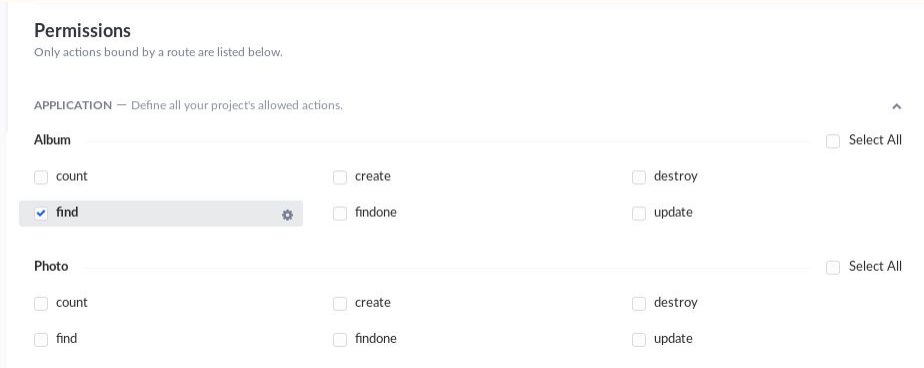
## 2. Relation

- In this case, an Album has many Photos
- Other options
  - \_\_\_ has one \_\_\_
  - \_\_\_ has and belongs to one \_\_\_
  - \_\_\_ belongs to many \_\_\_
  - \_\_\_ has and belongs to \_\_\_

The image shows two screenshots from the Drupal administration interface. The top screenshot displays the configuration for the 'Photo' content type, which is described as 'A model to store details of photo'. It lists three fields: 'Name' (String), 'Photo' (Media), and 'URL' (String). The bottom screenshot shows the 'Add New Relation' dialog, where a relation is being defined between the 'Photo' and 'Album' content types. The 'Photo' type has a field named 'album', and the 'Album' type has a field named 'photos'. The relation is configured as 'Album has many Photos', with the 'many' side on the 'Photos' field of the 'Album' type. The dialog includes tabs for 'DEFINE RELATION' and 'ADVANCED SETTINGS', and buttons for 'Cancel', 'Add New Field', and 'Continue'.

# Set roles and permissions

1. By default, Strapi publishes all Content types with restricted permissions.
  - a. Authentication can be accomplished by password (hashed with bcrypt [1]) or using OAuth to receive jwt token [2].
2. Each created Content Type must explicitly be given permissions
3. The Album type might be given “find” access for the Public Role



The screenshot shows the 'Permissions' configuration page in Strapi. It is titled 'Permissions' and has a subtitle 'Only actions bound by a route are listed below.' Below this, there is a section for 'APPLICATION' with the instruction 'Define all your project's allowed actions.' and an expand/collapse arrow. There are two content type sections: 'Album' and 'Photo'. Each section has a 'Select All' checkbox on the right. Under 'Album', there are checkboxes for 'count', 'find' (which is checked), 'create', 'findone', 'destroy', and 'update'. Under 'Photo', there are checkboxes for 'count', 'find', 'create', 'findone', 'destroy', and 'update'.

# Add Content

1. Two Albums were created
  - a. Name: Landscapes
  - b. Name: Macro-ish

The screenshot displays a web application interface for managing albums. It is divided into three main sections:

- Album (0 entries found):** Shows an empty album with a table header for 'Id' and 'Name'. The table content is 'There is no Album...'. A 'Filters' dropdown and a '+ Add New Album' button are visible.
- New Entry:** A form to create a new entry. The 'Name' field contains 'Landscapes'. On the right, there is a 'Photos (0)' dropdown menu with 'Add an Item...' and a 'Save' button.
- Album (2 entries found):** Shows the album after two entries have been added. The table lists:

Id	Name
1	Landscapes
2	Macro-ish

A 'Filters' dropdown and a '+ Add New Album' button are also present.

# Add Content

1. Two Albums were created
  - a. Name: Landscapes
  - b. Name: Macro-ish
2. Photos were added
  - a. Name entered
  - b. Photo asset dragged to upload area
  - c. “temp” entered for URL
  - d. Album selected and Photo saved
3. Hash given to the media asset was noted from the Files Upload plugin, and added as the URL field of the Photo model with the media type, “.jpg”, appended

New Entry

Reset Save

Name: Morning URL: temp

Photo

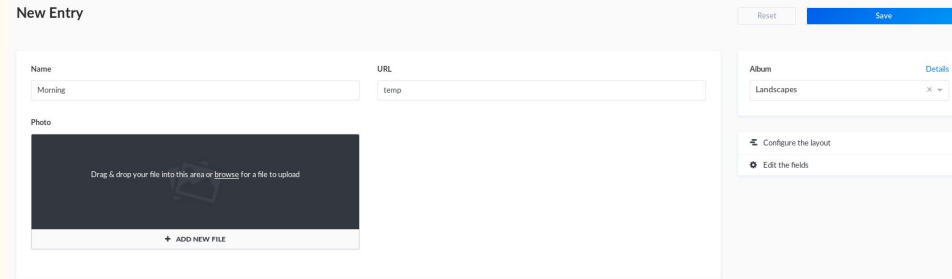
Drag & drop your file into this area or browse for a file to upload

+ ADD NEW FILE

Album: Landscapes

Configure the layout

Edit the fields



1 Delete

Reset Save

Name: Morning URL: 91246272152442b2bc104wd1059d8f6f.jpg

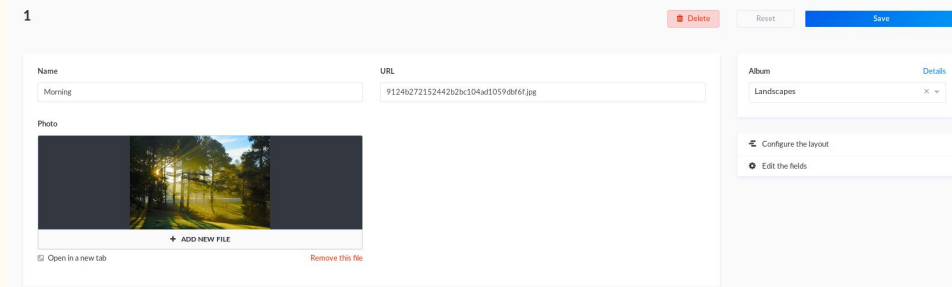
Photo

Open in a new tab + ADD NEW FILE Remove this file

Album: Landscapes

Configure the layout


Edit the fields



# Consume Content

## 1. Using Postman

- a. Issue GET request for album with name “Landscapes”
- b. Examine JSON response

```
Pretty Raw Preview JSON 
```

```
1  [
2
3
4  {
5    "id": 1,
6    "Name": "Landscapes",
7    "created_at": 1555195703645,
8    "updated_at": 1555195703737,
9    "photos": [
10     {
11       "id": 1,
12       "Name": "Morning",
13       "album": 1,
14       "URL": "9124b272152442b2bc104ad1059dbf6f.jpg",
15       "created_at": 1555195772134,
16       "updated_at": 1555196300760
17     },
18     {
19       "id": 2,
20       "Name": "EasternShore",
21       "album": 1,
22       "URL": "1f927edf549b47009f15867bf495d99a.jpg",
23       "created_at": 1555195805377,
24       "updated_at": 1555196306723
25     },
26     {
27       "id": 3,
28       "Name": "Canal",
29       "album": 1,
30       "URL": "6713c0d5d0494bbaac13562582f36d6b.jpg",
31       "created_at": 1555195832087,
32       "updated_at": 1555196312204
33     },
34     {
35       "id": 4,
36       "Name": "GeorgetownBridge",
37       "album": 1,
38       "URL": "9a3bbbb3c614421b9dcf20423160d54d.jpg",
39       "created_at": 1555195869596,
40       "updated_at": 1555196317149
41     }
42   ]
43 }
```

# Consume Content

1. Using Postman
  - a. Issue GET request for album with name “Landscapes”
  - b. Examine JSON response
2. Prototype simple HTML/JavaScript page

This is an example of consuming a photo album.

## Landscapes

Morning



EasternShore



Canal



# The Breakdown

- While Strapi is full-featured and occupies an attractive niche as an open source, free, JavaScript, headless CMS, it is not without shortcomings
- Two very impactful bugs were experienced while creating this simple demo
  - The first was a failure in SQLite when adding the “Required Field” setting to a content type. This is a known issue [1], and it was experienced twice.
  - The second was a shift of the elements in the albums. The first three Photo content types in the first album had no image asset, but these assets had been shifted to other Photo containers such that what had been the first asset was now held by the fourth Photo container. This is not believed to be a known issue, and it was experienced once during this project.
- There is an interesting reddit exchange from three months ago (written Apr 2019) with a co-creator of Strapi [2] in which many shortcomings (“frequent development blocking bugs” is a standout) are acknowledged

# Citations

## Slide Two

- [1] G. Luciano, “Does Your CMS Support Omnichannel?,” *Contentstack.com*, 06-Dec-2016. [Online]. Available: <https://www.contentstack.com/blog/all-about-headless/does-your-cms-support-omnichannel> [Accessed: 13-Apr-2019].
- [2] “Decoupled Content Management Systems,” *npgroup.com*. [Online]. Available: <https://www.npgroup.net/services/development/headless-decoupled-content-management-systems/>. [Accessed: 13-Apr-2019].
- [3] S. Streuli, “The Benefits of a Content Management System,” *Blog*, 03-Oct-2017. [Online]. Available: <https://www.zivtech.com/blog/benefits-content-management-system>. [Accessed: 13-Apr-2019].



# Citations

## Slide Three

- [1] “Customizable API,” *Looking for an efficient open source Headless CMS?* [Online]. Available: <https://strapi.io/overview>.  
[Accessed: 13-Apr-2019].
- [2] “Content,” *Public Roadmap*. [Online]. Available: <https://portal.productboard.com/strapi/tabs/2-under-consideration>.  
[Accessed: 13-Apr-2019].

# Citations

## Slide Four

- [1] D. Žoljom, “Headless WordPress: The Ups And Downs Of Creating A Decoupled WordPress,” *Articles*, 26-Oct-2018. [Online]. Available: <https://www.smashingmagazine.com/2018/10/headless-wordpress-decoupled/#working-with-default-wordpress-rest-api>. [Accessed: 13-Apr-2019].
- [2] A. Heinze, “About,” *Cockpit*. [Online]. Available: <https://getcockpit.com/about>. [Accessed: 13-Apr-2019].
- [3] “Databases,” *Strapi Documentation*. [Online]. Available: <https://strapi.io/documentation/3.x.x/guides/databases.html>. [Accessed: 13-Apr-2019].
- [4] “Creating Databases for WordPress,” *Support*. [Online]. Available: <https://wordpress.org/support/article/creating-database-for-wordpress/>. [Accessed: 13-Apr-2019].
- [5] “Some Key Features,” *Simple Content Platform to manage any structured content*. [Online]. Available: <https://getcockpit.com/>. [Accessed: 13-Apr-2019].
- [6] A. Heize, “Readme,” *CockpitQL*. [Online]. Available: <https://github.com/agentejo/CockpitQL>. [Accessed: 13-Apr-2019].
- [7] R. McCue, “REST API: Welcome the Infrastructure to Core,” *Make WordPress Core*, 28-Oct-2015. [Online]. Available: <https://make.wordpress.org/core/2015/10/28/rest-api-welcome-the-infrastructure-to-core/>. [Accessed: 13-Apr-2019].

# Citations

## Slide Four (continued)

[8] “Issues,” *Strapi*. [Online]. Available: <https://github.com/strapi/strapi/issues>. [Accessed: 13-Apr-2019].

[9] “Rest API Resources,” *Documentation*. [Online]. Available: <https://developer.wordpress.com/docs/api/>. [Accessed: 13-Apr-2019].

[10] “Community,” *Cockpit Community*. [Online]. Available: <https://discourse.getcockpit.com/>. [Accessed: 13-Apr-2019].

[11] “Posts,” *Strapi Community*. [Online]. Available: <https://spectrum.chat/strapi?tab=posts>. [Accessed: 13-Apr-2019].

# Citations

## Slide Ten

[1] “User.js,” *Strapi*. [Online]. Available:

<https://github.com/strapi/strapi/blob/196fa038bd213710798d3bb5b8e7602fc00e9f5d/packages/strapi-plugin-users-permissions/services/User.js>. [Accessed: 13-Apr-2019].

[2] “Authentication,” *Strapi Documentation*. [Online]. Available:

<https://strapi.io/documentation/3.x.x/guides/authentication.html#registration>. [Accessed: 13-Apr-2019].

# Citations

## Slide Fifteen

[1] "SQLite error when adding required field with blank default ," *github.com*. [Online]. Available:

<https://github.com/strapi/strapi/issues/2941>. [Accessed: 13-Apr-2019].

[2] "Learn Strapi (node.js headless CMS) in 12 minutes," *reddit.com*. [Online]. Available:

[https://www.reddit.com/r/node/comments/afaf5n/learn\\_strapi\\_nodejs\\_headless\\_cms\\_in\\_12\\_minutes/edwxtk4/?context=3](https://www.reddit.com/r/node/comments/afaf5n/learn_strapi_nodejs_headless_cms_in_12_minutes/edwxtk4/?context=3). [Accessed: 13-Apr-2019].