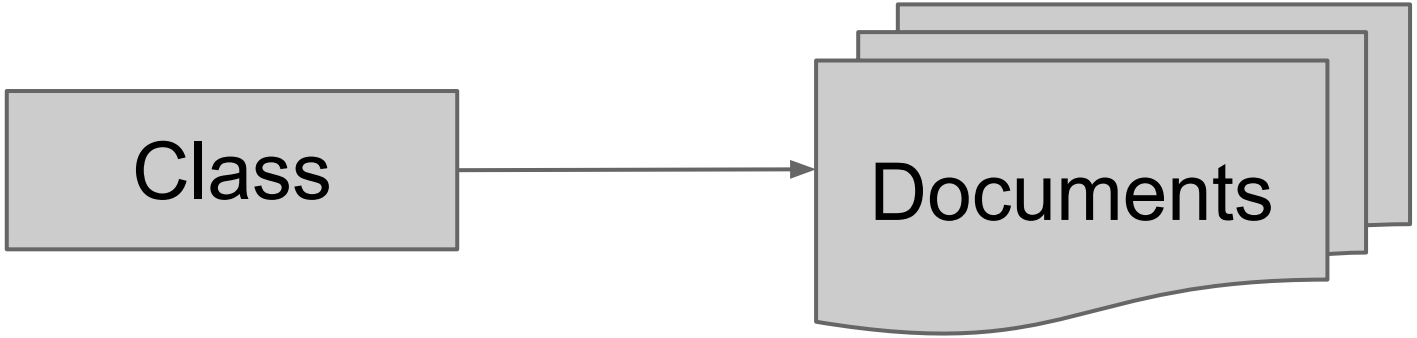
Hammock

Web Programming Environment + CMS

Classes and Documents

Hammock is based on the idea of Classes and Documents. A class is a kind of template for a JSON document.



Classes and Documents

Class Definition

```
{
  name: "Post",
  fields: {
    title: { type: "string", ...},
    category: { type: "select",
      options: ["main", "news", ...]
    },
    author: { type: "select",
      ref: 'User' }
    }
  }
}
```

Document Template

```
{
   title: "<string>",
   category: "<select:category>",
   author: "<select:User>"
}
```

How to define a class (like a schema)

```
{
  _id: "class/post",
  name: "Post",
  fields: {
    title: { type: "string", required: true },
    category: { type: "select",
                options: ["main", "business", "tech"] },
    author: { type: "select", ref: 'User' }
  }
}
```

Creating a new document with JSON editor

The JSON editor in Hammock understands the class definition, and fills in the template as you press Tab

Creating a new document with JSON editor

Just keep pressing Tab, autocomplete helps you fill in the template

```
1 {
2     "title":"Some title",
3     ""
4 }
```

category
state

```
1 {
2     "title":"Some title",
3     "category": ""
4 }
```

main
news
business
tech

New class template

Just like documents are instances of a class, classes themselves are documents which are instances of the “Meta-class”

Meta-class definition

```
{ "name": "class",
  "fields": {
      "name": { "type": "string", "required": true },
      "fields": { "type": "object", "required": true, "value_fields": {
          "type": { "type": "select", "required": true,
                    "options": ["string", "select", "boolean", "object", "array",
                                "number", "function", "markup", "date"] },
          "required": { "type": "boolean" /* default: false */ },
          "options": { "type": "array" },
          "fields": { "type": "object" },
          "value_fields": { "type": "object" },
          "ref": { "type": "select", "ref": "class" },
          "ref_field": { "type": "string" /* default: "name" */ }
}}}}}
```

Meta-class details: field types

field type	description
string	e.g. “something”
select	a choice from a list
boolean	true | false
object	a json object
array	a json array
number	e.g. 1234
function	a string which looks like “function(...) { … }”
markup	a string which contains markup
date	a string which is a date

Meta-class details: field attributes

attribute	type	required/allowed for
type	one of ["string", "select", "boolean", "object", "array"]	yes
required	boolean	no/allowed for any field (default false)
options	array	allowed for select, array*
fields	object	allowed for object*
value_fields	object	allowed for object, array*
ref	select	allowed for select, array*
ref_field	string	no/allowed if ref_class is specified (default is “name”)

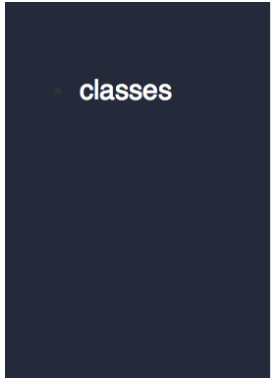
Using Hammock as a CMS

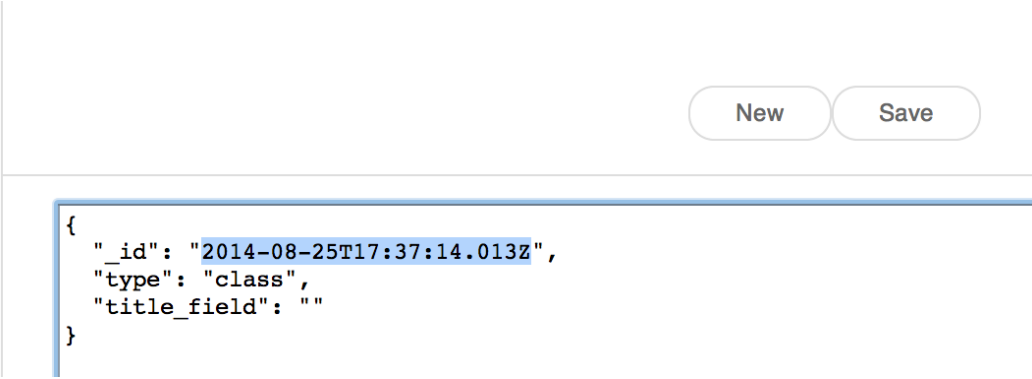
Hammock is currently in early development, we will show you a demo v0.1

As you will see, we can already use it as a simple CMS system.

Demo: create a new class

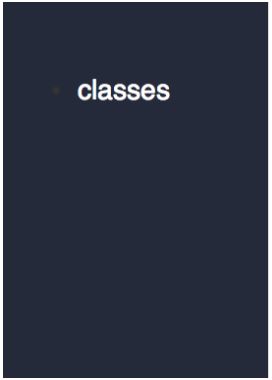
1. click classes
2. click New
    - you'll see a new object, with an auto-generated _id

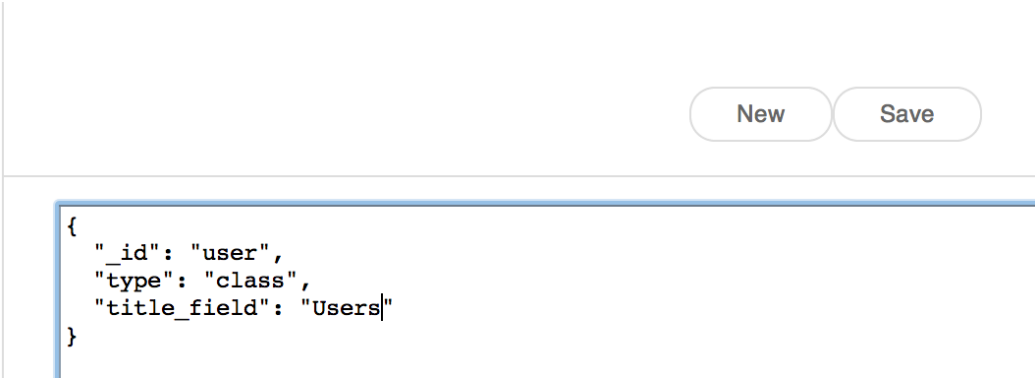




Demo: create a new class

3. replace the auto-generated _id with "class/user"
4. set title field to "Users"
5. click Save
    - the browser will ask you if you want to create a new object, click Yes





Demo: create a new instance document

1. click Users in left nav (this represents the user class we just created)
2. click New
3. you'll see autogenerated _id and class fields. You can leave this.
4. set the username (currently title_field)
5. set full name (currently body_field)
6. click Save

Hammock as a functional extension to JS

In imperative (object oriented) programming, instances of classes (objects) live in memory and are often variables. CoffeeScript and ES6 both implement this kind of a classes, and they fit into the imperative paradigm.

Instances of hammock classes are documents, not objects in memory, therefore **Hammock is a declarative (functional) language extension**.

Hammock for web programming

Hammock classes can be defined for application data, such as a blog post or user, but they can also be used as **building blocks** for web programming.

A UI component using React.js is a perfect example of how Hammock classes and documents can be used as a programming building block.

More to come...

Stay tuned for more news.

A mailing list will be set up soon. In the meantime, you can follow us on Facebook:

facebook.com/HammockRocks

questions, more info: vonwao@gmail.com