Using Keystone

# Creating a new keystone application

use the yeoman generator

some error about son

# Key files

## package.json

describes project

keystone.js (called web.js on the website)

run keystone website

also has the routes for the admin UI

## Models/ModelName.js

contains the data model

## Public

For CSS and images etc

## Routes

### index.js – defining the routes

defines the routes for the public views. This is where you tell the app what the routes should be. It doesn’t link the links to the routes for the users.

app.get('/', routes.views.index);

app.get('/blog/:category?', routes.views.blog);

app.get('/blog/post/:post', routes.views.post);

app.get('/knowledge/:category?', routes.views.knowledge);

app.get('/knowledge/knowledge/:knowledge', routes.views.knowledgeItem);

app.all('/contact', routes.views.contact);

### Middleware.js – exposing the routes to the user

exports.initLocals = function (req, res, next) {

res.locals.navLinks = [

{ label: 'Home', key: 'home', href: '/' },

{ label: 'Musings', key: 'blog', href: '/blog' },

{ label: 'Resources', key: 'knowledge', href: '/knowledge' },

{ label: 'Contact', key: 'contact', href: '/contact' },

];

res.locals.user = req.user;

next();

};

## Templates folder

Can use different template builders. I am using the yukky pug, because that’s what Keystone does. I preferred handlebars, and am working on using react.

### Layouts

Contains the general stuff to go across all pages.

Views

This contains the specific pages that you want to display.

# General file structure and flow

# Working in the app

# Useful notes

## Routes and views

Usually, the easiest and clearest way to configure the logic for different routes (or views) in your application is to set up all the bindings single file, then put any common logic (or middleware) in another file.

Then, the controller for each route you bind goes in its own file, organised similarly to the template that renders the views.

To see your home page, point your browser at localhost:3000. You should see our Hello World! message.

To sign in to Keystone's Admin UI, go to localhost:3000/keystone. Use the email and password you put in the update script above to sign in, and you'll be redirected to Keystone's home page.

# Working notes – my current problem

## How do I add content types as a way of retrieving and filtering the knowledge resources?

### Create the model of contentTypes

Create a model for content types. Do this by creating a contentTypes.js file in the models folder. Add the fields you want about the content types here. See the keystone documentation about the types of fields etc you can have, plus the mongo documentation.

Ensure that the contentTypes and knowledge resources are related, so that when you add a knowledge resource you can choose one or more contentTypes.

### Enable a user to add a contentType

This is done by the admin user. This is easy.

## Enable a user to see - for each knowledge resource - it’s contentTypes tags

### First pathway – user clicks on “Knowledge” tab and sees a list of all knowledge resources

I want to display the following details about EACH knowledge item.

1. Name of the resource
2. Link TO the resource – to either the FULL DESCRIPTION or “TAKE ME TO THE RESOURCE”.
3. **Type** of resource it is eg blog, article etc.
4. Topic that the resource has attached to it
5. Description of the resource

### Browser side – THE REQUEST

When a USER clicks on the **KNOWLEDGE NAV**, I need to send a REQUEST to the SERVER asking for the information to the KNOWLEDGE NAV. The information that we get from the BROWSER ie the LOCALS/PARAMS is that the KNOWLEDGE NAV TAB has been clicked. Somehow, the REQUEST object contains this information. Is this a GET REQUEST?

### SERVER SIDE

On the SERVER side, we need to trigger the following:

1. RETRIEVE THE ITEMS from the KNOWLEDGE COLLECTION of the DB: Search the relevant part of the KNOWLEDGE COLLECTION database to FIND ALL KNOWLEDGE ITEMS. We need to push these KNOWLEDG ITEMS into an object/array in the following format and with the following details:

[

{\_id: 12345

title: title name,

link: URL,

state: published,

dateAdded; [Ask Ben where this is],

publishedDate: date,

categories: [category 1, category 2],

contentTypes: [category 1, category 2],

content: {

description: “Text”,

extended: “Text”

},

{

\_id: 12346

title: title name,

link: URL,

state: published,

dateAdded; [Ask Ben where this is],

publishedDate: date,

categories: [category 1, category 2],

contentTypes: [category 1, category 2],

content: {

description: “Text”,

extended: “Text”

},

]

MAKE the information available to the BROWSEr

TELL the browser how to place the information on the apge.

will hit a button or a link. That will trigger the browser to send a GET request to the server to ask for information. That GET request needs to include information about which data we are seeking from the server.

In practice, when a user pressed the navigation link “Knowledge”, the browser sends a REQUEST to the server for the information which is associated with the navigation link “Knowledge”. I want to display all

Make the data in the database available to the browser to

### Enable a user to see all of the knowledge resources of a particular contentType.