package.json,

There is also a command to automatically install everything under “dependencies”

package.json is recipe for the ingredients that are needed for a specific package or library.

entry point field in package.json: entry point where our application starts.

Order of routes matter.

Route parameters/variables or path variables.

req.params contains all of the route parameters and their corresponding values.

Express uses ejs package to render an ejs file.

We can dry up our template using Partials

By default, it does not serve anything at all apart from **views** directory

The directory can be named anything but ‘public’ is a common name.

If we have a css file in a directory named public, then in the server file (app.js file) we can write:

app.use (express.static(“public”));

This will tell “express” to serve the content of the public directory.

If we want to tell Express to expect .ejs files

app.set (“view engine”, “ejs”);

Partials are file templates that we can write and include in other templates.

Send data to an ejs file through an object.

No need to write header.ejs if we have specified to look for ejs file.

req.body contains all of the data from the request body. We can send data to be added to the database in this req.body

Express does not actually create the req.body for us.

We need to explicitly tell it to take the req.body and turn it into a Javascript object for us.

To do this, we have to install body-parser package

Anytime we want to extract the data sent through the form on the server side, we need to use body-parser.

**API methods**: methods that the package supports.

When we have a form, all of the form data is put in the request body and then it gets to our Express app and we can pull it out of the request body.

Anytime we want to extract the data sent through the form on the server side, we need to use body-parser.

IFTTT: If this then that

Visual interface to connect API.

Using internet of things, we can control a physical object using an API.

Important point:

When we make http request, we get html back.

APIs do not respond with html. Html contains information about the structure of a page.

APIs respond with data, not structure.

To use javascript in order to make API requests, we use “request” package.

We can make an http and api requests using curl.

Requests from node can be made using “request” package.

We can make http requests using “request” package.

Debugging: use “locus” package.

“partials” directory is created inside “views” directory.

“request” package used in movie api, it makes a request to the url.

**IMP NOTE**:

req.query: For GET request

req.body: For POST request

7 different routes that follow REST convention.

We set EJS as the view engine for our Express application using app.set(‘view engine’, ‘ejs’)

We send a view to the user using res.render()

res.render() looks in a “views” folder for the view.

**NOTE**:

Handle the situation of submitting empty campground name and empty image by making those fields in the form as required.

Extended JSON is used to represent BSON data types.

Extended json is used to represent data stored in BSON format in readable format.

“db” refers to the database we are currently on.

const bodyParser = require(“body-parser”)

app.use( bodyParser.urlencoded( { extended: true } ) ) ;

body-parser is an NPM package that parses incoming request bodies in a middleware before our handlers, available under the req.body property.

app.use( bodyParser.json() ) looks at requests where the Content-type: application/json header is present and transforms the text-based JSON input into JS-accessible variables under req.body

app.use(bodyParser.urlencoded( { extended: true } ) ) does the same for URL-encoded requests

The extended: true precises that the req.body object will contain values of any type instead of just strings.

**NOTE**

We don’t need to install “request” package for Yelpcamp

render() takes in file names while redirect() takes in names of routes.

Mongoose: Object data mapper

Javascript layer on top of mongodb.

Route 🡨🡪 Route

Route 🡪 ejs Template

ejs tempate 🡪 Route

Create: 2 routes

Update: 2 routes

**RESTful Routes**:

Index, New, Create, Show, Edit, Update, Destroy

new.ejs 🡪 Post route (Create)

Edit 🡪 Update

“express-sanitizer” package

Data Associations

Embedding data,

Referencing data

Error driven development.

In case of associations there could be nested routes.

Grid system

Only a logged in user can make a new comment.

passport: Authentication middleware

passport-local: A scheme for authentication.

“passport-local” is for username and password.

“passport-local-mongoose”

Mongoose plugin that simplifies building username and password login with passport.

“sessions” add state to http requests.

To implement sessions, we use a package called “express-session”

Middleware: Runs before the route callback function.

Logging out:

Passport destroys the user data in this session. It no longer keeps track of user data in this session.

app.use(function(req, res, next) {

res.locals.currentUser = req.user;

next(); // move on to the actual next code.

});

“req” gets the “user” property. **How** ?

This function passes req.user to every single template.

It is a middleware that will run for every single route.

v6:

only a logged in user can make a comment on a campground.

What is a middleware ?

body-parser: Node.js parsing middleware

var express = require("express") ;

var router = express.Router();

Prevent an unauthenticated user from creating a campground.

Link user to the comment model.

Hide the form for creating a new campground from unauthenticated user.

Avoid sending post request from anywhere.

“req.user”

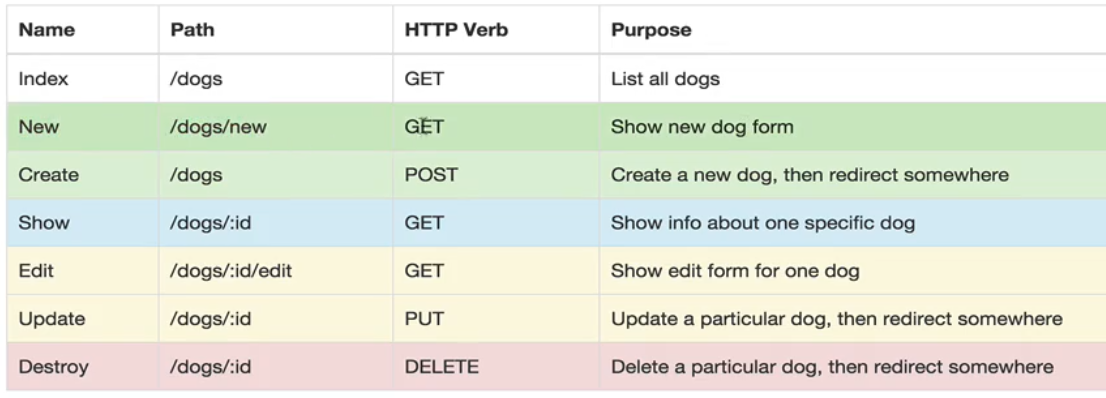
Passport creates this for a particular session. And it stores information about the currently logged in user.

Store user information for a campground.

express.Router({mergeParams: true});

to use a “variable” here that comes from the pre-pended route.

Make some of the fields of campground to be required in order to avoid submitting empty details.



If we have require(“directoryName”) then this gives us the contents of index.js file by default.

Show route should come after New route as the order of routes matter.

req.params

req.body

Only a campground author can edit or delete the campgrounds.

Similarly for comments.

Main file that we run in app.js

Method-override package

Used for http PUT requests.

Edit route: “value” field is useful

Check if the user is logged in ?

Use req.isAuthenticated()

res.redirect(“back”)

// will take user back to the place from where he came from.

Middleware is called before we get to the route handler/callback.

findByIdAndUpdate takes 3 things.

The id to find by, the data to update with and a callback to run afterwards.

“index.js” is a special name.

app.use(function(req, res, next) {

res.locals.currentUser = req.user;

res.locals.error = req.flash("error") ;

res.locals.success = req.flash("success");

next(); // move on to the actual next code.

});

The variables such as ‘currentUser’, ‘error’ and ‘success’ are available in the header template.

res.locals

flash does not flash itself right away. It only works or shows up on the next page.

We pass a key-value pair. We can access the flash message using the key.

Passport takes care of the case when we try to sign up with an existing user or provide an empty username or password.

Modernizr

express-session must be used before passport.

All the requests we make follow same origin policy. The starting url is the same for all routes.

For the animation, each image remains takes 5s to fade in, remains for 5s on the page and takes 5s to fade out.

Total 15 seconds. Which is 30% of the total animation time.

And we keep images separate by adding in delays.

And when the first image starts to fade out, the next image starts to fade-in in parallel.

This helps to reduce the time from 15 seconds to 10 seconds.