

---

## Nodal

### API Services Made Easy with Node.js



View the website at [nodaljs.com](https://nodaljs.com).

Nodal is a web server and opinionated framework for building data manipulation-centric (Create Read Update Destroy) API services in Node.js for web, mobile or IoT apps.

### Why Nodal?

Hello, Nodal — Building Node.js Servers for Everybody is our first blog post that helps you get acquainted with the reasons behind the creation of the framework. :)

Post Parse Prototyping is also a fantastic read explaining the benefits of Nodal for quick and easy mobile / IoT backend development.

### Overview

Nodal is built upon an ideology of a robust, scalable architecture for data storage and retrieval APIs. It is an opinionated, explicit, idiomatic and highly-extensible full-service framework that takes care of all of the hard decisions for you and your team. This allows you to focus on creating an effective product in a short timespan while minimizing technical debt.

Nodal servers are not meant to be monoliths. They're *stateless* and *distributed*, meant to service your needs of interfacing with your data layer effortlessly. While you can output any data format with Nodal, it's recommended you offload things like static page rendering to other optimized services like CDNs.

Check out the first Nodal Screencast [here](#).

---

## Stateless Dogma

It's important to note that Nodal is meant for **stateless** API services. This means you should not rely on memory within a specific process to serve multiple requests, and Nodal will use process clustering (even in development) to actively discourage this practice. If you need to work with unstructured data for rapid prototyping, *connect Nodal to a PostgreSQL database* and use the "JSON" field type. You'll find yourself encountering a lot of trouble if you start trying to use in-process memory across different requests.

Remember: **one input, one output**. Side effects dealing with model state should be managed via your Database. Nodal should not be used for streaming (long poll) requests and the HTTP request and response objects are intentionally obfuscated.

This also means you *can not rely on socket connections*. If you need to incorporate realtime functionality in your application, there should be a separate server responsible for this. It can interface with your Nodal API server and even receive events from it, but your API server should never have a stateful (prolonged) connection with any client.

## Getting Started

Getting started with Nodal is easy.

1. Download and install the newest Node 6.x version from [nodejs.org](http://nodejs.org)
2. Open terminal, and type `npm install nodal -g`. (If you get an error, run `sudo npm install nodal -g` or fix permissions permanently by following these directions
3. Using your terminal, visit your projects folder. Perhaps with `cd ~`.
4. Run `nodal new`.
5. Follow the on screen instructions, enter your new project directory and type `nodal s`.

That's it! Your Nodal webserver is up and running.

## Hooking Up Your Database

Once Nodal is up and running, it's likely that you'll want to connect your project to a database. Nodal comes packaged with Migrations, a Query Composer and full PostgreSQL integration.

First you'll need to install PostgreSQL. OS X users, I recommend using Postgres.app for your development environment.

Once you've installed Postgres, make sure to run:

---

```
1 $ createuser postgres -s
```

To create a default postgres superuser with no password. (Default for Nodal's configuration.)

To begin using your database, start with:

```
1 $ nodal db:create
```

To create the database and then,

```
1 $ nodal db:prepare
```

To prepare for migrations.

From here, `nodal db:migrate` runs all pending migrations and `nodal db:rollback` will roll back migrations, one at a time by default.

## Server Types

Nodal works best when you follow its ideology, and that means creating a new service to solve specific *Problem Domains* of your application and business.

The main three suggestions are **Branding Server**, **API Server** and **Application Server**.

Nodal's core competency is building API servers. We do, however, also have a project called dotcom for building Branding Servers (search engine optimized server-generated pages). More on this soon.

### API Server

Create an API server using Nodal's Models, PostgreSQL integration, built-in JSON API formatting, and Query Composer (ORM). Bi-directional migrations are packaged with Nodal, meaning you can maintain the integrity of your data. User (including password) and OAuth AccessToken models and controllers are pre-built for you and can be added easily to your project.

Packaged with Nodal are workers, scheduling modules, and much more for all of your data needs.

We can look at what an API Controller might look like for, say, blog posts:

```
1 class BlogPostsController extends Nodal.Controller {
2
3   index() {
4
5     BlogPost.query()
6       .join('user')
7       .join('comments')
```

---

```
8     .where(this.params.query)
9     .end((err, blogPosts) => {
10
11         this.respond(err || blogPosts);
12     });
13
14 }
15
16 show() {
17
18     BlogPost.find(this.params.route.id, (err, blogPost) => this.respond
19         (err || blogPost));
20
21 }
22
23 create() {
24
25     BlogPost.create(params.body, (err, blogPost) => this.respond(err ||
26         blogPost));
27
28 }
29
30 update() {
31
32     BlogPost.update(this.params.route.id, params.body, (err, blogPost)
33         => this.respond(err || blogPost));
34
35 }
36
37 destroy() {
38
39     BlogPost.destroy(this.params.route.id, (err, blogPost) => this.
40         respond(err || blogPost));
41
42 }
```

## Beginner's Guide

You'll be able to learn more about Nodal at [nodaljs.com](https://nodaljs.com).

## Documentation

Check out the website at [nodaljs.com](https://nodaljs.com).

---

## **Roadmap**

View the roadmap at ROADMAP.md.

## **About**

Nodal is under active development and maintained by Keith Horwood.

### **Contributors welcome!**

Follow me on Twitter, @keithwhor

Fork me on GitHub, keithwhor

Thanks for checking out Nodal!