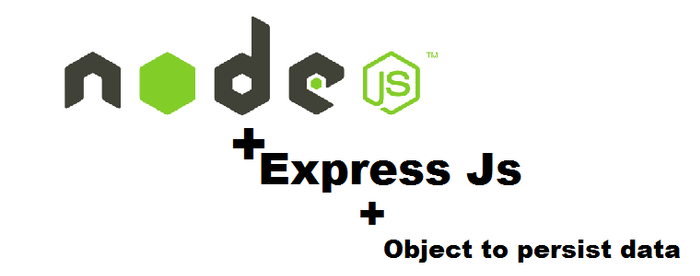
**Building a simple API with Nodejs, Expressjs and JS Data structure to persist data - 1**

Published Jul 29, 2018Last updated Feb 22, 2019

uilding a simple API with Nodejs, Expressjs and JS Data structure 

I met someone recently with some difficulties understanding node js and expressjs. So I decided to put this together with the hope that it will help somebody.

In this tutorial, you'll learn how to build a simple API with Nodejs using ES6. What is Nodejs, you may ask- According to [Wikipedia](https://en.wikipedia.org/wiki/Node.js) Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside the browser. It is a javascript runtime built on Chrome V8 Javascript Engine. Simply put Nodejs give us the ability to write backend code in Javascript.

**What we will build?**

You'll learn how to build a reflection API - A reflection app gives users the ability to reflect and document daily successes, failures and a plan on what to do better the next day.

*Success is not final; failure is not fatal: It is the courage to continue that counts. -- Winston S. Churchill*

**Getting Started**

Before you continue, install the following on your system if you don't already have

* [Nodejs](https://nodejs.org/en/download/)
* [NPM](https://www.npmjs.com/) - NPM is a package manager for Javascript

On your terminal, run the following command to check if Nodejs and npm are successfully installed on your system



**Project Setup**

* Create a new project directory on your system, you can call node\_express\_tutorial
* Change working directly to the project and run npm init on your terminal or command prompt if you're using a window system - Running npm init will prompt you with some questions to help set up your project



What that is done, you should see package.json file in your project and it contains basic information about your project.

{

"name": "node\_express\_tutorial",

"version": "1.0.0",

"description": "node express tutorial",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "Olawale Aladeusi",

"license": "ISC"

}

Next, let's install all the package dependencies that we'll need to build the project

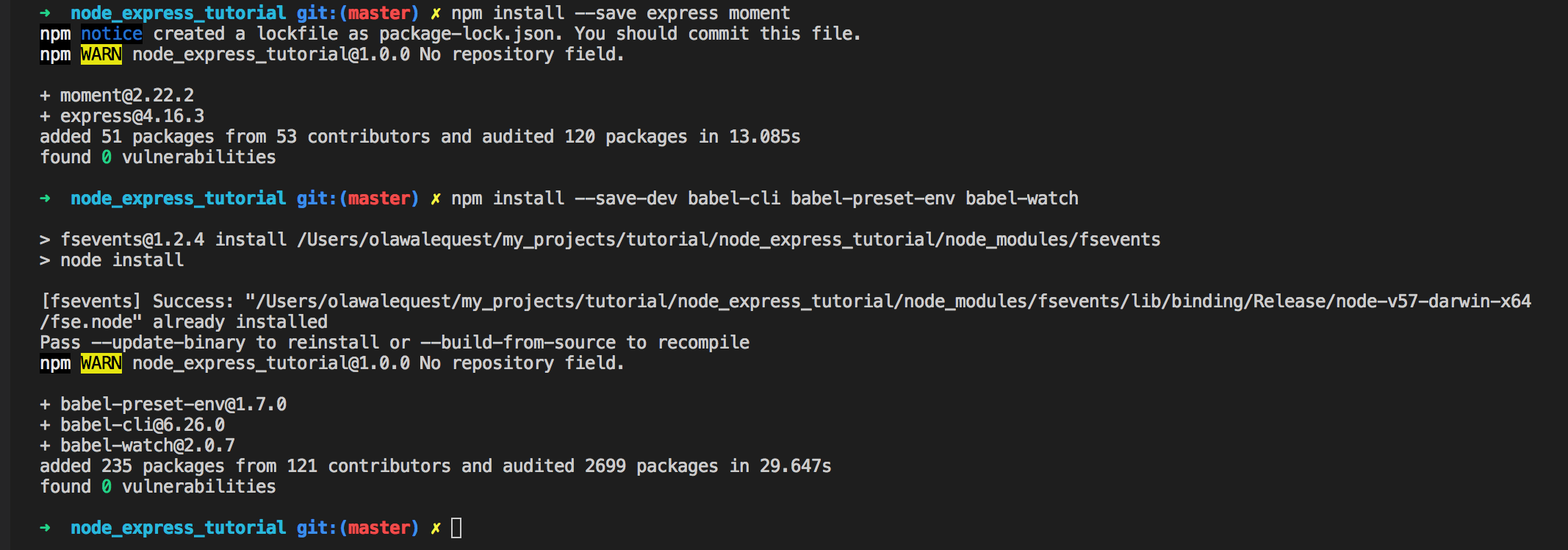
* [expressjs](https://expressjs.com/) - Expressjs is a nodejs web application framework that gives us the ability to create quick and easy APIs.
* [momentjs](https://momentjs.com/) - This gives us the ability to validates and manipulates dates.
* uuid npm package](https://www.npmjs.com/package/uuid)
* [babeljs](https://babeljs.io/setup#installation) - Since we'll be writing all our JavaScript code using ES6, babeljs will help in converting our ES6 codes to ES5.
* [babel watch](https://www.npmjs.com/package/babel-watch) - This is needed for development. One thing that babel watch package does is to compile our code and reload the server each time we make changes to our code.

Run the following command to install all the above packages

$ npm **install** --save express moment uuid

$ npm **install** --save-dev **babel-cli babel-preset-env babel-watch**

If all went well, you should see something similar to this



and your package.json file should contain the following

{

"name": "node\_express\_tutorial",

"version": "1.0.0",

"description": "node express tutorial",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "Olawale Aladeusi",

"license": "ISC",

"dependencies": {

"express": "^4.16.3",

"moment": "^2.22.2"

},

"devDependencies": {

"babel-cli": "^6.26.0",

"babel-preset-env": "^1.7.0",

"babel-watch": "^2.0.7"

}

}

You'll notice that express and moment is under dependencies, that is because those are needed by the time we deploy our code to production. babel-cli, babel-preset-env and babel-watch are only needed during development.  
You should also see a new folder called node\_modules in the project root directory - This folder contains the source code of the package we downloaded.

**Project Structure**

Set up your project structure using the following format

-node\_express\_tutorial

|-package.json

|-.babelrc

|-server.js

|-src

|-controllers

|-Reflection.js

|-models

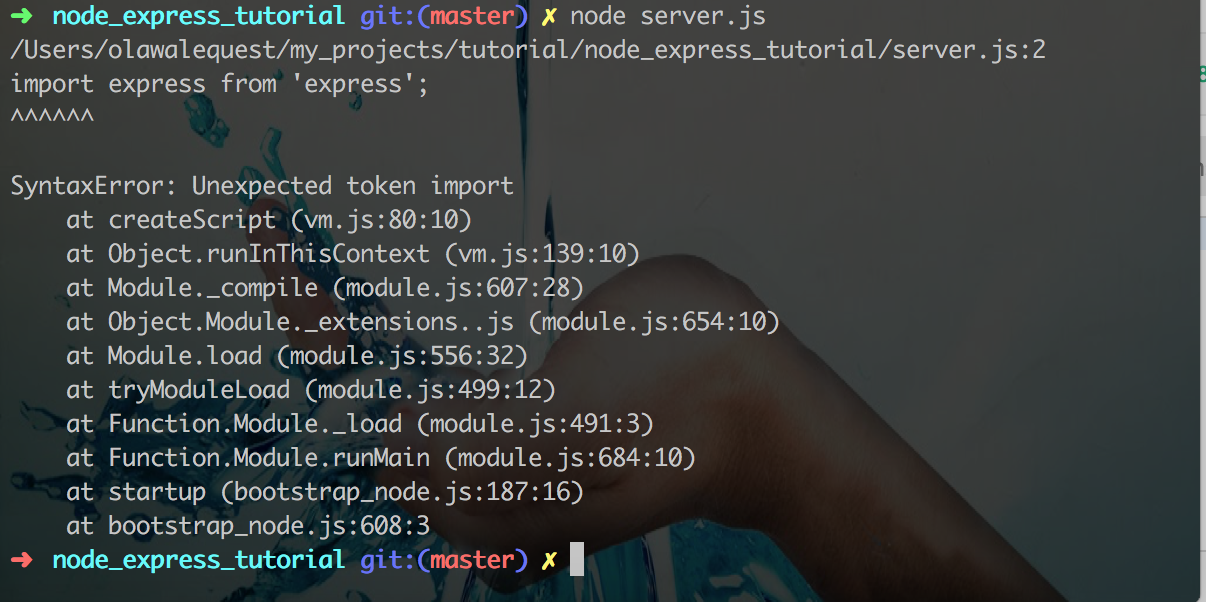
|-Reflection.js

**Server SetUp**

Add the following code to server.js

From the code above, we imported express and set up a new express instance const app = express(). We set up a new express.json() middleware - this is needed to get access to request body. Lastly, we set up a sample endpoint to test if the server is working.  
Note: express.json() is only available in express version 4.16.0 and above

If you run the server using node server.js command, you'll get an error



Don't worry, you're getting the error because nodejs runtime cannot understand some of the ES6 features we used - e.g import. In the next section we will set up babel - babel will help us compile the code to ES5 that nodejs runtime can understand.

**Set Up babel**

In the project root directory, create a new file and name it .babelrc and add

{

"presets": ["env"]

}

Next, open package.json and add "build": "babel server.js --out-dir build"under script property.

{

"name": "node\_express\_tutorial",

"version": "1.0.0",

"description": "node express tutorial",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"build": "babel server.js --out-dir build"

},

#################

# existing **code** #

################

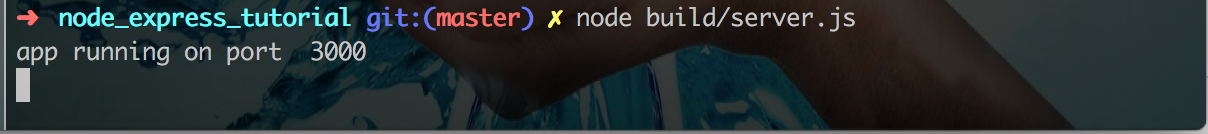
}

Now, run the following command

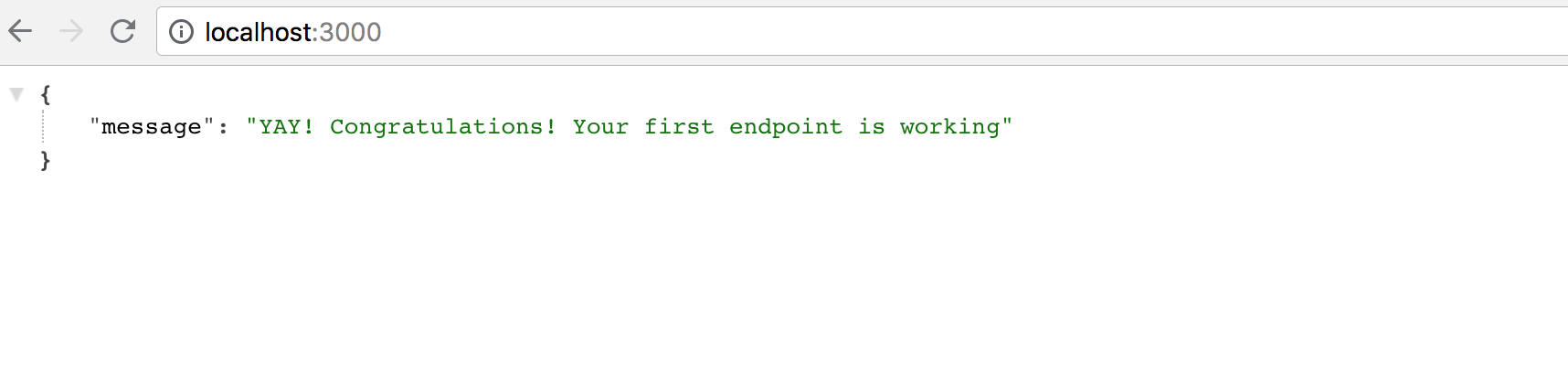
$ npm **run** build

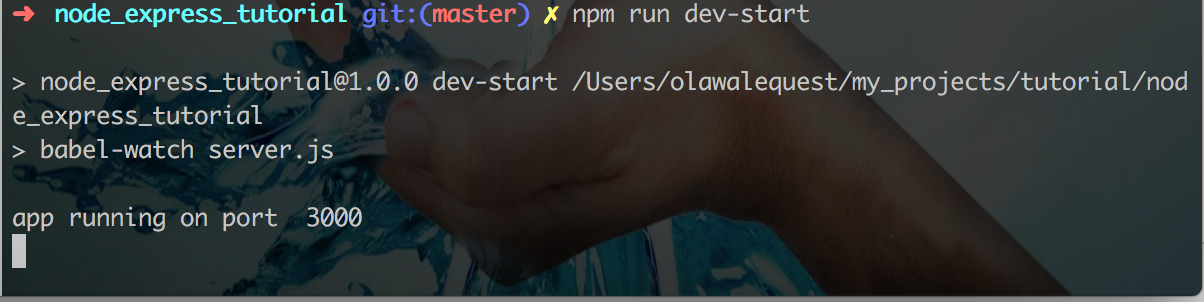
If the above command was successful you will notice that a new build folder has been created in the project directory. Open it and click on server.js - notice the original code has been compiled down to es5. So, running the server.js inside build folder should successfully run your server without error.

$ **node** **build**/server.js



To test if our sample endpoint is working, navigate to http://localhost:3000/ on the browser.



One last thing we need to do in this section is setup babel-watch. Currently, if you make any changes to server.js, you'll need to run a new build and restart the server before your changes will reflect. Going through the build process and restarting server does not make sense for development purpose. This is where babel-watch came to the rescue, as the name implies, babel-watch watch over any changes you made in your code and automatically compile and restart your server.  
Add "dev-start": "babel-watch server.js" to your package.json under script.  
Running npm run dev-start should start your server  


**Set up Reflection Model**

To make things simple, we'll not make use of database for our project. Instead we'll use JavaScript Objects to persist user's data.  
Open src/models/Reflection.js and add the following code

We created Reflection class with five methods create() - to create a new reflection, findOne() - find one reflection, findAll() - find all reflections, update()- update a reflection and delete() - delete a reflection. Each reflection object will contains six propeties id, success, lowPoint, takeAway, createdDate and modifiedDate. We use uuid.v4() to set the default reflection id, moment to set default values for createdDate and modifiedDate.

**Set up Controller and Endpoints**

The following endpoints will be defined

* Create a Reflection - POST /api/v1/reflections
* Get All Reflections - GET /api/v1/reflections
* Get A Reflection - GET /api/v1/reflections/:id
* Update A Reflection - PUT /api/v1/reflections/:id
* Delete A Reflection - DELETE /api/v1/reflections/:id

But first, let's set up our reflection controller, to do that add the following code to src/controllers/Reflection

We created Reflection object that contains create - to create a new reflection, getAll - get all reflections, getOne - get one reflection, update - update a reflection and delete - delete a reflection properties.

Finally, let's defined our endpoints, open server.js and add the following code

**import** Reflection from './src/controllers/Reflection';

app.post('/api/v1/reflections', Reflection.create);

app.get('/api/v1/reflections', Reflection.getAll);

app.get('/api/v1/reflections/:id', Reflection.getOne);

app.put('/api/v1/reflections/:id', Reflection.update);

app.**delete**('/api/v1/reflections/:id', Reflection.**delete**);

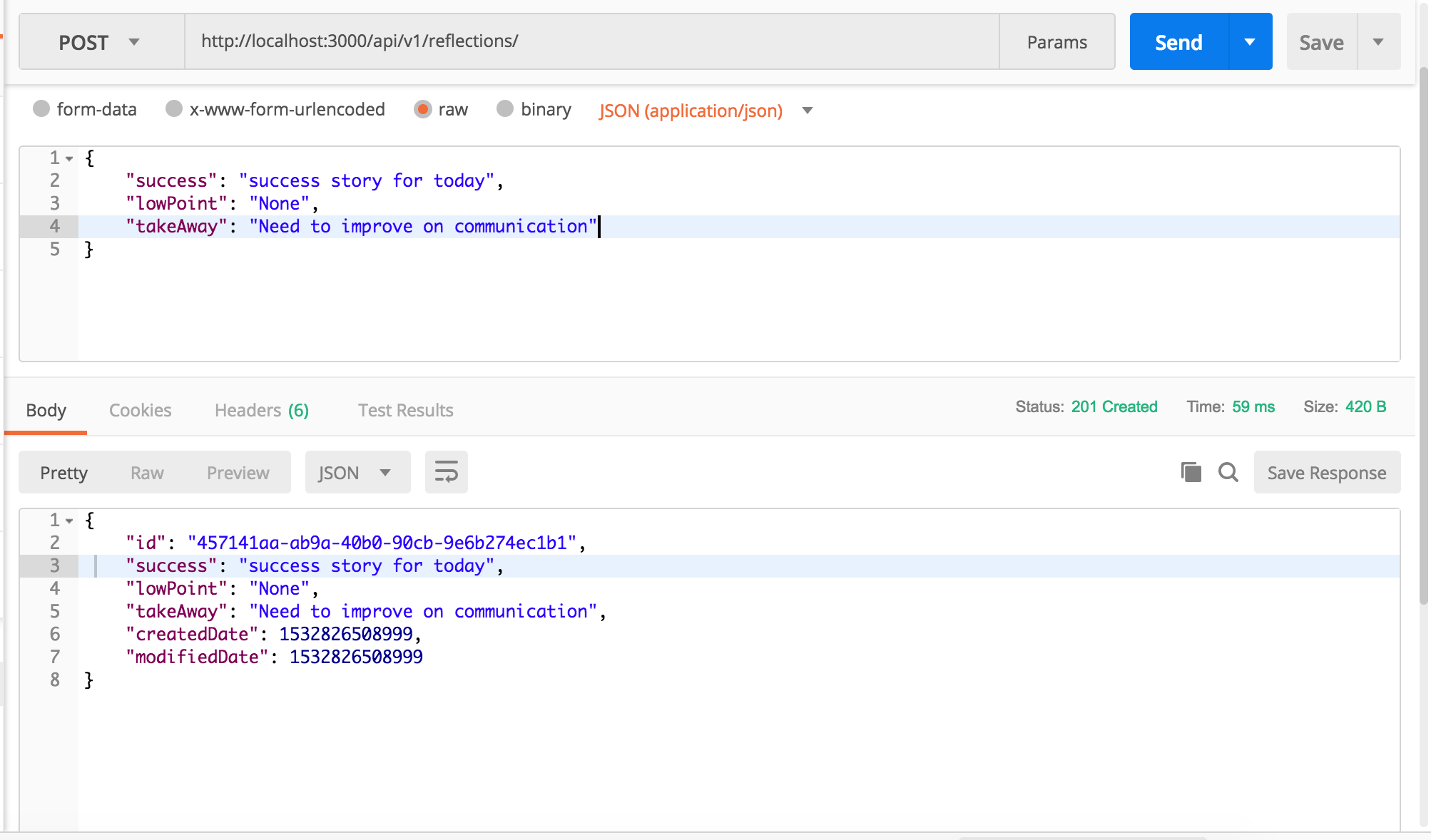
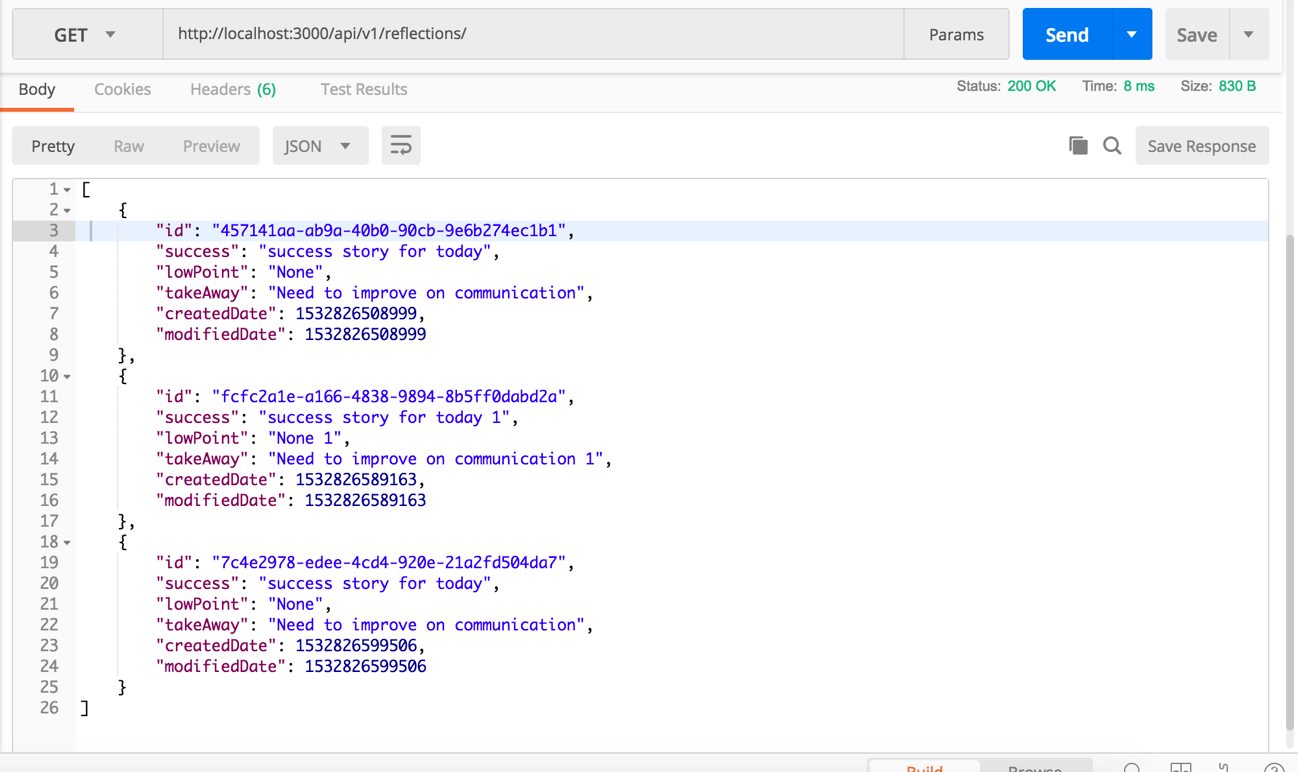
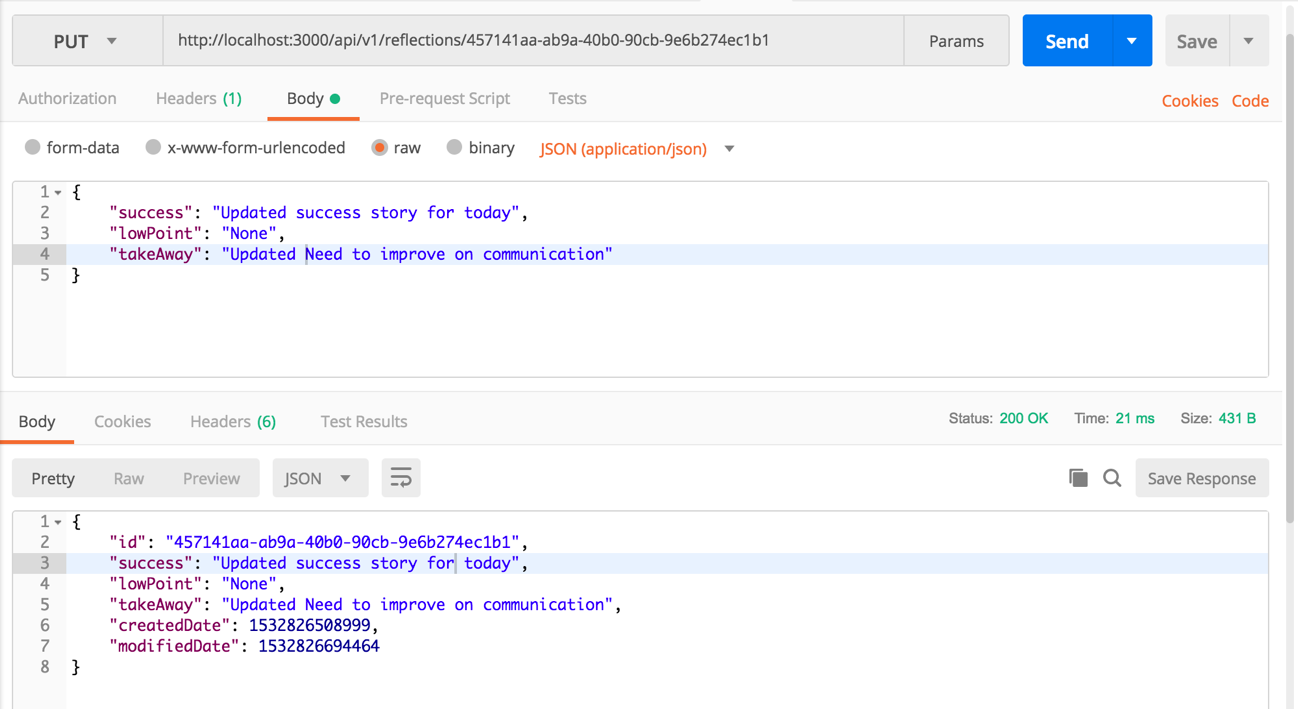
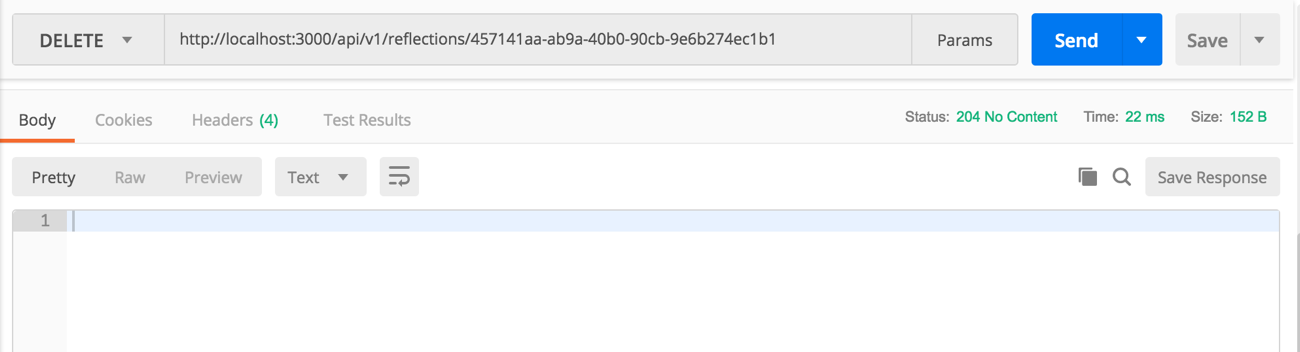
The above defines all our five endpoints, each connecting to it's respective controller method.

**Test on**[**POSTMAN**](https://www.getpostman.com/)

Use the following command to run the server

$ npm **run** dev-start

You can use POSTMAN to test all our endpoints

* Create A Reflection  
  
* Get All Reflections  
  
* Get A Reflection  
  
* Update A Reflection  
  
* Delete A Reflection  
  

**Conclusion**

In this tutorial, we've learned how to setup node expressjs server, how to setup models using object and how to create basic endpoints.

Drop your questions and comments. Don't forget to like this post if you learned one or two things.