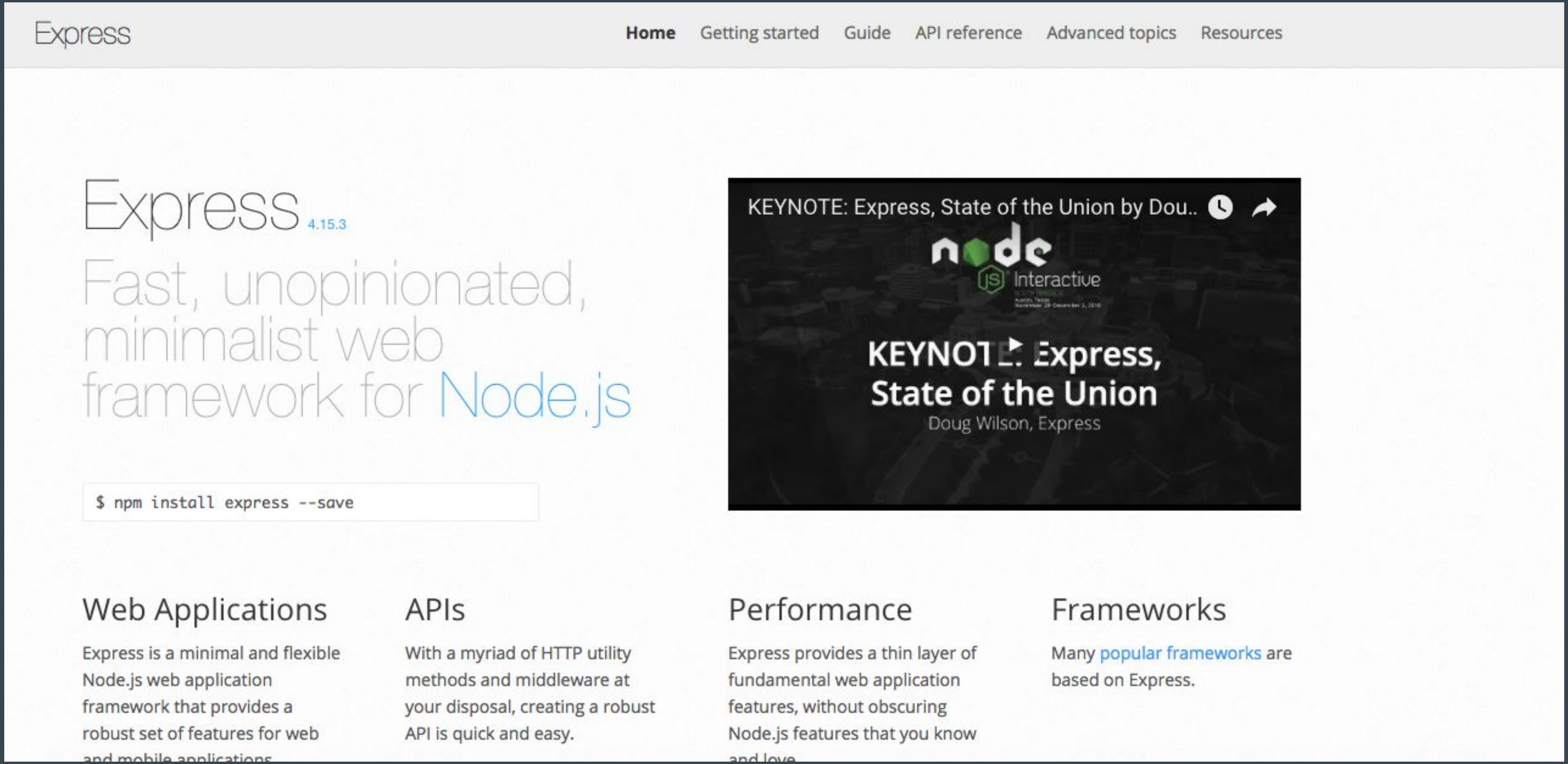
Asia Developer Academy

Creating a RESTful API with MongoDB and Express

Step 1: Install the following modules:

1) express
2) mongoose
3) body-parser

express


Express

Home Getting started Guide API reference Advanced topics Resources

Express 4.15.3

Fast, unopinionated, minimalist web framework for Node.js

```
$ npm install express --save
```

KEYNOTE: Express, State of the Union by Dou..
node Interactive
KEYNOTE: Express, State of the Union
Doug Wilson, Express

Web Applications

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications

APIs

With a myriad of HTTP utility methods and middleware at your disposal, creating a robust API is quick and easy.

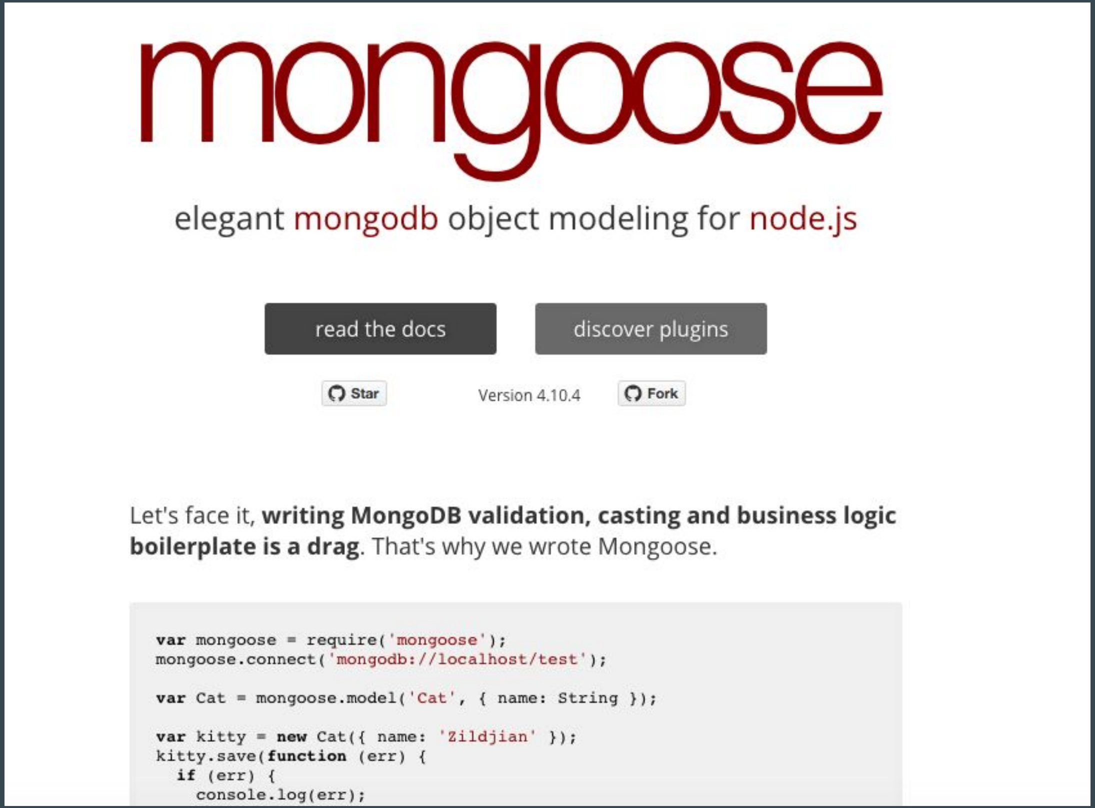
Performance

Express provides a thin layer of fundamental web application features, without obscuring Node.js features that you know and love.

Frameworks

Many popular frameworks are based on Express.

mongoose

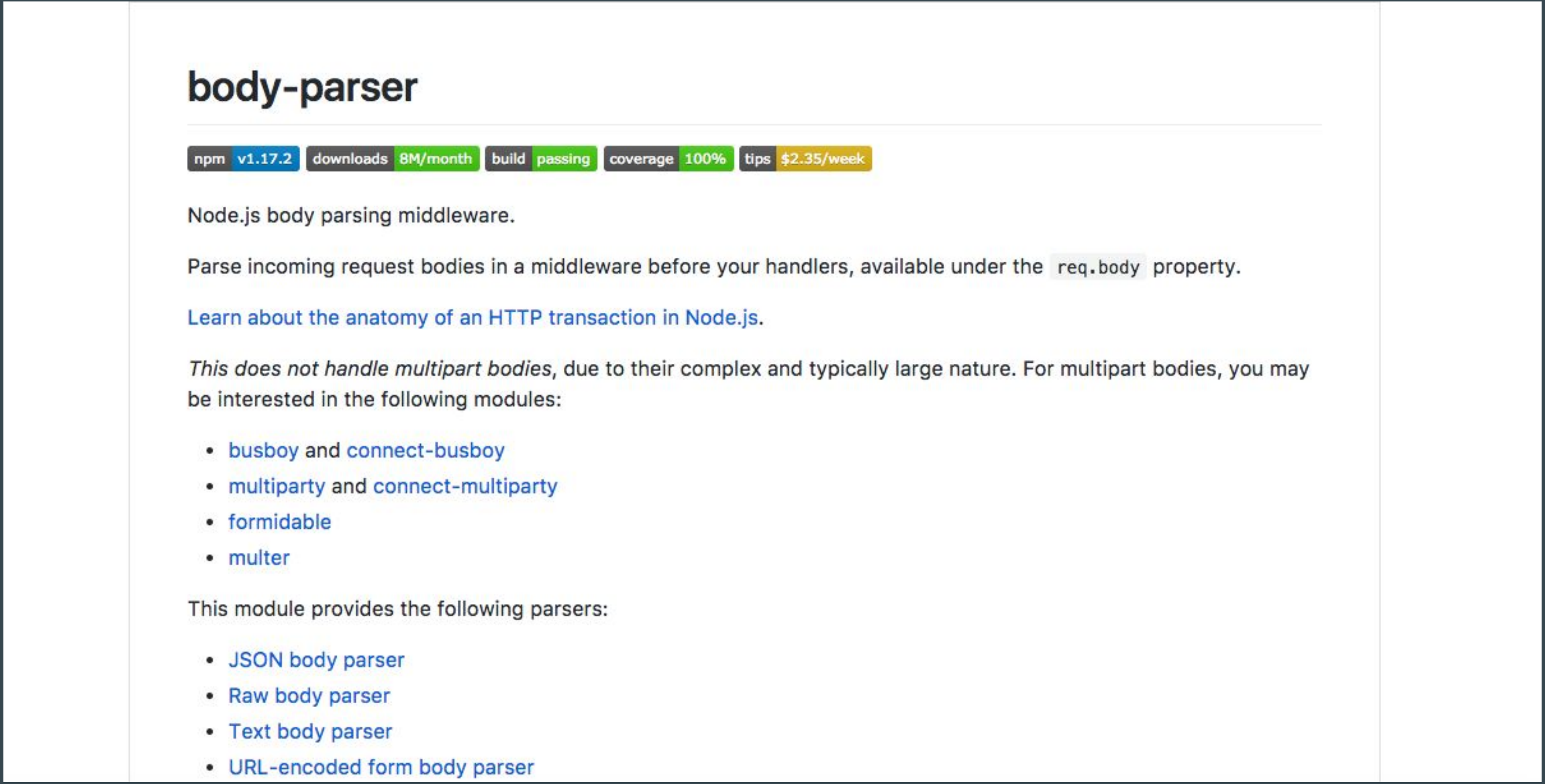


elegant mongodb object modeling for node.js

Let's face it, **writing MongoDB validation, casting and business logic boilerplate is a drag**. That's why we wrote Mongoose.

```
var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/test');

var Cat = mongoose.model('Cat', { name: String });

var kitty = new Cat({ name: 'Zildjian' });
kitty.save(function (err) {
  if (err) {
    console.log(err);
```

body-parser



body-parser

npm v1.17.2 downloads 8M/month build passing coverage 100% tips $2.35/week

Node.js body parsing middleware.

Parse incoming request bodies in a middleware before your handlers, available under the `req.body` property.

Learn about the anatomy of an HTTP transaction in Node.js.

This does not handle multipart bodies, due to their complex and typically large nature. For multipart bodies, you may be interested in the following modules:

- busboy and connect-busboy
- multiparty and connect-multiparty
- formidable
- multer

This module provides the following parsers:

- JSON body parser
- Raw body parser
- Text body parser
- URL-encoded form body parser

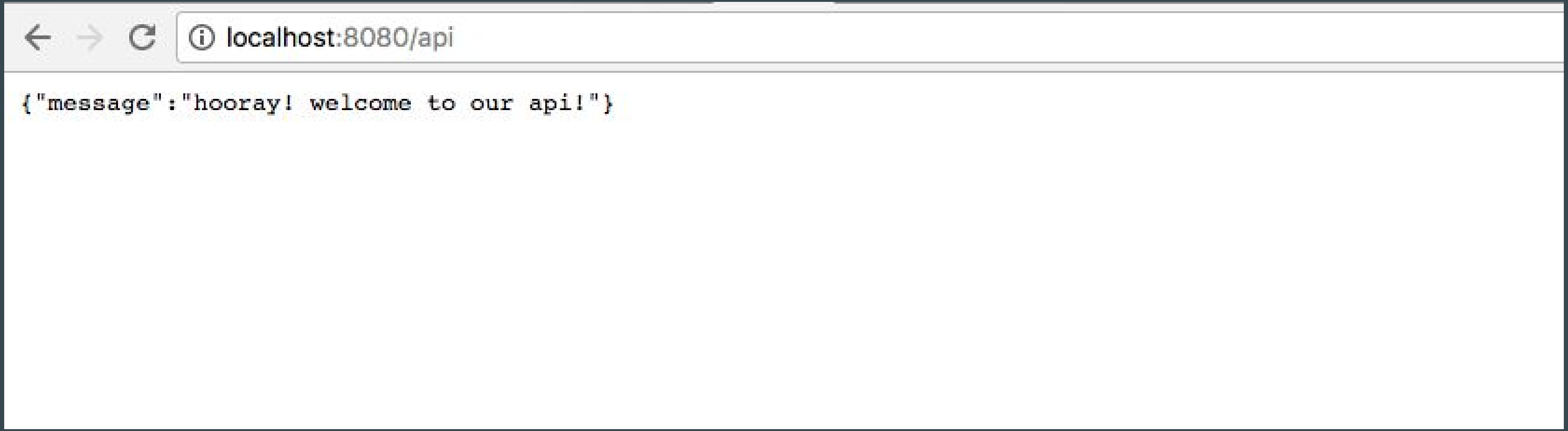
2. Create your server.js file

```
var express    = require('express');        // call express
var app        = express();                 // define our app using express
var bodyParser = require('body-parser');

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

var port = process.env.PORT || 8080;        // set our port
```

3. Set up the route to our API

```
var router = express.Router();


router.get('/', function(req, res) {
    res.json({ message: 'hooray! welcome to our api!' });
});


app.use('/api', router);


app.listen(port);
console.log('Magic happens on port ' + port);
```

4. Test the route in browser


localhost:8080/api

```
{"message":"hooray! welcome to our api!"}
```

5. Connect to Mongodb database using mongoose

```
var mongoose   = require('mongoose');

mongoose.connect('<mongodb address from mlab>')
```

6. Create the Object Schema of our Model.

```
var mongoose      = require('mongoose');
var Schema        = mongoose.Schema;

var PlaceSchema    = new Schema({
    name: String,
    description: String,
    country: String,
    categories: [],
    createdAt : {type: Date, default: Date.now}
});

module.exports = mongoose.model('Place', PlaceSchema);
```

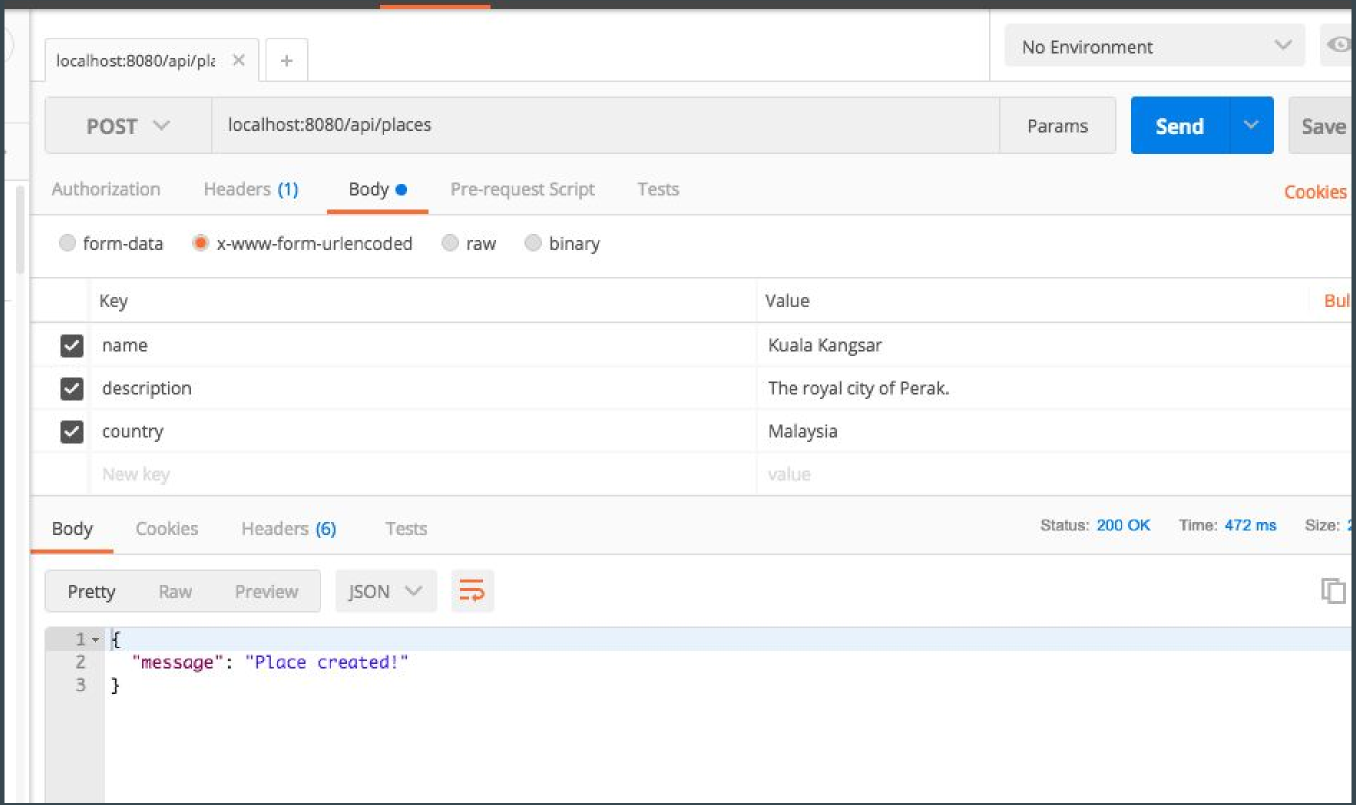
7. Import the schema inside server.js

```
var Place      = require('./app/models/place);
```

8. Create route to Create new place.

```
router.route('/places')

    .post(function(req, res) {

        var place = new Place();
        place.name = req.body.name;
        place.description = req.body.description;
        place.country = req.body.country;

        place.save(function(err) {
            if (err)
                res.send(err);

            res.json({ message: 'Place created!' });
        });

    });
```

9. Test the API on Postman



localhost:8080/api/pl

No Environment

POST localhost:8080/api/places

Params Send Save

Authorization Headers (1) Body ● Pre-request Script Tests Cookies

form-data [✓ x-www-form-urlencoded] raw binary

Key	Value
✓ name	Kuala Kangsar
✓ description	The royal city of Perak.
✓ country	Malaysia
New key	value

Body Cookies Headers (6) Tests — Status: 200 OK Time: 472 ms Size:

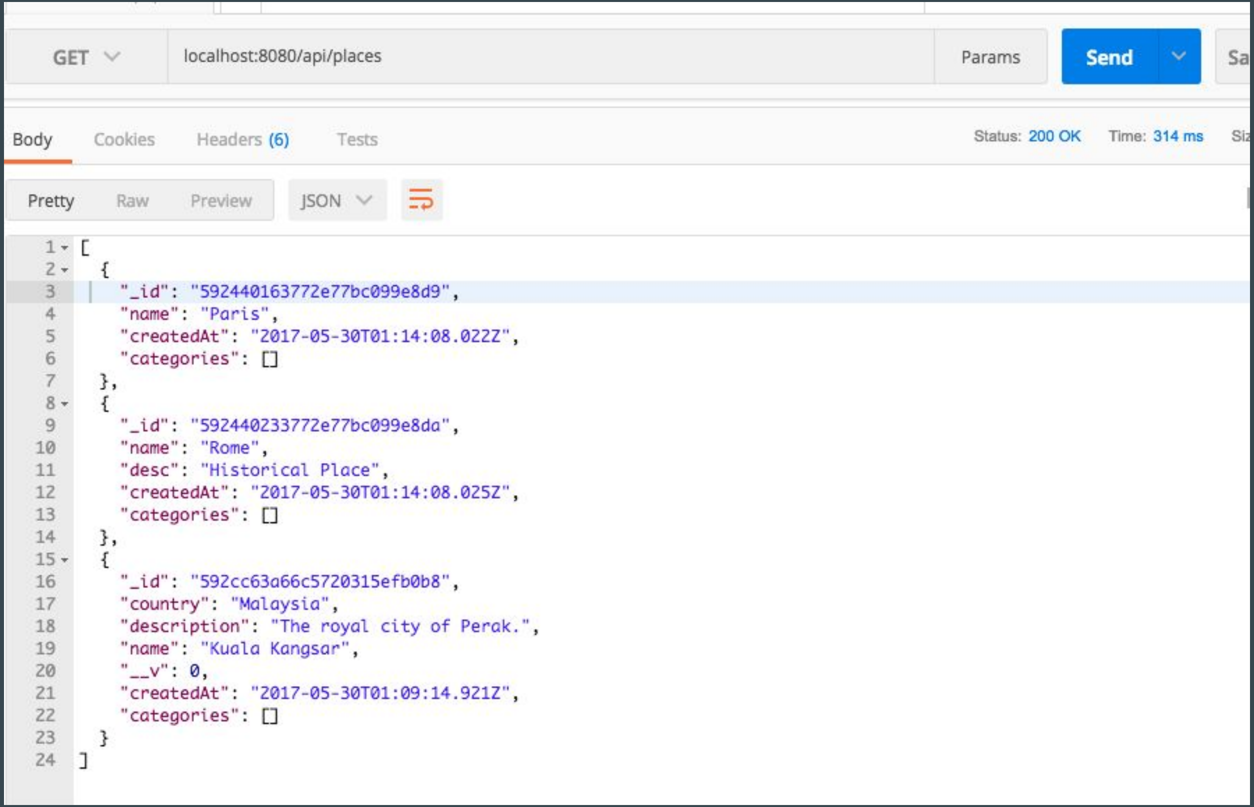
Pretty Raw Preview JSON

```
{
  "message": "Place created!"
}
```

10. Create route to GET place

```
.get(function(req, res) {
        Place.find(function(err, places) {
            if (err)
                res.send(err);

            res.json(places);
        });
    });
```

11. Test the request in POSTMAN



localhost:8080/api/places

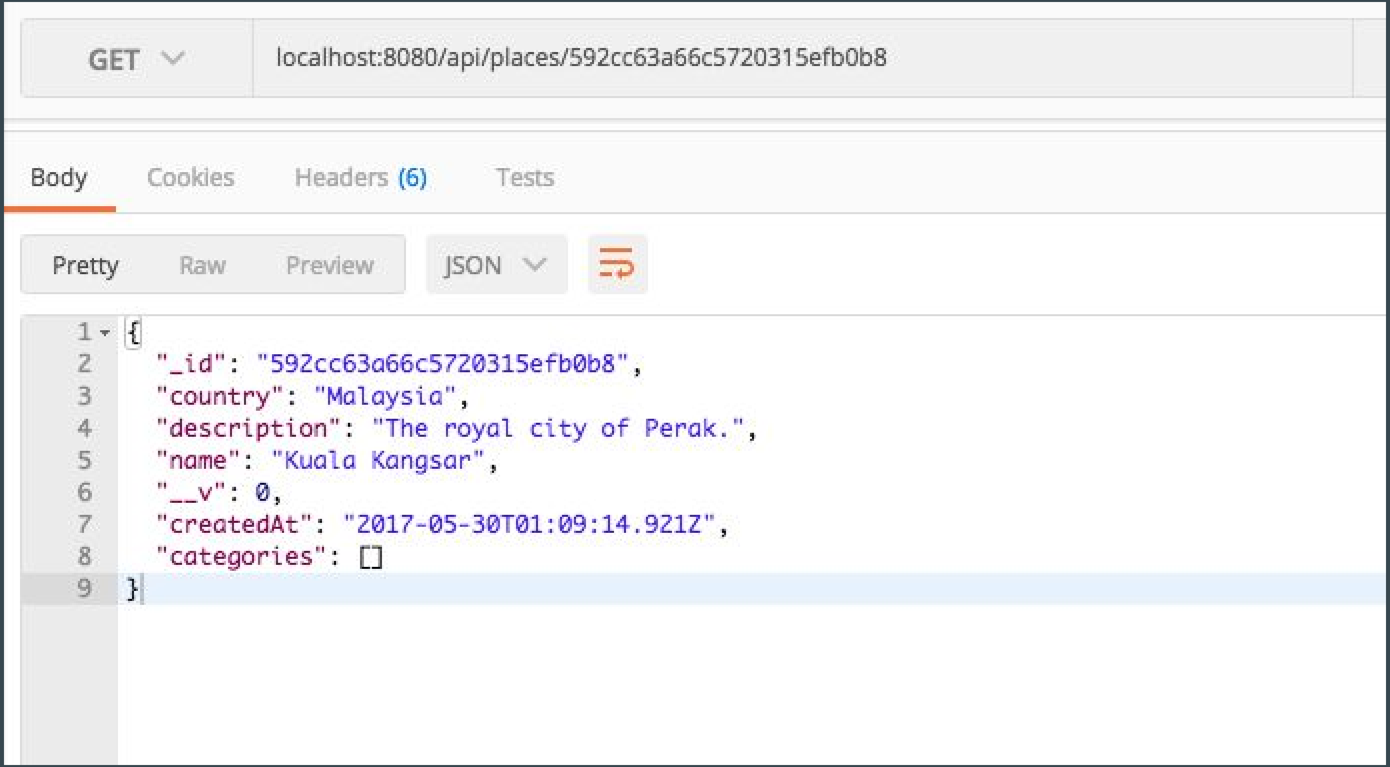
Status: 200 OK Time: 314 ms

```
[
  {
    "_id": "592440163772e77bc099e8d9",
    "name": "Paris",
    "createdAt": "2017-05-30T01:14:08.022Z",
    "categories": []
  },
  {
    "_id": "592440233772e77bc099e8da",
    "name": "Rome",
    "desc": "Historical Place",
    "createdAt": "2017-05-30T01:14:08.025Z",
    "categories": []
  },
  {
    "_id": "592cc63a66c5720315efb0b8",
    "country": "Malaysia",
    "description": "The royal city of Perak.",
    "name": "Kuala Kangsar",
    "__v": 0,
    "createdAt": "2017-05-30T01:09:14.921Z",
    "categories": []
  }
]
```

12. Create GET by ID function

```
.get(function(req, res) {
       Place.findById(req.params.place_id, function(err, place) {
           if (err)
               res.send(err);
           res.json(place);
       });
   });
```

13. Test the request in POSTMAN



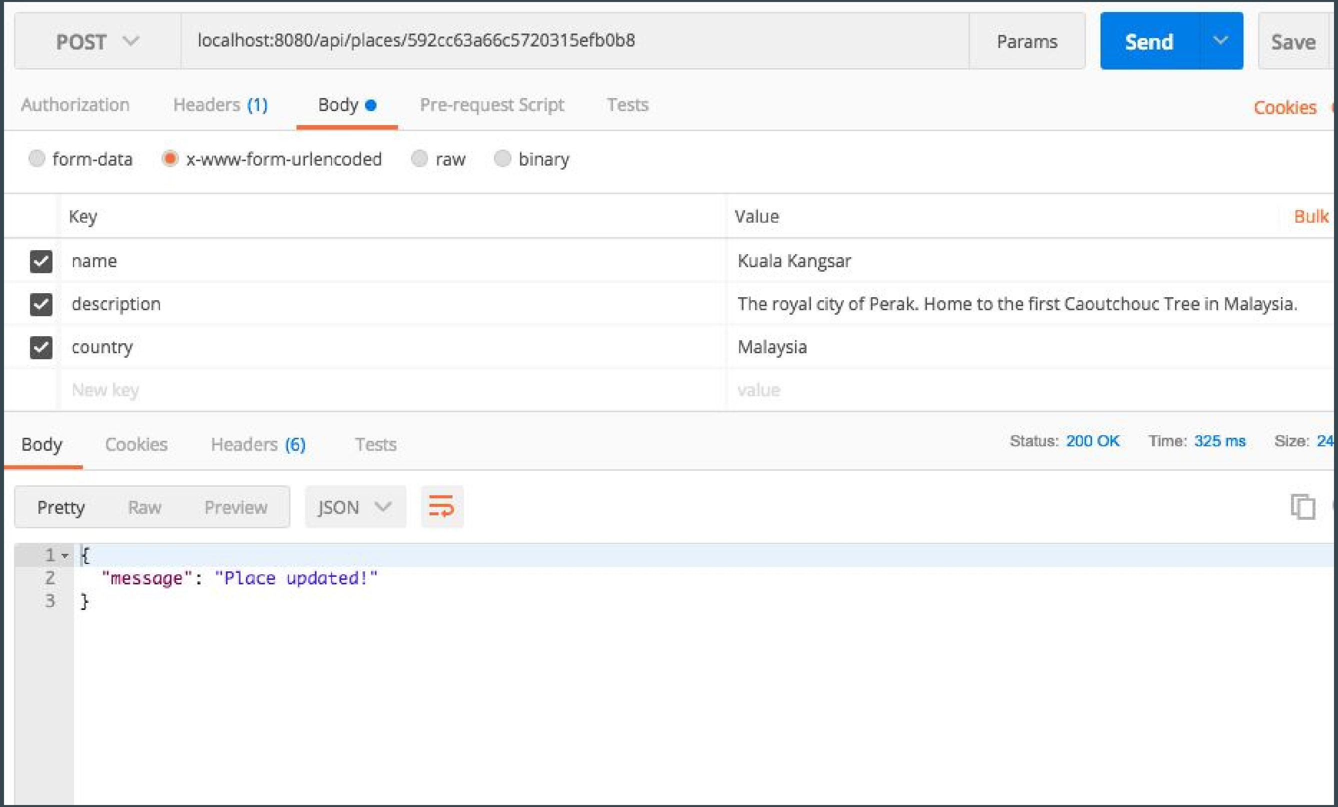
```
localhost:8080/api/places/592cc63a66c5720315efb0b8
```

```
{
  "_id": "592cc63a66c5720315efb0b8",
  "country": "Malaysia",
  "description": "The royal city of Perak.",
  "name": "Kuala Kangsar",
  "__v": 0,
  "createdAt": "2017-05-30T01:09:14.921Z",
  "categories": []
}
```

14. Update the places using POST.

```
.post(function(req, res) {

        Place.findById(req.params.place_id, function(err, place) {

            if (err)
                res.send(err);

            place.name = req.body.name;
            place.description = req.body.description;
            place.country = req.body.country;

            // save the place
            place.save(function(err) {
                if (err)
                    res.send(err);

                res.json({ message: 'Place updated!' });
            });

        });
    });
```

15. Test Update on POSTMAN



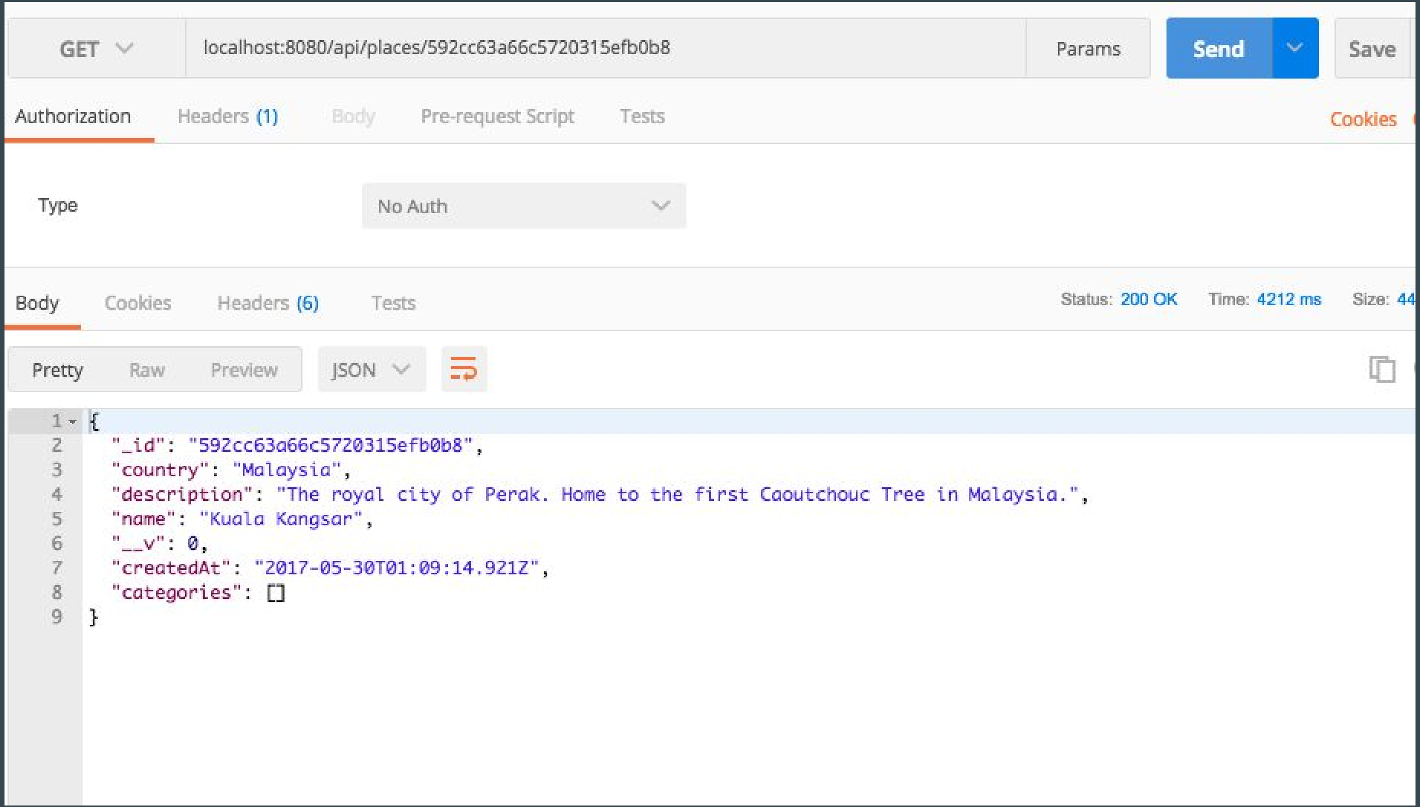
localhost:8080/api/places/592cc63a66c5720315efb0b8

Key	Value
name	Kuala Kangsar
description	The royal city of Perak. Home to the first Caoutchouc Tree in Malaysia.
country	Malaysia
New key	value

Status: 200 OK Time: 325 ms Size: 24

```
{
  "message": "Place updated!"
}
```

16. Retrieve updated content in POSTMAN.



localhost:8080/api/places/592cc63a66c5720315efb0b8

Type: No Auth

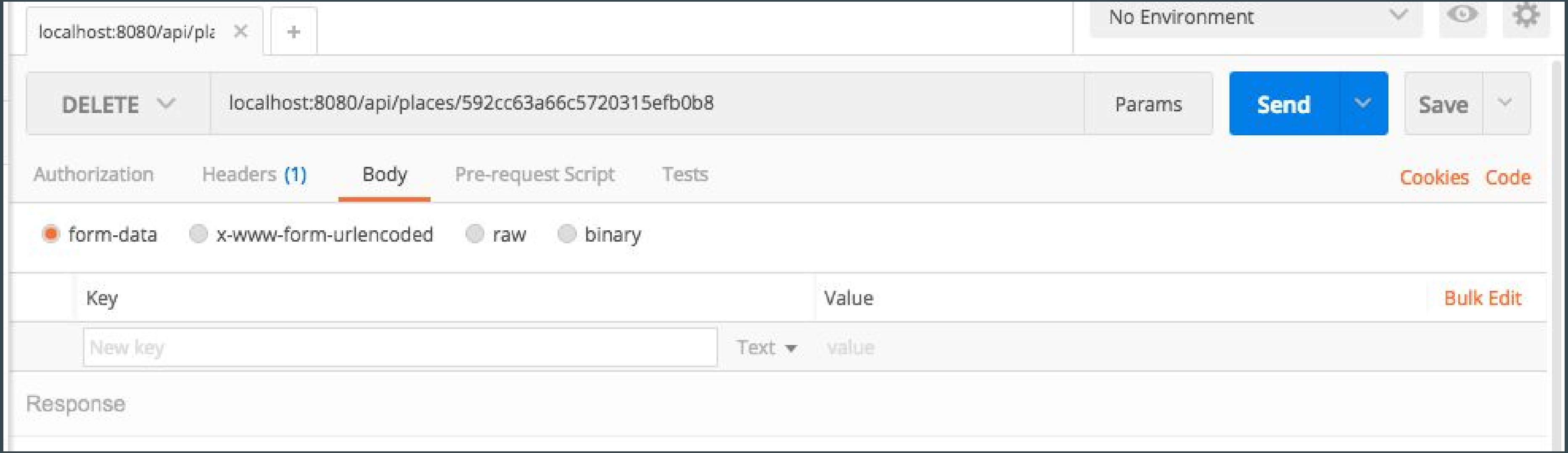
Status: 200 OK Time: 4212 ms Size: 44

```
{
  "_id": "592cc63a66c5720315efb0b8",
  "country": "Malaysia",
  "description": "The royal city of Perak. Home to the first Caoutchouc Tree in Malaysia.",
  "name": "Kuala Kangsar",
  "__v": 0,
  "createdAt": "2017-05-30T01:09:14.921Z",
  "categories": []
}
```

17. Add Delete function.

```
.delete(function(req, res) {
        Place.remove({
            _id: req.params.place_id
        }, function(err, place) {
            if (err)
                res.send(err);

            res.json({ message: 'Successfully deleted' });
        });
    });
```

18. Test on POSTMAN



DELETE localhost:8080/api/places/592cc63a66c5720315efb0b8