← Node.js MongoDB User Registration | How To Add Routes and Models To Node Rest API →

Information Expert Principle Applied To Mongoose Models

```
userSchema.methods
    .generateAuthToken = function () {

    };
```

In programming the Information expert, or the expert principle, is an approach used to determine where to delegate responsibilities. In other words, where should you place the code that completes specific tasks. The Information expert principle will help a developer to place the responsibility in the class with the most information required to fulfill it. In this tutorial we are going to clean up the process of generating JSON Web Tokens to make our code more clear and easier to maintain.

Removing Duplicate Code

If you've been following along with our user registration tutorial for Node.js, you know that we are generating a JWT in more than one place currently. Both users.js and auth.js are doing a task using cookie cutter code. We should extract that logic so that there is only one place that generates the token. That way, if anything changes in the future, you make your changes in one place, not many. In our case, we can add a method to a Mongoose model to do this.

Adding A Method To A Mongoose Model

Right now, the User model looks like so.

/models/user.js

D3 DOM Selection With D3 JavaScript
React useState Hook
ES6 Class Tutorial
WordPress Theme Development Tutorial Step By Step
Laravel Repository Pattern
How Do Linux Permissions Work?
Hunt Down The Nouns
Liskov Substitution Principle
What Are Scalable Vector Graphics?
WordPress Links and Images
Reference Types And Value Types In C#
Laravel Aliases and Contracts

```
const Joi = require('joi');
const mongoose = require('mongoose');

const User = mongoose.model('User', new mongoose.
   name: {
      type: String,
      required: true,
      minlength: 5,
      maxlength: 50
   },
   email: {
      type: String,
      required: true,
      minlength: 5,
      maxlength: 255,
      unique: true
   },
   password: {
      type: String,
      required: true,
      minlength: 5,
      maxlength: 1024
   }
}));

function validateUser(user) {
   const schema = {
      name: Joi.string().min(5).max(50).required(),
      email: Joi.string().min(5).max(255).required().e
      password: Joi.string().min(5).max(255).requirec
   };
   return Joi.validate(user, schema);
}

exports.User = User;
exports.validate = validateUser;
```

In the code above, the second argument to the mongoose.model() function is a new instance of mongoose.Schema(). The first thing we need to do is to extract this to it's own constant like so.

[Laravel File Structure](#)

[Higher Order Functions In JavaScript](#)

[Douglas Crockford The Good Parts Examples](#)

[CSS Crash Course Tutorial For Beginners](#)

[4 Useful Collection Types In Python](#)

[What Is Guzzle PHP?](#)

[Open Closed Principle](#)

[Working With HTML Images](#)

[Learn PHP and Programming by using the Eclipse PDT Debugger](#)

[PHP Simple HTML DOM Parser vs FriendsOfPHP Goutte](#)

[Understanding PHP Variable Scope](#)

[Send a Tweet with Laravel](#)

[Dependency Injection for Beginners](#)

[How To Send Email To New Users](#)

[Create React App Tutorial](#)

[Send Email With Laravel](#)

```
const userSchema = new mongoose.Schema({
   name: {
      type: String,
      required: true,
      minlength: 5,
      maxlength: 50
   },
   email: {
      type: String,
      required: true,
      minlength: 5,
      maxlength: 255,
      unique: true
   },
   password: {
      type: String,
      required: true,
      minlength: 5,
      maxlength: 1024
   }
});
```

With that in place, setting up the user is now a simple matter of this line.

```
const User = mongoose.model('User', userSchema);
```

Adding the method to the mongoose model

We can add the method now to the userSchema just like this.

```
userSchema.methods.generateAuthToken = function ()
   const token = jwt.sign({_id: this._id}, config.get('Pr
   return token;
};
```

The user model now needs to work with both the config and jsonwebtoken packages, so we make sure to include those at the top of the file. The rest of the code is pretty self-explanatory with regard to extracting the user schema, adding a new method, and then creating a new user model.

Adding A Form To Submit Thread Replies
Laravel Migration Generator
Laravel Blade Templating
How To Use The Linux Terminal
Angular Styles Vs StyleUrls
How To Create A Custom Menu In WordPress
What Is Twitter Bootstrap?
Creating C# Class Properties And Tests
Refactor The Laravel Regex Tool To Use Repositories and Dependency Injection
Introduction To React.js
How To Filter Via Query Strings
The Top 15 Most Popular JavaScript String Functions
WordPress Widgets Tutorial
Single File Components In VueJS
Global Query Scopes And Pagination
How To Add Search To A WordPress Theme

```javascript
const config = require('config');
const jwt = require('jsonwebtoken');
const Joi = require('joi');
const mongoose = require('mongoose');

// Extract Schema to it's own constant
const userSchema = new mongoose.Schema({
    name: {
        type: String,
        required: true,
        minlength: 5,
        maxlength: 50
    },
    email: {
        type: String,
        required: true,
        minlength: 5,
        maxlength: 255,
        unique: true
    },
    password: {
        type: String,
        required: true,
        minlength: 5,
        maxlength: 1024
    },
});

// Information Expert Principle (add method to model)
userSchema.methods.generateAuthToken = function
    const token = jwt.sign({ _id: this._id }, config.get(
    return token;
};

// Create new user model
const User = mongoose.model('User', userSchema);

function validateUser(user) {
    const schema = {
        name: Joi.string().min(5).max(50).required(),
        email: Joi.string().min(5).max(255).required().e
        password: Joi.string().min(5).max(255).required
    };
    return Joi.validate(user, schema);
}

exports.User = User;
exports.validate = validateUser;
```

- Html Tutorials For Beginners
- Introduction to MySQL
- Axios Powered VueJS Form Component
- What Are PHP Arrays?
- New String Methods In ES6
- PHP String Helper Functions
- Create a Twitter Bootstrap Page
- Twitter Bootstrap Classes
- Crud In Laravel 4
- PHP URL Encode Example
- JavaScript Events Tutorial
- PHP Booleans and Constants Tutorial
- Underscore JS Each Function
- How To Favorite A Model
- Angular Service Dependency Injection
- Install Laravel on Windows
- Applying RESTful Methods to the Reviews Resource
- ES6 let vs var vs const

Now we have a nice method we can use in other places in our code. So now note the changes in auth.js and users.js. We have essentially removed the commented out code and replaced it with a call to user.generateAuthToken().

/routes/auth.js

```
const Joi = require('joi');
const bcrypt = require('bcrypt');
const _ = require('lodash');
const { User } = require('../models/user');
const express = require('express');
const router = express.Router();

router.post('/', async (req, res) => {
    // First Validate The HTTP Request
    const { error } = validate(req.body);
    if (error) {
        return res.status(400).send(error.details[0].mes
    }

    //  Now find the user by their email address
    let user = await User.findOne({ email: req.body.em
    if (!user) {
        return res.status(400).send('Incorrect email or p
    }

    // Then validate the Credentials in MongoDB match
    // those provided in the request
    const validPassword = await bcrypt.compare(req.bo
    if (!validPassword) {
        return res.status(400).send('Incorrect email or p
    }
    // const token = jwt.sign({ _id: user._id }, config.g
    const token = user.generateAuthToken();
    res.send(token);
});

function validate(req) {
    const schema = {
        email: Joi.string().min(5).max(255).required().e
        password: Joi.string().min(5).max(255).requirec
    };

    return Joi.validate(req, schema);
}

module.exports = router;
```

[Render HTML In Node.js](#)

[How To Fix The N+1 Problem](#)

[How To Create A Child Component In VueJS](#)

[Php Tutorials For Beginners](#)

[How To Quickly Test PHP Snippets](#)

[How To Bind Events In AngularJS](#)

[What are MySQL Operators?](#)

[VueJs Parent Child Communication](#)

[How To Remember Form Data](#)

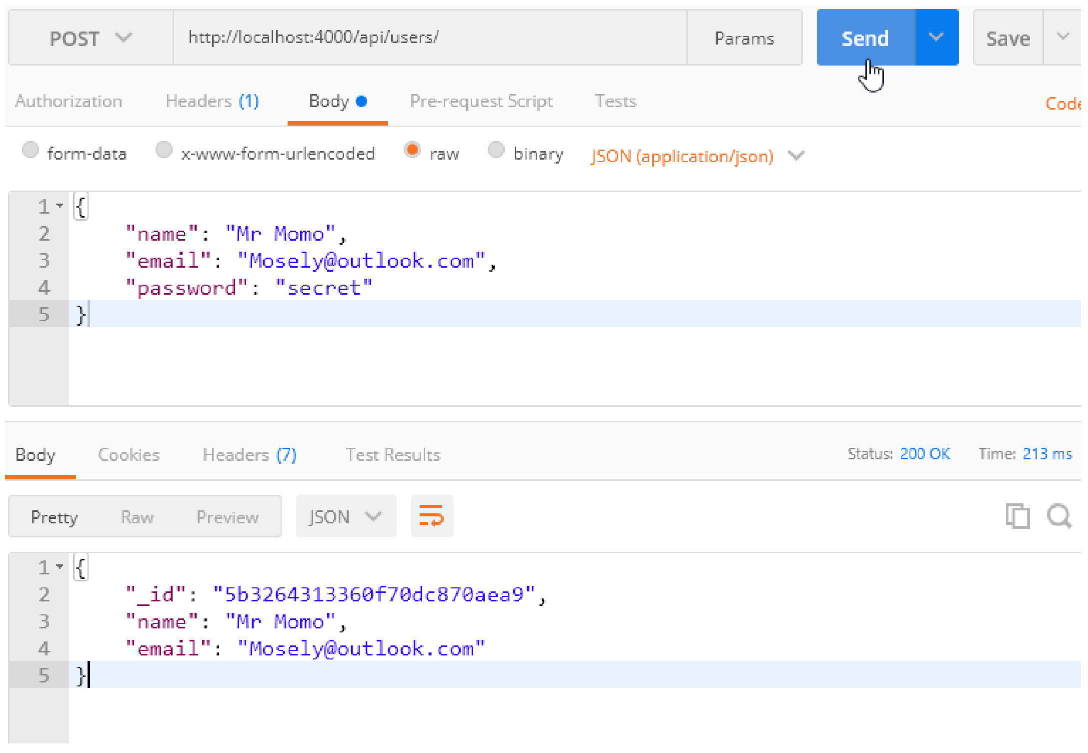
/routes/users.js

```
const bcrypt = require('bcrypt');
const _ = require('lodash');
const { User, validate } = require('../models/user');
const express = require('express');
const router = express.Router();

router.post('/', async (req, res) => {
    // First Validate The Request
    const { error } = validate(req.body);
    if (error) {
        return res.status(400).send(error.details[0].mes
    }

    // Check if this user already exisits
    let user = await User.findOne({ email: req.body.em
    if (user) {
        return res.status(400).send('That user already e
    } else {
        // Insert the new user if they do not exist yet
        user = new User(_.pick(req.body, ['name', 'ema
        const salt = await bcrypt.genSalt(10);
        user.password = await bcrypt.hash(user.passwor
        await user.save();
        // const token = jwt.sign({ _id: user._id }, confi
        const token = user.generateAuthToken();
        res.header('x-auth-token', token).send(_.pick(us
    }
});

module.exports = router;
```

Fantastic!

Testing The Refactor With Postman

That was a nice refactor, but we need to make sure the API still works as intended. Here we test with Postman by sending a new Post request to http://localhost:4000/api/users/ with a JSON object in the body for a new user. Note we get back a proper user object in the response, so this means it worked!



```
{
    "name": "Mr Momo",
    "email": "Mosely@outlook.com",
    "password": "secret"
}
```

```
{
    "_id": "5b3264313360f70dc870aea9",
    "name": "Mr Momo",
    "email": "Mosely@outlook.com"
}
```

Adding a method to a Mongoose Model Summary

In this tutorial we had a quick look at how to add a method to a Mongoose model in order to reduce duplicate code in other areas of our application. You can add as many methods as needed as long as it makes sense and follows the **Information Expert Principle**.

#javascript #nodejs

← Node.js MongoDB User Registration | How To Add Routes and Models To Node Rest API →