



**FPT POLYTECHNIC**



**THỰC HỌC – THỰC NGHIỆP**



**Conceive Design Implement Operate**

# **NODEJS & RESFUL WEB SERVICE**

---

## **MVC NODEJS**

- ⊙ Biết cách tổ chức code với mô hình MVC
- ⊙ Xây dựng được model
- ⊙ Xây dựng tốt controller
- ⊙ Hiểu và cài đặt ORM với thư viện sequelize



- 📖 Mô hình tổ chức code?
- 📖 Mô hình MVC
- 📖 Tổ chức MVC với NodeJS
  - ❖ Tạo cotroller
  - ❖ Tạo model
  - ❖ Xây dựng hệ thống router
- 📖 Cài đặt và sử dụng thư viện sequelize

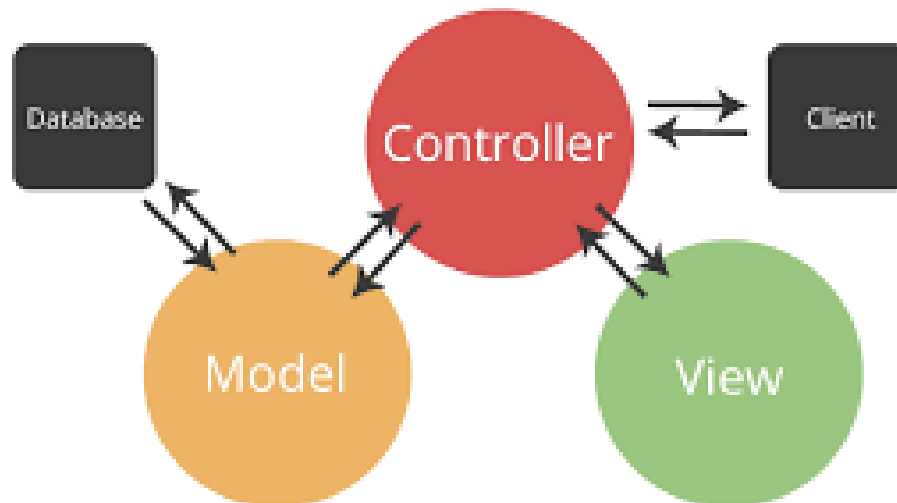




# PHẦN 1: TỔ CHỨC CODE MỚI MÔ HÌNH MVC TRONG NODEJS

- ❑ Là cấu trúc, cách tổ chức code cho ứng dụng, website
- ❑ Có rất nhiều mô hình lập trình
  - ❖ MVC
  - ❖ MVVM
  - ❖ MVP
  - ❖ ...

- ❑ **MVC** (MVC Design Pattern) là viết tắt của **Model — View — Controller**.
- ❑ Là **mô hình lập trình** phổ biến được sử dụng để tạo cấu trúc cho nhiều trang web, ứng dụng tiên tiến.



# MÔ HÌNH MVC

Routes

Model

View

Controller

Tầng dữ liệu cho ứng dụng

Là những gì mà người dùng thấy

Kết nối giữa view với model

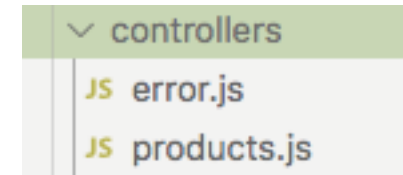
Làm việc với dữ liệu như thêm, xóa, đọc

Tách biệt với code của ứng dụng

Ràng buộc về logic của ứng dụng

Tạo thư mục Controllers trong project  
(products và error)

Controller products bổ sung các phương  
thức

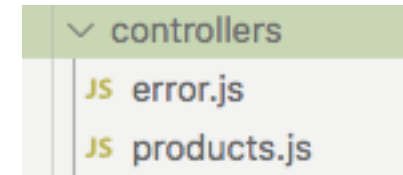


```
const products = [];  
exports.getAddProduct = (req, res, next) => {  
  res.render('add-product', {  
    pageTitle: 'Add Product',  
    path: '/admin/add-product',  
    activeAddProduct: true  
  });  
};
```



Tạo thư mục Controllers trong project  
(products và error)

Controller products bổ sung các phương  
thức



```
exports.getProducts = (req, res, next) => {  
  res.render('shop', {  
    prods: products,  
    pageTitle: 'Shop',  
    path: '/',  
    hasProducts: products.length > 0,  
    activeShop: true,  
  });  
};
```

Tạo các route trong thư mục routes để nhận các request từ client

Route admin

```

  routes
  JS admin.js
  JS shop.js
  
```

```

const productsController = require('../controllers/products');

const router = express.Router();

// /admin/add-product => GET
router.get('/add-product', productsController.getAddProduct);

// /admin/add-product => POST
router.post('/add-product', productsController.postAddProduct);

module.exports = router;
  
```

Tạo các route trong thư mục routes để nhận các request từ client

Route shop

```
▼ routes  
JS admin.js  
JS shop.js
```

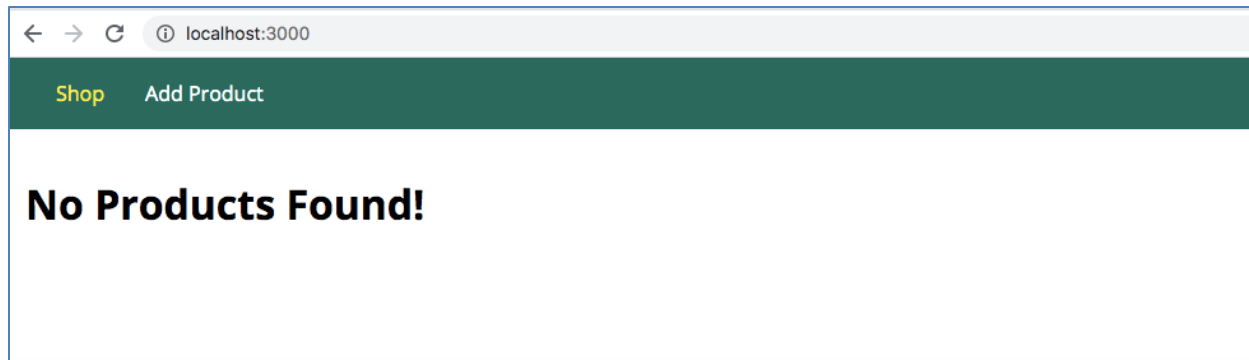
```
const productsController = require('../controllers/products');
```

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

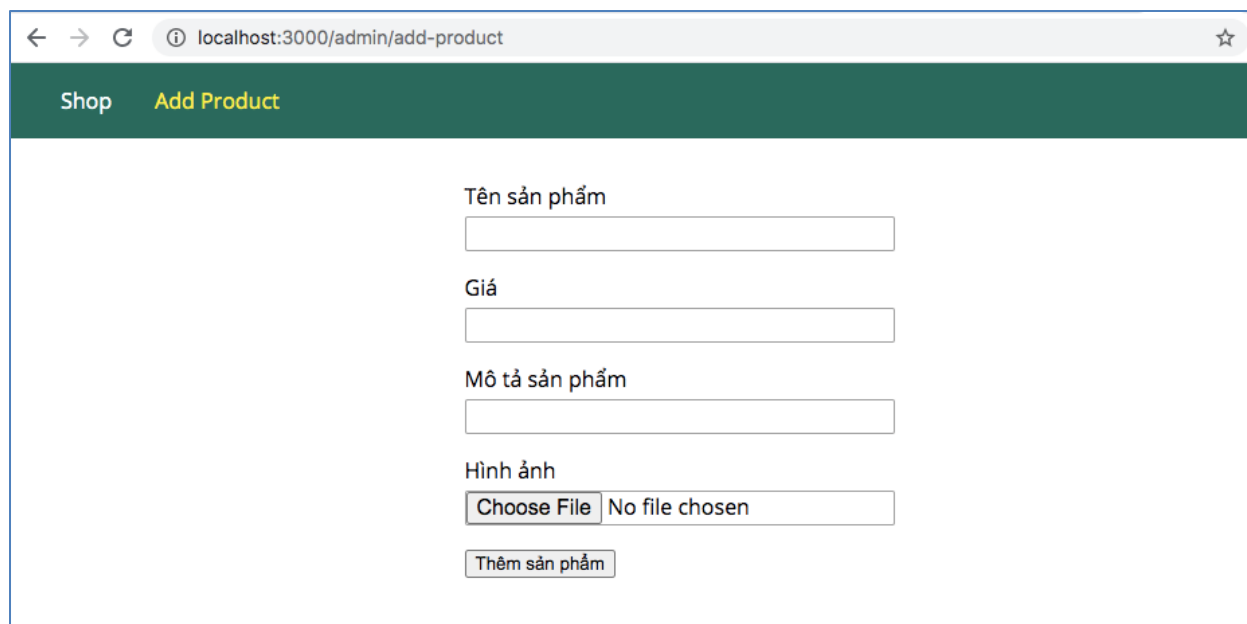
```
router.get('/', productsController.getProducts);
```

```
module.exports = router;
```

## Sản phẩm Demo



A screenshot of a web browser window. The address bar shows 'localhost:3000'. The page has a dark green header with 'Shop' and 'Add Product' links. The main content area displays 'No Products Found!' in bold black text.



A screenshot of a web browser window showing the 'Add Product' form. The address bar shows 'localhost:3000/admin/add-product'. The page has a dark green header with 'Shop' and 'Add Product' links. The form contains the following fields and buttons:

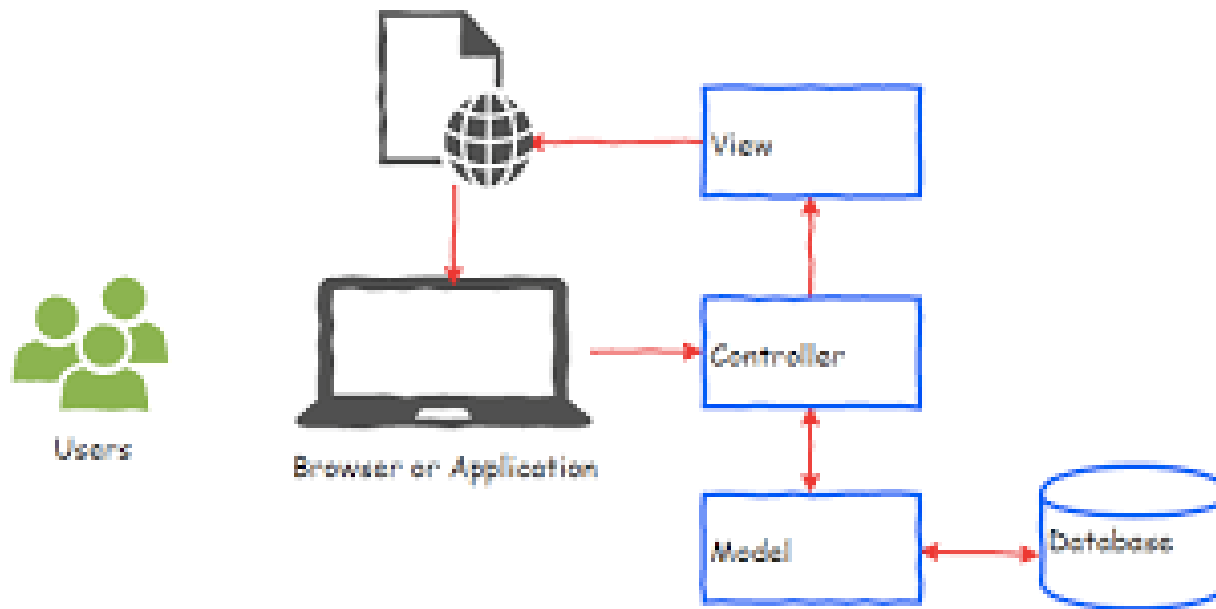
- Tên sản phẩm:
- Giá:
- Mô tả sản phẩm:
- Hình ảnh:  No file chosen
-

demo

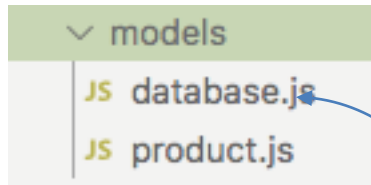


## PHẦN 2: XÂY DỰNG MODEL

Là tầng giao tiếp trực tiếp với cơ sở dữ liệu trong mô hình MVC



Tạo thư mục models, chứa các model tương ứng cho ứng dụng

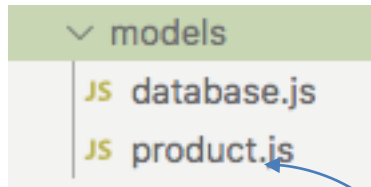


Database trả về kết nối với cơ sở dữ liệu

```
var mysql = require('mysql');
var db = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'book'
});
db.connect(function(err) {
  if (err) throw err;
  console.log('Database is connected successfully !');
});
module.exports = db;
```



Tạo thư mục models, chứa các model tương ứng cho ứng dụng



Lớp product chứa các phương thức tính thao tác trên database

```
var db=require('./database');
var products=[];
module.exports = class Product {
  constructor() {
  }
  static saveProduct() {//thêm một sản phẩm
  }
  static fetchAll() {//trả về tất cả sản phẩm
    let sql = `SELECT * FROM products`;
    db.query(sql, function(err, data) {
      if (err) throw err;
      products=data;
    });
    return products;
  }
}
```

Model được sử dụng bởi controller khi cần

Require('../models/modelname')

```
exports.getProducts = (req, res, next) => {  
  const products = Product.fetchAll();  
  res.render('shop', {  
    prods: products,  
    pageTitle: 'Shop',  
    path: '/',  
    activeShop: true,  
  });  
};
```



## Thêm phương thức lưu trữ dữ liệu trong model

```
//thêm một sản phẩm  
static saveProduct(product) {  
  db.query('insert into products SET ?',product, function(err, data) {  
    if (err) throw err;  
    return true;  
  })  
}
```

Gọi phương thức lưu trữ dữ liệu trong model thông qua controller ( cập nhật lại phương thức postAddProduct)

```
exports.postAddProduct = (req, res, next) => {  
  const file = req.file  
  let title=req.body.productName;  
  let price=req.body.price;  
  let description=req.body.description;  
  let nameImage=file.filename;  
  product={  
    nameProduct:title,  
    priceProduct:price,  
    sortDescription:description,  
    images:nameImage,  
  }  
  Product.saveProduct(product);  
  res.redirect('/');  
};
```

## Thêm phương thức xoá dữ liệu trong model

```
//xoa sản phẩm theo mã  
static delProduct(id) {  
  db.query(`delete from products where productId= ${id}`, function(err, data) {  
    if (err) throw err;  
    return true;  
  })  
}
```

## Controller sử dụng model xóa dữ liệu

```
//xoa sản phẩm theo mã  
exports.delProduct = (req, res, next) => {  
  let prodId=req.params.id;  
  Product.delProduct(prodId);  
  res.redirect('/');  
}
```

# XOÁ DỮ LIỆU TRONG MODEL

Trường hợp có tham số dạng get, ví dụ cho trường hợp xóa

```
/delete product  
router.get('/delete/:id', productsController.delProduct);
```

Trường hợp tham số truyền vào là một chức năng riêng như upload ảnh

```
// /admin/add-product => POST  
router.post('/add-product', upload.single('productImage'), productsController.postAddProduct);
```

- Tại Controller viết hàm trả về thông tin chi tiết theo id
  - id được truyền khi **route**

```
exports.getProduct = (req, res, next) => {  
  const proId = req.params.productId;  
  Product.findById(proId, product => {  
    console.log(product);  
  });  
  res.redirect('/');  
};
```

```
//router  
router.get('/products/:productId', shopController.getProduct);
```



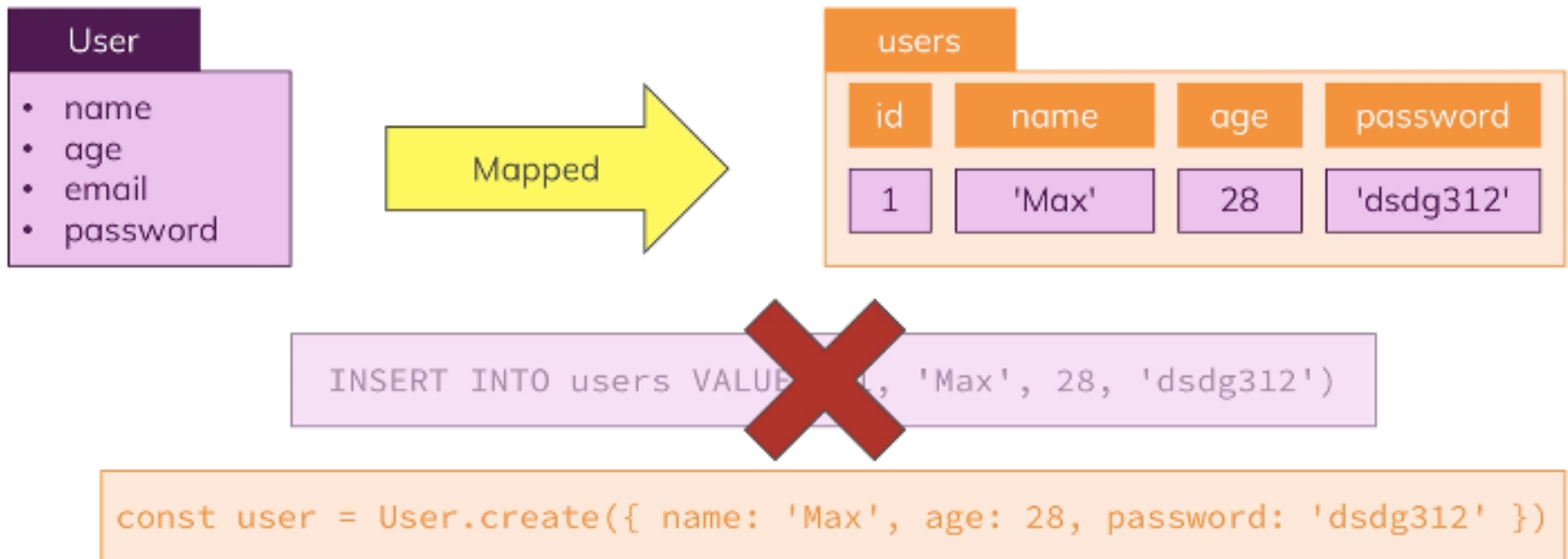
- Tại model viết hàm tìm thông tin theo id
  - id được truyền vào

```
static findById(id, cb) {  
  getProductsFromDatabase(products => {  
    const product = products.find(p => p.id === id);  
    cb(product);  
  });  
}
```

## Sequelize là gì?

Là một ORM dành cho **Node. Js**

An Object – Relational Mapping Library



## Thực hiện

### Cài đặt thư viện sequelize

Npm install sequelize

### Khai báo

```
const Sequelize = require('sequelize');  
const sequelize = new Sequelize('nodeProject', 'root', '', {  
  dialect: 'mysql',  
  host: 'localhost'  
});
```

```
module.exports = sequelize;
```

## Thực hiện ánh xạ - mapping

### Ví dụ ánh xạ với bảng product

```
const Sequelize = require('sequelize');

const sequelize = require('./database');

const Product = sequelize.define('product', {
  id: {
    type: Sequelize.INTEGER,
    autoIncrement: true,
    allowNull: false,
    primaryKey: true
  },
```

```
  title: Sequelize.STRING,
  price: {
    type: Sequelize.DOUBLE,
    allowNull: false
  },
  imageUrl: {
    type: Sequelize.STRING,
    allowNull: false
  },
  description: {
    type: Sequelize.STRING,
    allowNull: false
  }
});

module.exports = Product;
```

## Controller sử dụng model

Import model vào controller

```
const Product = require('../models/product');
```

Sử dụng các phương thức có sẵn của thư viện sequelize như: create, findById, save,...

```
exports.postAddProduct = (req, res, next) => {  
  const title = req.body.title;  
  const imageUrl = req.body.imageUrl;  
  const price = req.body.price;  
  const description = req.body.description;  
  Product.create({title: title, price: price, imageUrl: imageUrl,  
    description: description  
  })  
  .then()  
  .catch();  
};
```

## Đồng bộ các khai báo với database

Tại server khai báo các 2 lệnh sau để đồng bộ

```
const sequelize = require('./database');
```

```
Sequelize.sync()
```

```
.then(result=>{
```

```
app.listen(3000);
```

```
})
```

demo

- ☑ Mô hình tổ chức code?
- ☑ Mô hình MVC
- ☑ Tổ chức MVC với NodeJS
  - ❖ Tạo cotroller
  - ❖ Tạo model
  - ❖ Xây dựng hệ thống router
- ☑ Cài đặt và sử dụng thư viện sequelize





thank  
you!