# เอกสารประกอบการเรียน วิชา 01204271
# วิศวกรรมคอมพิวเตอร์เบื้องต้น
# การทดลองที่ 3 การใช้ EJS Template ใน Express

## วัตถุประสงค์

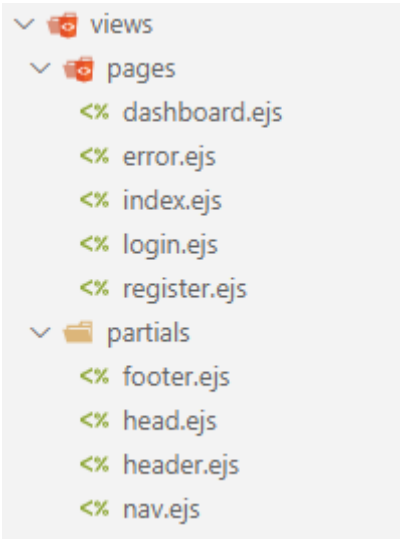
1. สามารถใช้ EJS Template Engine ได้
2. เข้าใจ Method ของ Express Module
3. สามารถกำหนดโครงสร้างไฟล์ Template Layout ได้

## การทดลองที่ 1.1 ติดตั้ง Package

1. เปิดโปรแกรม Visual Studio Code
2. สร้างโฟลเดอร์โปรเจ็คใหม่
3. ใช้คำสั่งต่อไปนี้
   3.1 npm init
   3.2 npm install express
   3.3 npm install ejs
   3.4 npm install @hapi/joi
   3.5 npm install nodemon

## การทดลองที่ 1.2 ใช้ EJS Template Layout ในการสร้างระบบสมาชิก

1. สร้างโฟลเดอร์ใหม่โดยมีโครงสร้างไฟล์ดังนี้




2. เขียนโค้ดในแต่ละหน้าดังนี้

2.1 ไฟล์ head.ejs [views/partials/head.ejs]

```
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1, shrink-to-fit=no">
```

```
    <title><?= pageData.title ?></title>
    <link rel="stylesheet"
href="https://unpkg.com/bootstrap@4.3.1/dist/css/bootstrap.min
.css" >
    <link rel="stylesheet" href="css/mycss.css">
    <script
src="https://unpkg.com/jquery@3.3.1/dist/jquery.min.js"></scri
pt>
    <script
src="https://unpkg.com/bootstrap@4.3.1/dist/js/bootstrap.min.j
s"></script>
</head>
```

2.2 ไฟล์ header.ejs [views/partials/header.ejs]

```
<div class="container">
    <header class="bg-light text-center">
        THIS IS HEADER
    </header>
</div>
```

2.3 ไฟล์ nav.ejs [views/partials/nav.ejs]

```
<nav class="text-center">
THIS IS NAV <br>
<a href="/login">Login</a> | <a href="/register">Register</a>
| <a href="/me/logout">Logout</a>
</nav>
```

2.4 ไฟล์ footer.ejs [views/partials/footer.ejs]

```
<footer class="bg-light text-center">
THIS IS FOOTER
</footer>
```

2.5 ไฟล์ login.ejs [views/pages/login.ejs]

```
<!doctype html>
<html>
    <?- include('../partials/head') -?>
<body>
<?- include('../partials/header') -?>
<?- include('../partials/nav') -?>

<div class="container">
    <h1 class="text-center">LOGIN</h1>
    <? if(typeof errors !== 'undefined'){ ?>
    <div class="form-group row">
        <div class="col-7 mx-auto">
            <div class="alert alert-warning alert-dismissible fade
show" role="alert">
            <?= errors.message ?>
            <button type="button" class="close" data-dismiss="alert"
aria-label="Close">
                <span aria-hidden="true">&times;</span>
            </button>
            </div>
        </div>
    </div>
    <? } ?>
```

```
<form class="mt-3" action="/login" method="POST" novalidate>
    <div class="form-group row">
        <div class="col-7 mx-auto">
            <input type="email" class="form-control"
            name="email" value="<?= user.email ?>"
            placeholder="Your email">
        </div>
    </div>
    <div class="form-group row">
        <div class="col-7 mx-auto">
            <input type="password" class="form-control"
            name="password" value="<?= user.password ?>"
            placeholder="Your password">
        </div>
    </div>
    <div class="form-group row">
      <div class="col-7 mx-auto">
       <button type="submit" class="btn btn-primary btn-block mx-auto">
            Login</button>
      </div>
    </div>
</form>

</div>

<?- include('../partials/footer') -?>
</body>
</html>
```

2.6 ไฟล์ register.ejs [views/pages/register.ejs]

```
<!doctype html>
<html>
    <?- include('../partials/head') -?>
<body>
<?- include('../partials/header') -?>
<?- include('../partials/nav') -?>

<div class="container">
    <h1 class="text-center">REGISTER</h1>
    <? if(typeof errors !== 'undefined'){ ?>
    <div class="form-group row">
        <div class="col-7 mx-auto">
            <div class="alert alert-warning alert-dismissible fade show"
role="alert">
            <?= errors.message ?>
            <button type="button" class="close" data-dismiss="alert"
aria-label="Close">
                <span aria-hidden="true">&times;</span>
            </button>
            </div>
        </div>
    </div>
    <? } ?>
<form class="mt-3" action="/register" method="POST" novalidate>
    <div class="form-group row">
        <div class="col-7 mx-auto">
            <input type="text" class="form-control"
            name="name" value="<?= user.name ?>"
            placeholder="Your name">
```

```
            </div>
        </div>
        <div class="form-group row">
            <div class="col-7 mx-auto">
                <input type="email" class="form-control"
                name="email" value="<?= user.email ?>"
                placeholder="Your email">
            </div>
        </div>
        <div class="form-group row">
            <div class="col-7 mx-auto">
                <input type="password" class="form-control"
                name="password" value="<?= user.password ?>"
                placeholder="Your password">
            </div>
        </div>
        <div class="form-group row">
            <div class="col-7 mx-auto">
                <input type="password" class="form-control"
                name="confirm_password" value="<?= user.confirm_password ?>"
                placeholder="Confirm your password">
            </div>
        </div>
        <div class="form-group row">
            <div class="col-7 mx-auto">
                <button type="submit" class="btn btn-primary btn-block mx-
    auto">
                Register</button>
            </div>
        </div>
    </form>

    </div>

    <?- include('../partials/footer') -?>
    </body>
    </html>
```

2.7 ไฟล์ dashboard.ejs [views/pages/dashboard.ejs]

```
    <!doctype html>
    <html>
        <?- include('../partials/head') -?>
    <body>
    <?- include('../partials/header') -?>
    <?- include('../partials/nav') -?>

    <div class="container">
        <h1 class="text-center">Hello</h1>
        <div class="text-center my-3 border border-success">
            John Doe
        </div>
        <div class="text-center">

        </div>
    </div>

    <?- include('../partials/footer') -?>
    </body>
    </html>
```

2.8 ไฟล์ index.ejs [views/pages/index.ejs]

```
<!doctype html>
<html>
    <?- include('../partials/head') -?>
<body>
<?-  include('../partials/header') -?>
<?-  include('../partials/nav') -?>

<div class="container text-center">
    <h1><?= pageData.title ?></h1>
    <p>Welcome to <?= pageData.title ?></p>
</div>

<?-  include('../partials/footer') -?>
</body>
</html>
```

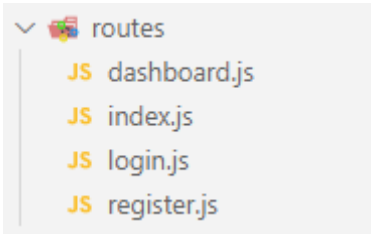
2.9 ไฟล์ error.ejs [views/pages/error.ejs]

```
<!doctype html>
<html>
    <?- include('../partials/head') -?>
<body>
<?- // include('../partials/header') -?>
<?- // include('../partials/nav') -?>

<div class="container">
    <h1><?= message ?></h1>
    <h2><?= error.status ?></h2>
    <pre><?= error.stack ?></pre>
</div>

<?- // include('../partials/footer') -?>
</body>
</html>
```

3. กำหนด Routes Path ของไฟล์ต่างๆดังนี้ สร้างโฟลเดอร์ routes และไฟล์ดังรูป




3.1 ไฟล์ index.js [routes/index.js]

ส่วนของหน้าแรก เรียกผ่าน path: "/" ซึ่งเราจะกำหนดในไฟล์ app.js

```
const express = require('express')
const router = express.Router()

router.get('/', function(req, res, next) {
    res.locals.pageData = {
        title:'Home Page'
    }
    res.render('pages/index')
})

module.exports = router
```

3.2 ไฟล์ login.js [routes/login.js]

ส่วนของหน้าล็อกอิน เรียกผ่าน path: "/login" ซึ่งเราจะกำหนดในไฟล์ app.js

```
const express = require('express')
const router = express.Router()
const { validation } = require('../validator/users')

router.route('/')
    .all((req, res, next) => {
        // ตัวแปรที่กำหนดด้วย res.locals คือค่าจะส่งไปใช้งานใน template
        res.locals.pageData = {
            title:'Login Page'
        }
        // ค่าที่จะไปใช้งาน ฟอร์ม ใน template
        res.locals.user = {
            email:'',
            password:''
        }
        // กำหนดหน้าที่ render กรณี error ไม่ผ่านการตรวจสอบข้อมูล
        req.renderPage = "pages/login"
        next()
    })
    .get((req, res, next) => {
        res.render('pages/login')
    })
    .post(validation(), (req, res, next) => {
        // ผ่านการรวจสอบ ลิ้งค์ไปหน้า /me
        res.redirect('/me')
    })

module.exports = router
```

3.3 ไฟล์ register.js [routes/register.js]

ส่วนของหน้าสมัครสมาชิก เรียกผ่าน path: "/register" ซึ่งเราจะกำหนดในไฟล์ app.js

```
const express = require('express')
const router = express.Router()
const { validation } = require('../validator/register')

router.route('/')
    .all((req, res, next) => {
        // ตัวแปรที่กำหนดด้วย res.locals คือค่าจะส่งไปใช้งานใน template
        res.locals.pageData = {
            title:'Register Page'
        }
        // ค่าที่จะไปใช้งาน ฟอร์ม ใน template
        res.locals.user = {
            name:'',
            email:'',
            password:'',
            confirm_password:''
        }
        // กำหนดหน้าที่ render กรณี error ไม่ผ่านการตรวจสอบข้อมูล
        req.renderPage = "pages/register"
        next()
    })
    .get((req, res, next) => {
        res.render('pages/register')
```

```
    })
    .post(validation(), (req, res, next) => {
        // เมื่อสมัครสมาชิกผ่านการรวจสอบ ลิ้งค์ไปหน้า /me
        res.redirect('/me')
    })

module.exports = router
```

3.4 สร้างไฟล์ dashboard.js [routes/dashboard.js]

ส่วนของหน้าสมาชิก เรียกผ่าน path: "/me" และ "/me/logout" ซึ่งเราจะกำหนดในไฟล์ app.js

```
const express = require('express')
const router = express.Router()

router.route('/')
    .get((req, res, next) => {
        res.locals.pageData = {
            title:'Dashboard Page'
        }
        res.render('pages/dashboard')
    })

router.route('/logout')
    .get((req, res, next) => {
        res.redirect('/login')
    })

module.exports = router
```

4. สร้างไฟล์ app.js และเขียนคำสั่งดังนี้

```
const express = require('express')  // ใช้งาน module express
const app = express()  // สร้างตัวแปร app เป็น instance ของ express
const path = require('path') // เรียกใช้งาน path module
const createError = require('http-errors') // เรียกใช้งาน http-errors module
const port = 3000  // port

// ส่วนของการใช้งาน router module ต่างๆ
const indexRouter = require('./routes/index')
const loginRouter = require('./routes/login')
const registerRouter = require('./routes/register')
const dashboardRouter = require('./routes/dashboard')

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');
app.set('view options', {delimiter: '?'});
// app.set('env','production')

app.use(express.json())
app.use(express.urlencoded({ extended: false }))
app.use(express.static(path.join(__dirname, 'public')))


// เรียกใช้งาน indexRouter
app.use('/', indexRouter)
app.use('/login', loginRouter)
app.use('/register', registerRouter)
app.use('/me', dashboardRouter)
```

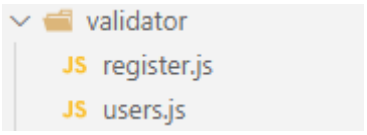
```
 // ทำงานทุก request ที่เข้ามา
app.use(function(req, res, next) {
    var err = createError(404)
    next(err)
})

// ส่วนจัดการ error
app.use(function (err, req, res, next) {
    // กำหนด response local variables
    res.locals.pageData = {
        title:'Error Page'
    }
    res.locals.message = err.message
    res.locals.error = req.app.get('env') === 'development' ? err : {}

    // กำหนด status และ render หน้า error page
    res.status(err.status || 500) // ถ้ามี status หรือถ้าไม่มีใช้เป็น 500
    res.render('pages/error')
})

app.listen(port, function() {
    console.log(`Example app listening on port ${port}!`)
})
```

5. การตรวจสอบความถูกต้องของข้อมูล สร้างโฟลเดอร์ validatior และไฟล์ดังรูป




5.1 สร้างไฟล์ users.js [validator/users.js]

```
const Joi = require('@hapi/joi')

const validation = () =>{
    return ((req, res, next) => {
        const schema = Joi.object({
            email: Joi.string() .min(6) .required() .email(),
            password: Joi.string() .min(6) .max(15) .required() });

            // ทำการตรวจสอบความถูกต้องของข้อมูล req.body ที่ส่งมา
            const result = schema.validate(req.body);
            if (result.error == undefined)
                next()
            else {
              // กรณีเกิด error ข้อมูลไม่ผ่านการตรวจสอบ
              res.locals.errors = {
                message: result.error.details[0].message
              };
              res.locals.user = req.body;
              return res.render(req.renderPage);
            }
    })
}

module.exports = { validation }
```

6. รันโปรแกรม