



UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Programador Web Avanzado

Clase N° 3: Express

Profesor: Ing. Leandro Rodolfo Gil Carrano

Email: leangilutn@gmail.com

Crear una Nueva ruta

Express – Crear una nueva ruta

Debemos crear el archivo correspondiente en la carpeta **routes**

```
└─ routes
   ├── index.js
   ├── productos.js
   └── users.js
```

Express – Crear una nueva ruta

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

/* GET home page. */
router.get('/:id([0-9]+)', function(req, res, next) {
  console.log(req.query.nombre);
  console.log(req.params.id);
  res.render('catalogo', { title: 'Productos' });
});

module.exports = router;
```

Express – Crear una nueva ruta

Luego se debe agregar lo siguiente en el archivo **app.js**

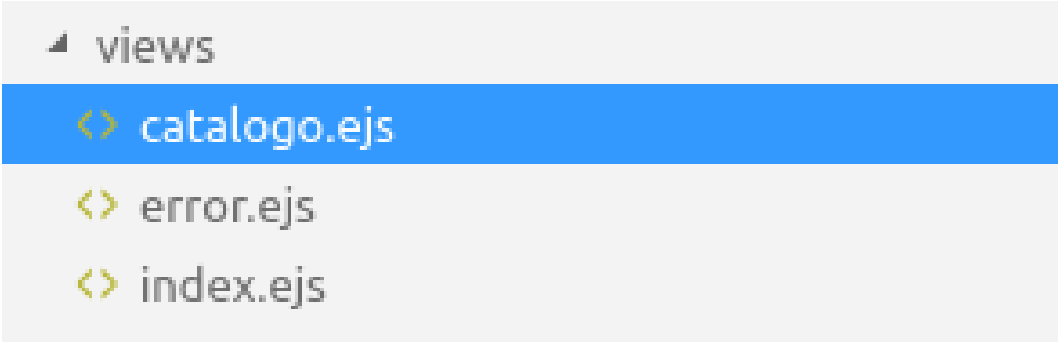
```
var indexRouter = require('./routes/index');  
var usersRouter = require('./routes/users');  
var productosRouter = require('./routes/productos');
```

```
app.use('/', indexRouter);  
app.use('/users', usersRouter);  
app.use('/productos', productosRouter);
```

```
// catch 404 and forward to error handler
```

Express – Crear una nueva ruta

Luego debemos generar la vista, en caso de querer utilizar la misma. Generarla en la carpeta **views**



```
views
├── catalogo.ejs
├── error.ejs
└── index.ejs
```

Express – Crear una nueva ruta

```
<!DOCTYPE html>
<html>
  <head>
    <title><%= title %></title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
  </head>
  <body>
    <h1>Catalogo de productos</h1>
    <p>Welcome to <%= title %></p>
  </body>
</html>
```

Interacción

Ruta - Vista

Express – Ruta - Vista

Para pasar parámetros entre una ruta y la vista debemos hacerlo de la siguiente manera:

```
res.render('catalogo', { title: 'Productos', subtitle: 'Este es el id: '+req.params.id });
```

En el primer parámetro se especifica el nombre de la vista (sin la extensión) ubicada en el directorio **views**

En el segundo parámetro se recibe un json con los parámetros que recibirá la vista

Express – Ruta - Vista

En la vista esos índices del json se mapearan en variables

```
<!DOCTYPE html>
<html>
  <head>
    <title><%= title %></title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
  </head>
  <body>
    <h1>Catalogo de productos</h1>
    <p>Welcome to <%= subtitle %></p>
  </body>
</html>
```

Parámetros En la url

Express – Parámetros en la URL

En la declaración de las rutas, podemos indicar la recepción de un parámetro por URL

```
router.get('/:id([0-9]+)', function(req, res, next) {
```

En este caso el dato enviado en la Url se asignara a la variable **id**.

También se especifica que la misma debe ser solo numérica (expresión regular)



UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Express – Parámetros en la URL

Ejemplo si navegamos la siguiente URL

```
localhost:3000/productos/1
```

Vemos el siguiente dato en la consola

```
> myapp@0.0.0 start /home/leandro/Documentos/myapp
> node ./bin/www

myapp:server Listening on port 3000 +0ms
leandro
1
GET /productos/1?nombre=leandro 200 36 170 ms - 232
```



UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Express – Parámetros en la URL

Del lado del router tenemos el siguiente código

```
router.get('/:id([0-9]+)', function(req, res, next) {  
  |  
  console.log(req.params.id);  
  res.render('catalogo', { title: 'Productos', subtitl  
  });
```

Como vemos el objeto **req.params** tiene la información de las variables mapeadas desde la URL. En este ejemplo **id**

Express – Parámetros en la URL

Si recibimos parámetros por query string

```
localhost:3000/productos/1?nombre=leandro
```

Lo recibimos de la siguiente manera

```
router.get('/:id([0-9]+)', function(req, res, next) {  
  console.log(req.query.nombre);  
  |  
  res.render('catalogo', { title: 'Productos', subtitle:  
});
```

El objeto query tiene la información enviada por query string en la URL

Recibir y retornar Datos Por json

Express – Recibir datos por json

Los datos enviados con json se reciben en el body, para lo cual debemos acceder a la propiedad **body** del objeto **req**

```
logger.info(req.body);
```

Express – Retornar datos por json

Para retornar datos por json debemos utilizar el metodo **json()** del objeto **res**

```
res.json(req.body);
```

Mysql

Express - Mysql

Ejecutar **npm install --save mysql2**

```
npm install --save mysql2
```

Express - Mysql

Luego en el código debemos incluir lo siguiente

```
// get the client
const mysql = require('mysql2');

// create the connection to database
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  database: 'test'
});

// simple query
connection.query(
  'SELECT * FROM `table` WHERE `name` = "Page" AND `age` > 45',
  function(err, results, fields) {
    console.log(results); // results contains rows returned by server
    console.log(fields); // fields contains extra meta data about results,
  }
);
```

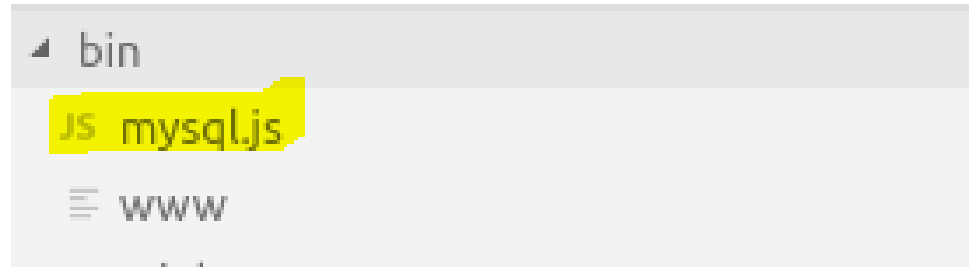
Express - Mysql

Con el uso de placeholder

```
// with placeholder  
connection.query(  
  'SELECT * FROM `table` WHERE `name` = ? AND `age` > ?',  
  ['Page', 45],  
  function(err, results) {  
    console.log(results);  
  }  
);
```

Express - Mysql

Para acomodar un poco mejor el código, creemos un archivo **mysql.js** en el directorio **bin**



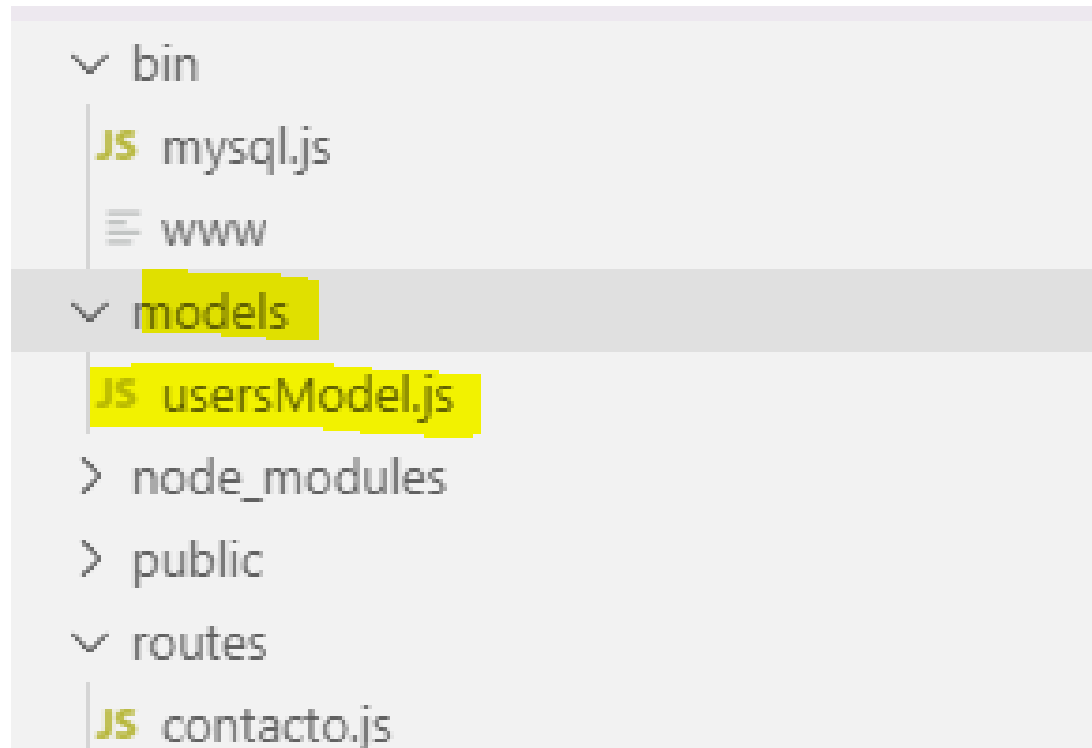
Express - Mysql

En ese archivo incluimos lo siguiente:

```
const mysql = require('mysql2/promise');  
module.exports.pool = mysql.createPool({  
  host: 'localhost',  
  user: 'root',  
  password: '',  
  database: 'pwa',  
  waitForConnections: true,  
  connectionLimit: 10,  
  queueLimit: 0  
}); // query database
```


Express - Mysql

Creamos el directorio **models**



Express - Mysql

Dentro de **usersModel.js** incluimos el siguiente código

```
var db = require("../bin/mysql")

module.exports.getAll = async function(){
  const [rows, fields] = await db.pool.execute("select * from usuarios");

  return rows;
}

module.exports.save = async function(nombre, apellido){
  await db.pool.query(
    'INSERT INTO usuarios SET nombre = ?, apellido = ?',
    [ nombre, apellido ]
  );
  return;
}
```

Express - Mysql

Creamos el directorio controllers, en el creamos el archivo users.js

```
└─ bin
   JS mysql.js
   ≡ www
  ✓ controllers
    JS users.js
  > models
  > node_modules
  > public
  ✓ routes
```

Express - Mysql

En el controlador incluimos modelo y definimos modulo con métodos a disponibilizar

```
var users = require("../models/usersModel")

module.exports = {
  getAll: async function(req, res, next){
    var data = await users.getAll()
    console.log("data",data)
    res.status(200).json(data)
  },
  save: async function(req, res, next){
    await users.save("Leandro","Gil2")
    res.status(200).json({"status":"ok"})
  }
}
```

Express - Mysql

Por ultimo modificamos routes para hacer un require del controlador

```
var express = require('express');  
var router = express.Router();  
var users = require("../controllers/users")  
  
/* GET home page. */  
router.get('/', users.getAll);  
router.post('/save', users.save);  
  
module.exports = router;
```

Express - Mysql

Model

```
var db = require("../bin/mysql")

module.exports.getAll = async function(){
  const [rows, fields] = await db.pool.execute("select * from usuarios");

  return rows;
}

module.exports.save = async function(nombre, apellido){
  await db.pool.query(
    'INSERT INTO usuarios SET nombre = ?, apellido = ?',
    [ nombre, apellido ]
  );
  return;
}
```