UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Preguntas?

Lo que vamos a ver hoy!

- ❏ Qué es Express?
- ❏ Ruteo de Express
- ❏ Manejo de archivos estáticos
- ❏ Recepción de formularios

Express

Qué es Express?

Qué es Express?

Es un framework (o marco de trabajo) para Node.js que sirve para ayudarnos a crear aplicaciones web en menos tiempo.

Facilita principalmente la creación de nuestro servidor.

Express

Qué es Express?

Hasta ahora trabajamos con código que se ejecuta en el navegador.

Se realiza una petición por HTTP (http://www.pagina.com.ar) y nos devuelve un HTML que puede tener código JavaScript que el navegador que estemos usando interpreta y muestra.

Después vimos bases de datos. Estas se encuentran alojadas en algún servidor y las instrucciones que vimos (insertar, buscar, modificar, borrar), se hacen en ese servidor.

Qué es Express?

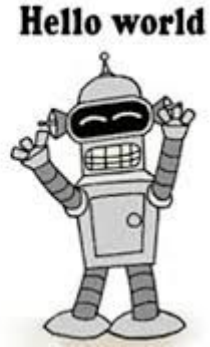
Qué pasa si queremos hacer "cosas" en el servidor. Que se ejecuten en el servidor y recién después, enviar la página al navegador?

Por ejemplo, loguear un usuario, registrar un usuario.

Para esto podemos usar NodeJS con Express!

NodeJS se ejecuta en el servidor, realiza operaciones y luego envía la página al navegador.

Con **Express**, además podemos hacer más fácil un **servidor** que "escuche" los pedidos de los usuarios para poder entregarles las cosas que nos piden.



INSTALACIÓN

Instalación de express

Desde la línea de comandos y parados en la carpeta del proyecto, ejecutar el comando:

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

(si no se quiere añadirlo a la lista de dependencias y que la instalación sea temporal, omitir --save)

Nota: todo lo que se instala con **--save** se añade a la lista de dependencias en el archivo package.json

Primer programa básico en Express

Primer programa básico

index.js

```
var express = require('express');
var app = express();

app.get('/usuario', function (req, res) {
  res.send('Hola Mundo!');
});

app.listen(3000, function () {
  console.log('Ejemplo de aplicacion escuchando en el port 3000!');
});
```

Ruteo

Métodos HTTP GET y HTTP POST

Permiten enviar información al servidor.

Métodos HTTP GET

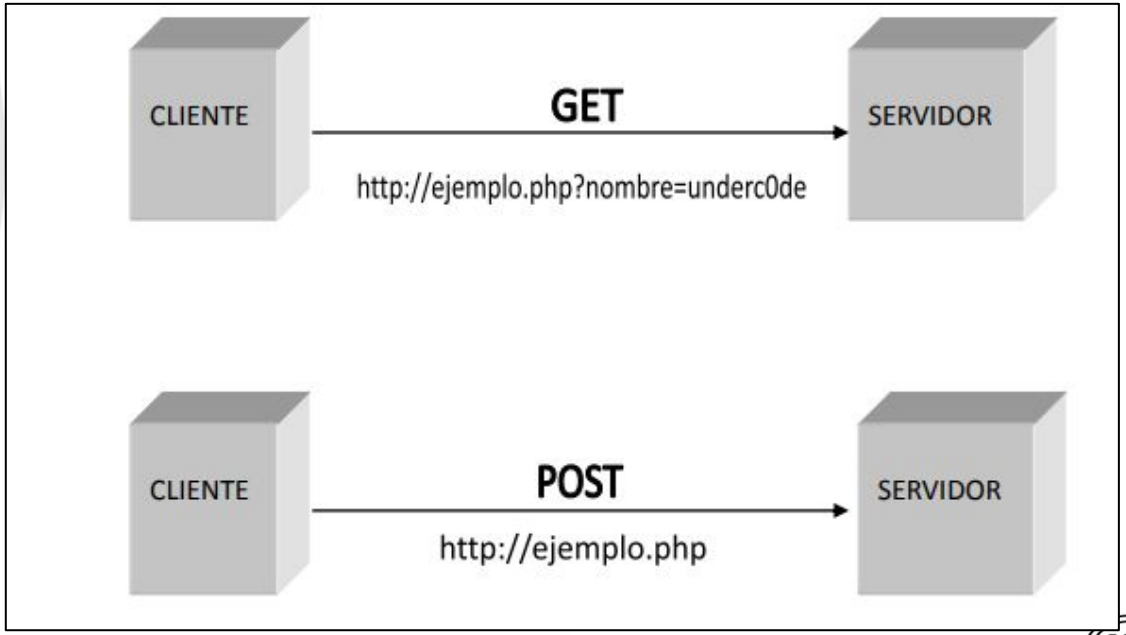
- La info viaja visible directamente en la URL (limitado a 2000 caracteres).
- No se pueden enviar datos binarios (imágenes, archivos, etc.)
- La página web y la información codificada se separan por un interrogante ? y los bloques de datos se separan con &:

```
www.ejemplo.com/index.htm?key1=value1&key2=value2&key3=value3...
```

Métodos HTTP POST

- La **info** se envía a través del **body del HTTP Request** (no aparece en la URL)
- No tiene límite de cantidad de información a enviar.
- La información proporcionada no es visible, por lo que se puede enviar información sensible.
- Se puede usar para enviar datos binarios (archivos, imágenes...).

Diferencia entre los métodos



Direccionamiento básico

Cómo responde una aplicación a una solicitud del cliente en un determinado punto final?

Tiene 2 componentes básicos:

- una vía de acceso (URI)
- un método de solicitud HTTP específico (GET, POST, etc.)

Direccionamiento básico

Estructura de una ruta

```
app.METHOD(                    )
```

app: instancia de express
METHOD: método de solicitud HTTP (GET, POST, etc.)
: vía de acceso al servidor
: función que se ejecuta cuando se correlaciona la ruta

Direccionamiento básico

Ejemplo:

Responde con Hola! en la página inicial

```
app.get('/', function (req, res) {
        res.send('Hola!');
});
```

Responde una solicitud POST en la página de inicio

```
app.post('/', function (req, res) {
        res.send('Hola pedido POST');
});
```

PENSEMOS…

NodeJS

Pensemos

Si **'/'** es el raiz **www.pagina.com.ar**

Cómo sería la ruta para **www.pagina.com.ar/usuario/**

SIGAMOS…

Direccionamiento básico

direccionamiento all()

No se deriva con ningún método HTTP.

Se utiliza para responder a cualquier método HTTP.

```
app.all('/productos', function (req, res, next) {
        console.log('Accediendo a la sección de productos ...');
});
```

Métodos de respuesta

Método	Descripción
res.download()	Solicita un archivo para descargarlo.
res.end()	Finaliza el proceso de respuesta.
res.json()	Envía una respuesta JSON.
res.jsonp()	Envía una respuesta JSON con soporte JSONP.
res.redirect()	Redirecciona una solicitud.
res.render()	Representa una plantilla de vista.
res.send()	Envía una respuesta de varios tipos.
res.sendFile	Envía un archivo como una secuencia de octetos.
res.sendStatus()	Establece el código de estado de la respuesta y envía su representación de serie como el cuerpo de respuesta.

Direccionamiento básico

app.route()

Sirve para incluir en un único lugar los controladores de rutas para una vía

```
app.route('/book')
        .get(function (req, res) {
            res.send('Elegir un libro al azar')
        })
        .post(function (req, res) {
            res.send('Agregar un libro')
        })
```

Solicitud con parámetros

Se puede hacer de 2 formas:

- http://localhost:3000/usuario/Lorena
- http://localhost:3000/usuario?nombre=Lorena

Pedido con parámetros

http://localhost:3000/usuario/Lorena

```
app.get('/usuario/:nombre', function (req, res) {
    res.send('Hola '+req.params.nombre);
});
```

Express

Pedido con parámetros

http://localhost:3000/usuario?nombre=Lorena

```
app.get('/usuario', function (req, res) {
        // llamar con ?nombre=algo
        res.send('Hola '+req.query.nombre);
});
```

Manejo de archivos estáticos

Manejo de archivos estáticos

Por ejemplo, imágenes, archivos CSS y archivos JavaScript.

Manejo de archivos estáticos

Hay que pasar el nombre de la carpeta que contiene los archivos estáticos con la siguiente sentencia:

```
app.use(express.static('public'));
```

Ahora se pueden cargar los archivos que hay en la carpeta 'carpeta', por ejemplo:

```
http://localhost:3000/images/kitten.jpg
http://localhost:3000/css/style.css
http://localhost:3000/js/app.js
http://localhost:3000/images/bg.png
http://localhost:3000/hello.html
```

Express busca los archivos relativos a la carpeta, por lo que el nombre de la carpeta no forma parte de la URL.

Manejo de archivos estáticos

Se pueden incluir varias carpetas con archivos estáticos agregando una sentencia por cada carpeta:

```
app.use(express.static('public'));
app.use(express.static('files'));
```

Siendo así, Express busca los archivos en el orden en el que se definen las carpetas.

Manejo de archivos estáticos

Prefijo de vía de acceso virtual

No existe la vía de acceso realmente.

```
app.use('/static', express.static('public'));
```

Entonces ahora puede cargar los archivos que hay en la carpeta public desde el prefijo de vía de acceso /static.

```
http://localhost:3000/static/images/kitten.jpg
http://localhost:3000/static/css/style.css
http://localhost:3000/static/js/app.js
http://localhost:3000/static/images/bg.png
http://localhost:3000/static/hello.html
```

Manejo de archivos estáticos

Prefijo de vía de acceso virtual

La vía de acceso que se proporciona sigue siendo relativa a la carpeta desde donde inicia el proceso node.

Por eso, si se ejecuta la aplicación desde cualquier otra carpeta, es más seguro utilizar la vía de acceso absoluta de la carpeta a la que se desea dar servicio:

```
app.use('/static', express.static(__dirname + '/public'));
```

Recepción de formularios

Recepción de formularios

Instalar

```
npm install --save body-parser
```

Incluir en el programa

```
app.use(require('body-parser')());
```

Recepción de formularios

Tenemos que tener hecho el formulario con method = “POST” y lo ubicamos dentro de alguna carpeta de archivos estáticos

```
<form action="/procesar" method="POST">
    <input type="hidden" name="oculto" val="campo oculto pero no invisible!">
    <div>
            <label for="campofruta">Su fruta preferida: </label>
                    <input type="text" id="campofruta" name="fruta">
    </div>
    <div>
            <button type="submit">Enviar</button>
    </div>
</form>
```

Recepción de formularios

Código del servidor

```
var express = require('express');
var bodyParser = require('body-parser');
var app = express();

app.use(express.static(__dirname + '/public'));

app.use(bodyParser());

app.set('port', process.env.PORT || 3001);

app.post('/procesar', function (req, res) {
    var fruta = req.body.fruta;
    var html = 'Tu Fruta Favorita es: ' + fruta + '.<br>' +
               '<a href="/ejemplo_formulario.html">Probar de nuevo.</a>';
    res.send(html);
});


app.listen(app.get('port'), function(){
  console.log( 'Express se ha iniciado en http://localhost:' +
    app.get('port') + '; presione Ctrl-C para cerrar el servidor.' );
});
```

PRÁCTICA

PRÁCTICA EXPRESS

Ejercicio 1

Crear el servidor que reciba las siguientes URI

http://localhost:3005/color_favorito/rosa o http://localhost:3005/color_favorito/amarillo etc

http://localhost:3005/registro_usuario?nombre=<un_nombre>&apellido=<un_apellido>&edad=<una_edad>

Frente a la recepción de la petición, deberá devolver un pequeño HTML que contenga la información pasada como parámetro.

PRÁCTICA EXPRESS

Ejercicio 2

Crear un formulario de registración con los siguientes datos: **nombre, apellido, edad, número de celular, país de nacimiento, país de residencia.**

Recibir los datos en el servidor y armar otra página de respuesta que incluya los datos del usuario y un enlace a la página de registración nuevamente.

FIN!!!!