EXPRESS

rortegag

Métodos express()

```
import express from "express"
const app = express()
```

express.json([opciones])

inflate
Para gestionar los cuerpos comprimidos como habilitar y deshabilitar.

limit
Controla el tamaño máximo del cuerpo de la petición.

reviver
Se pasa directamente a JSON.parse como segundo argumento.

type
Se utiliza para determinar el tipo de middleware que analizará.

verify
Es una función no definida que se utiliza para verificar el análisis del middleware.

express.raw([opciones])

inflate
Para gestionar los cuerpos comprimidos como habilitar y deshabilitar.

limit
Controla el tamaño máximo del cuerpo de la petición.

type
Se utiliza para determinar el tipo de middleware que analizará.

verify
Es una función no definida que se utiliza para verificar el análisis del middleware.

express.Router([opciones])

caseSensitive
Permite distinguir entre mayúsculas y minúsculas.

mergeParams
Si los nombres de los parámetros del hijo y del padre están en conflicto, el hijo tiene prioridad.

strict
Habilita el enrutamiento estricto.

express.urlencoded([opciones])

extended
Permite elegir entre analizar los datos codificados en la URL o la biblioteca qs.

parameterLimit
Controla el número de parámetros.

inflate
Para gestionar los cuerpos desinflados como habilitar y deshabilitar.

limit
Controla el tamaño máximo del cuerpo de la petición.

type
Se utiliza para determinar el tipo de middleware que analizará.

verify
Es una función no definida que se utiliza para verificar el parseo del middleware.

express.static(root, [opciones])

dotfiles
Determina cómo se utilizan los dotfiles.

etag
Opera la generación de etags.

extensions
Utiliza la extensión de archivo fallback.

fallthrough
Activar o desactivar la directiva inmutable en la cabecera de respuesta Cache-Control.

index
Envía el fichero índice del directorio especificado.

LastModified
Establece la cabecera Last-Modified con la fecha de la última modificación.

setHeaders
Función para establecer las cabeceras HTTP que se servirán con el archivo.

express.text([opciones])

defaultCharset
Establece el conjunto de caracteres por defecto para el contexto de texto.

inflate
Para gestionar los cuerpos desinflados como habilitar y deshabilitar.

limit
Controla el tamaño máximo del cuerpo de la petición.

type
Se utiliza para determinar el tipo de middleware que analizará.

verify
Es una función no definida que se utiliza para verificar el análisis del middleware.

Para más información: https://expressjs.com/

EXPRESS

rortegag

Application

Propiedades

app.local

```
app.locals.title = “Título”;
console.dir(app.locals.title);
```

Crear objetos con variables locales

Eventos

```
admin.on(’mount’, (parent) {
  console.log(’Montado en Admin’)
})
```

Montaje en una aplicación principal.

app.mountpath

```
const admin = express()
admin.get(’/’, function(req,res) {
  console.log(admin.mountpath)
  res.send(’Página principal del administrador’)
})
app.use(’<carpeta>’, admin);
```

Montaje de un sub - app.

Métodos

```
app.get(’/’, function(req, res) {
  res.send(’Petición GET del mensaje’)
})
```

Obtener peticiones GET a la ruta especificada.

```
app.post(’/’, function(req,res) {
  res.send(’Petición POST a una página web’)
})
```

Enviar la petición POST a la ruta especificada.

```
app.put(’/’, function(req,res) {
  res.send(’Petición PUT a una página web’)
})
```

Enviar la petición PUT a la ruta especificada.

```
app.delete(’/’, function(req,res) {
  res.send(’Petición DELETE a una página web’)
})
```

Enviar la petición DELETE a la ruta especificada.

```
app.all(’/’, function(req,res,next) {
  console.log(’Acceder a la sección secreta...’)
  next()
})
```

Enrutamiento de todo tipo de peticiones HTTP.

```
app.use(function(req,res,next) {
  res.send(’Hola Mundo’)
})
```

Para invocar la capa de middleware que desea añadir.

```
app.param(’user’, function(req,res,next) {
  User.find(id, function(err, user) {
    if(err) {
      next(err)
    } else if (user) {
      req.user = user
      next()
    } else {
      next(new Error(’Error al cargar el usuariosuario’))
    }
  })
})
```

Añadir trigger callback a parámetros de ruta.

Request

Propiedades

```
req.param(’nombre’)
```

Solicita el nombre del parámetro cuando está presente.

```
app.post(’/’, function (req, res, next) {
  console.log(req.body)
  res.json(req.body)
})
```

Datos enviados en el cuerpo de la solicitud.

```
console.dir(req.cookies.name)
```

Contiene cookies enviadas por la petición.

```
console.dir(req.query.q)
```

Parámetro de cadena de consulta en la ruta.

```
console.log(req.route)
res.send(’GET’)
```

Salida de todas las capas, métodos, ruta.

```
console.dir(req.signedCookies.user)
```

Registra todas las cookies firmadas enviadas por la solicitud.

Métodos

```
req.get(’content-type’)
```

Devuelve el encabezado req HTTP especificado.

```
req.accepts(’html’)
```

Comprueba si los tipos de contenido especificados están disponibles o no.

```
req.is(’json’)
```

Solicita el tipo de contenido que corresponda.

```
var range = req.range(1000)
if(range.type === 'bytes') {
  range.forEach(function(r) {
      // Código...
  })
}
```

Analizador de cabeceras de rango.

Para más información: https://expressjs.com/

EXPRESS

rortegag

Response

Métodos

```
res.redirect('https://google.com')
```
Redirecciona a la página deseada.

```
res.send({message: 'Mensaje'})
```
Respuesta a la página web.

```
res.json({alert: '¡Alerta!'})
```
Respuesta en tipo JSON.

```
res.render('index')
```
Renderizado del archivo previsto.

```
const file = req.params.name;
res.sendFile(file, options, function(err) {
  if(err) {
    next(err)
  } else {
    console.log('Enviado: ', file)
  }
})
```
Envía el archivo a la ruta deseada.

BodyParser

```
const BodyParser = require('body-parser')
app.use(BodyParser.json())
app.use(BodyParser.urlencoded({
  extended: true
}))
```
Analiza los cuerpos de las solicitudes entrantes.

Para más información: https://expressjs.com/