VIETNAM AUSTRALIA
Vocational College

React Advance
Bài 3: Xử lý Routing

daynghevietuc.com

Nội dung

1. Routing
2. Route path
3. Route parameters
4. Route handlers

1. Routing

Routing

- Routing đề cập đến cách phản hồi (*response*) từ các *endpoint* cho các request của client *(kiến thức cơ bản của phần này đã được giới thiệu ở bài trước)*
- Chúng ta có thể định nghĩa các Routing bằng cách sử dụng các method của Express app tương ứng với các method của HTTP
- Ví dụ
  - **app**.*get*() để xử lý (handle) ***GET*** request
  - **app**.*post*() để xử lý (handle) ***POST*** request
  - **app**.*all*() để xử lý (handle) cho tất cả các request method của HTTP
  - **app**.*use*() để định nghĩa middleware như là các callback function

Routing

- Đây là một đoạn code cơ bản làm mẫu để định nghĩa route

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

// respond with "hello world" when a GET request is made to the homepage
app.get('/', function (req, res) {
  res.send('hello world')
})
```

Route methods

- Route method có nguồn gốc từ HTTP methods, và được cài đặt vào thành mộ class của của Express
- Đoạn code sau cài đặt 2 phương thức GET và POST ở root của app

```
// GET method route
app.get('/', function (req, res) {
  res.send('GET request to the homepage')
})

// POST method route
app.post('/', function (req, res) {
  res.send('POST request to the homepage')
})
```

Route methods

- Express cũng hỗ trợ phương thức **app**.*all*() để xây dựng middleware.
- Phương thức **app**.*all*() này sẽ duyệt tất cả các request sử dụng tất cả các phương thức ***GET, POST, PUT, DELETE***

```
app.all('/secret', function (req, res, next) {
  console.log('Accessing the secret section ...')
  next() // pass control to the next handler
})
```

2. Route path

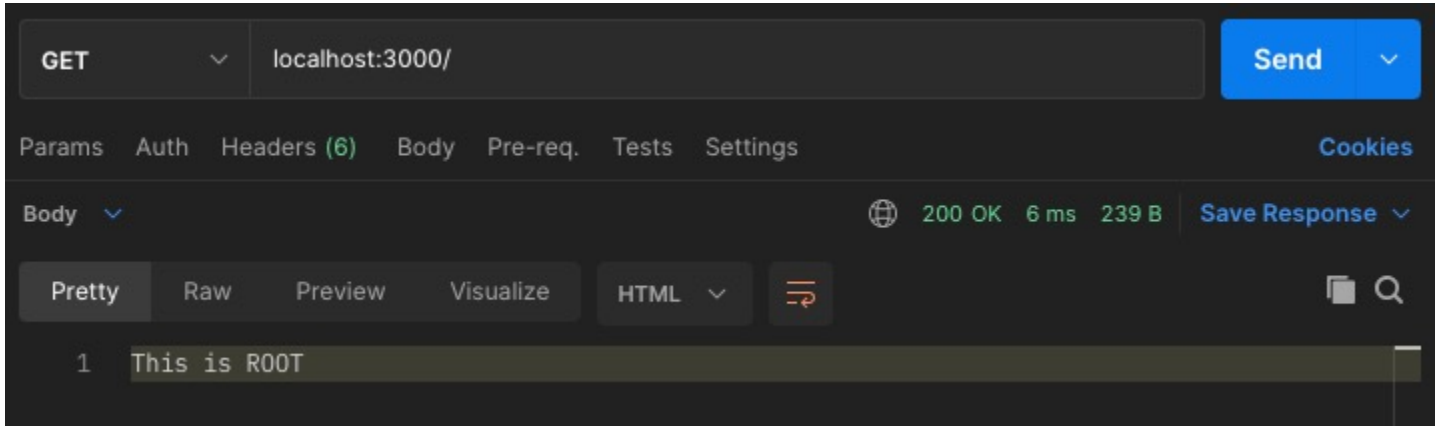
Route path

- Route path là sự kết hợp với các request method
- Route path sẽ định nghĩa/xác định các endpoint khi các request được gửi đến
- Route path có thể là String, String Pattern hoặc là Regular Expression
- Các kỹ tự đặc biệt *?*, *+*, *** và *()* là các mẫu/ký tự đặc biệt của biểu thức Regular Expression
- Các dấu gạch nối (**-**) và các dấu chấm (**.**) là được sử dụng để định nghĩa các Route path
- Nếu cần sử dụng dấu (**$**) in string path, thì cần phải bọc lại trong dấu ([và]).
- Ví dụ
  - Khi chúng ta muốn định nghĩa đường dẫn "/data/$book"
  - Thì nên viết "/data/ ([\$])book"

Route path

- Để định nghĩa root path ta sẽ sử dụng ký tự /

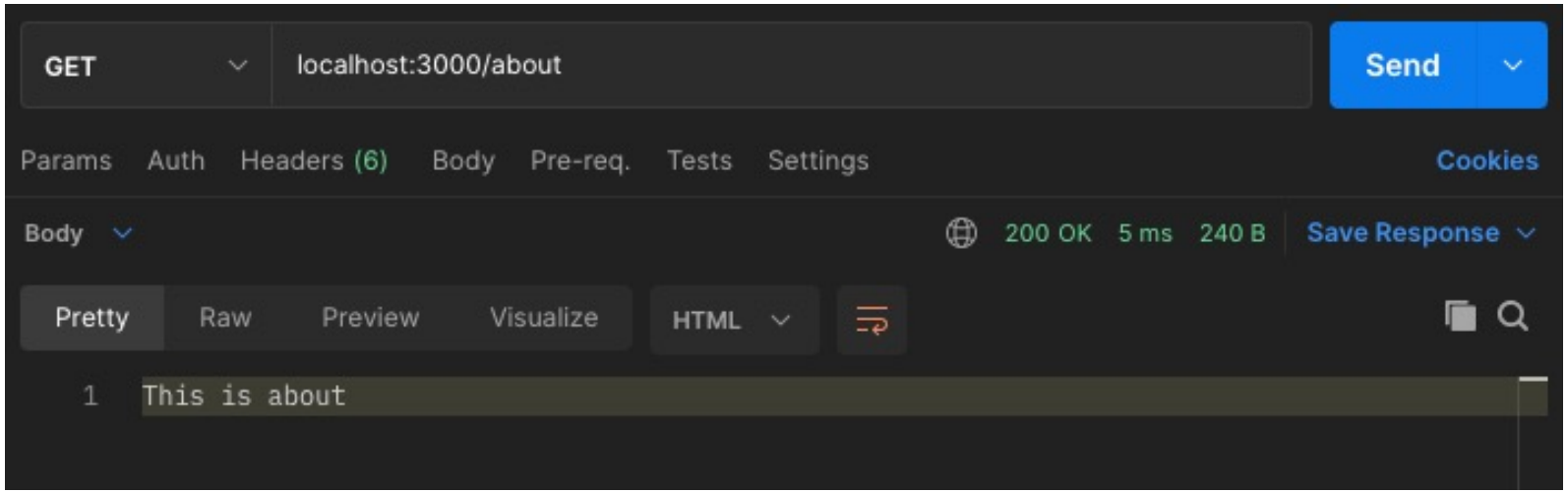
```
app.get('/', (req, res) => {
  res.send('This is ROOT')
})
```



Route path

- Định nghĩa Route path cho endpoint /about

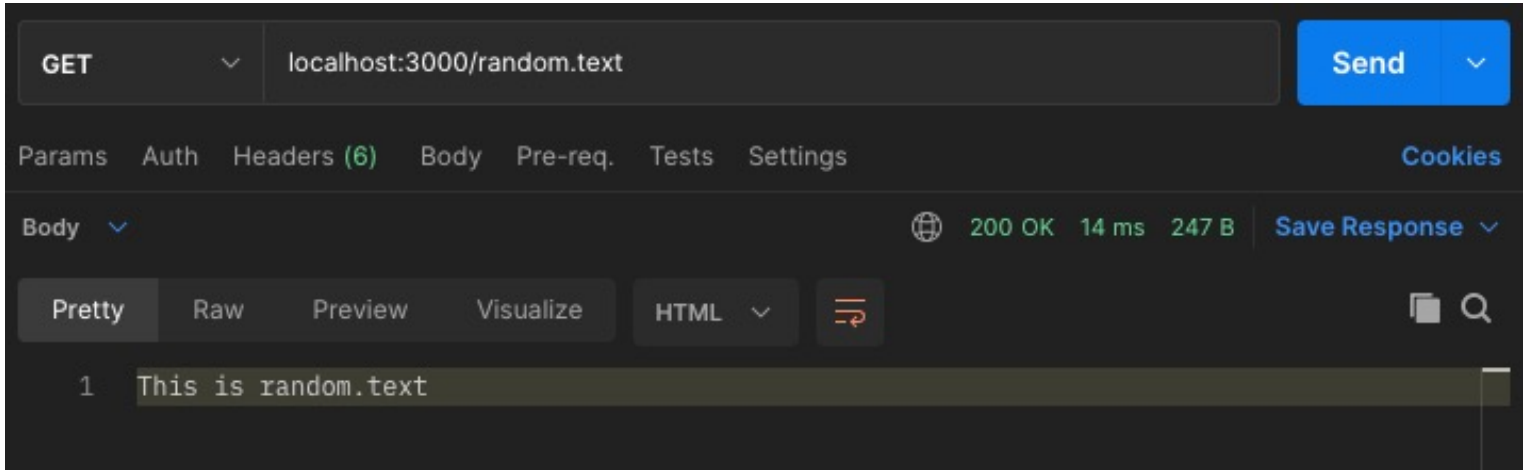
```
app.get('/about', function (req, res) {
  res.send('This is about')
})
```



Route path

- Định nghĩa Route path cho endpoint /random.text

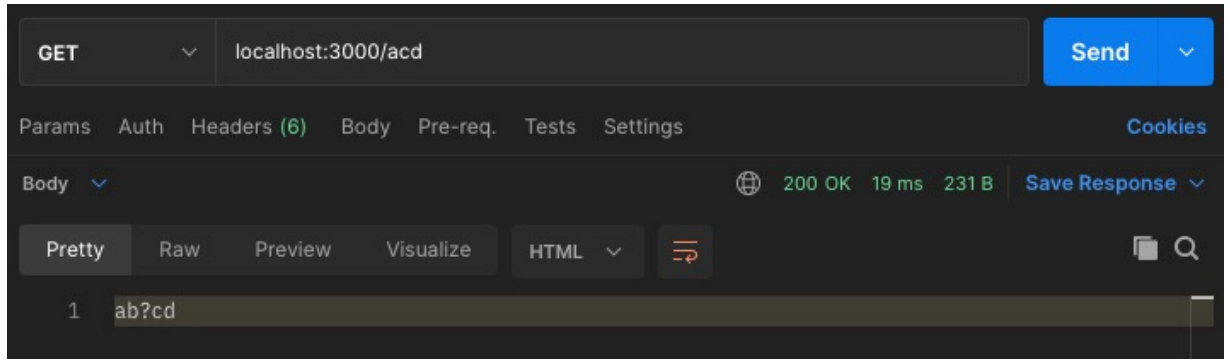
```
app.get('/random.text', (req, res) => {
  res.send('This is random.text')
})
```

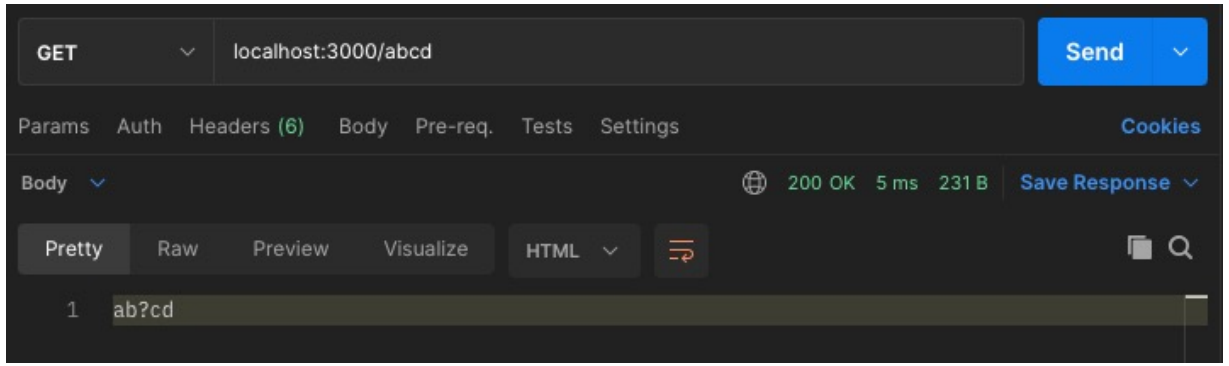


Route path

- Sử dụng ? để định nghĩa Route path cho endpoint /acd và /abcd

```
app.get('/ab?cd', function (req, res) {
  res.send('ab?cd')
})
```

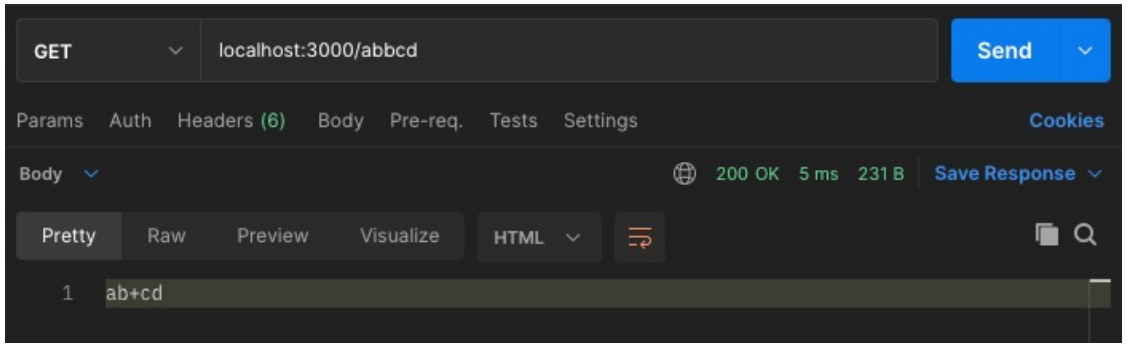


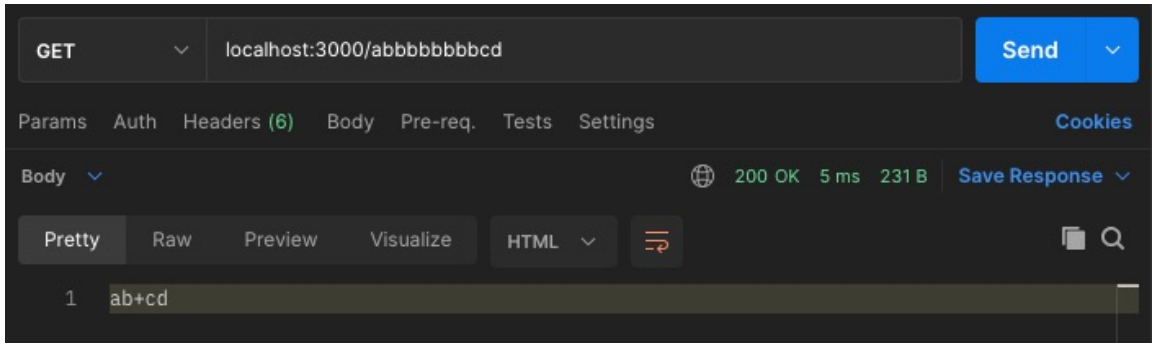


Route path

- Sử dụng + để định nghĩa Route path cho endpoint /abcd, /abbcd, /abbbcd

```
app.get('/ab+cd', function (req, res) {
  res.send('ab+cd')
})
```

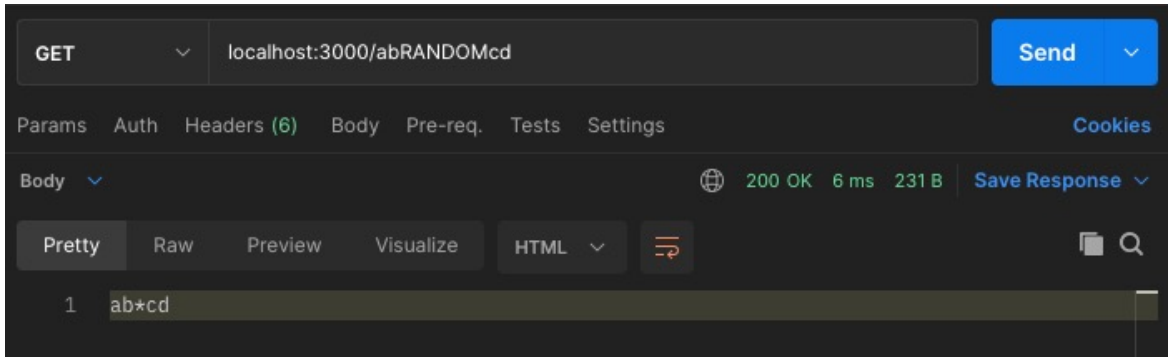


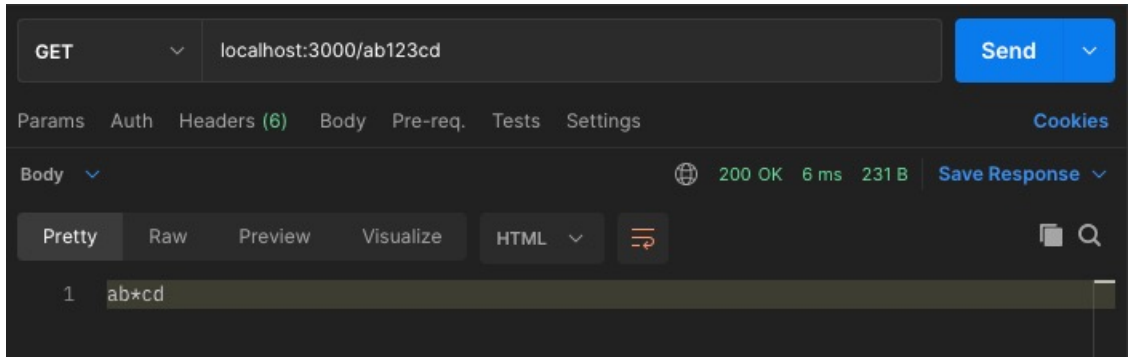


Route path

- Sử dụng * để định nghĩa Route path cho endpoint /abcd, /abxcd, /abRAMDOMcd, /ab123cd

```
app.get('/ab*cd', function (req, res) {
  res.send('ab*cd')
})
```

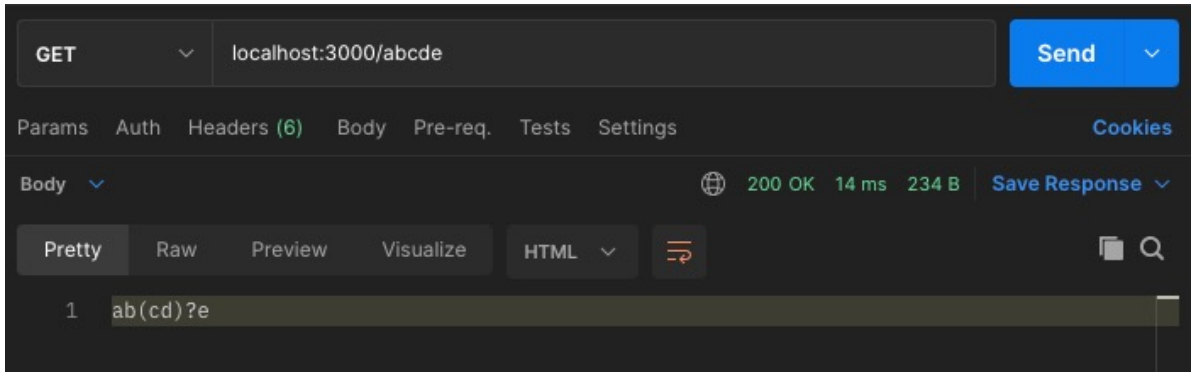


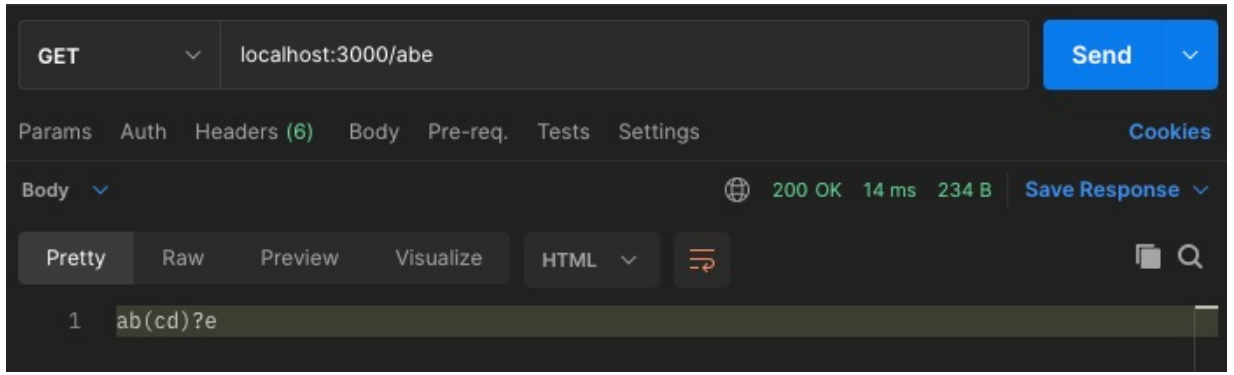


Route path

- Sử dụng ( ) để định nghĩa Route path cho endpoint /abe, /abcde

```
app.get('/ab(cd)?e', function (req, res) {
  res.send('ab(cd)?e')
})
```

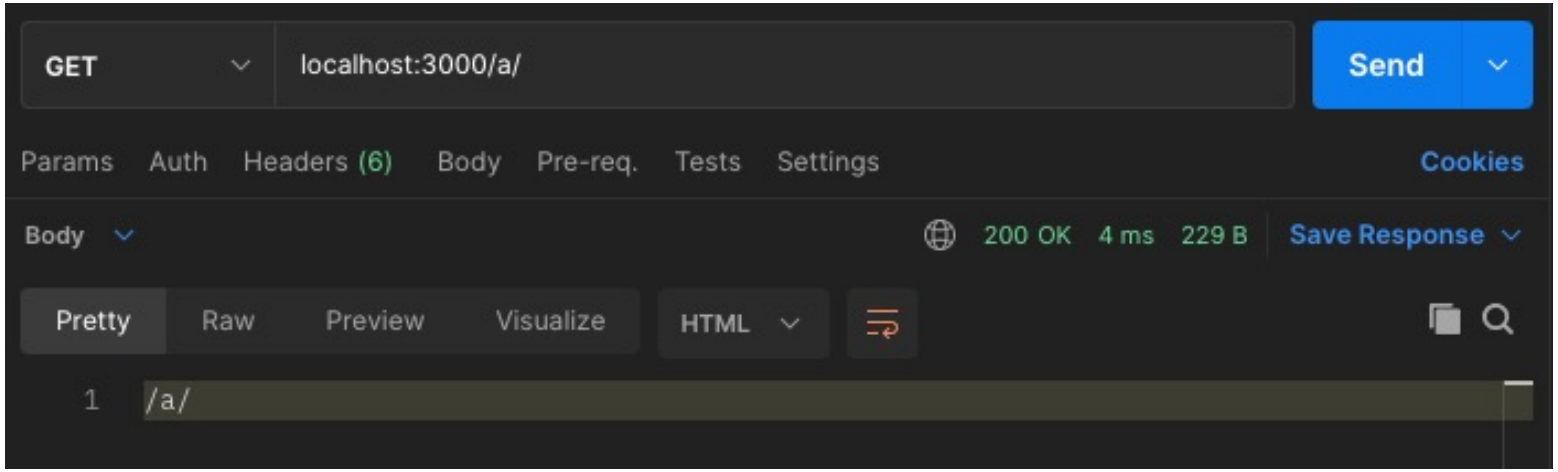




Route path

- Định nghĩa Route path cho endpoint với ký tự "a"

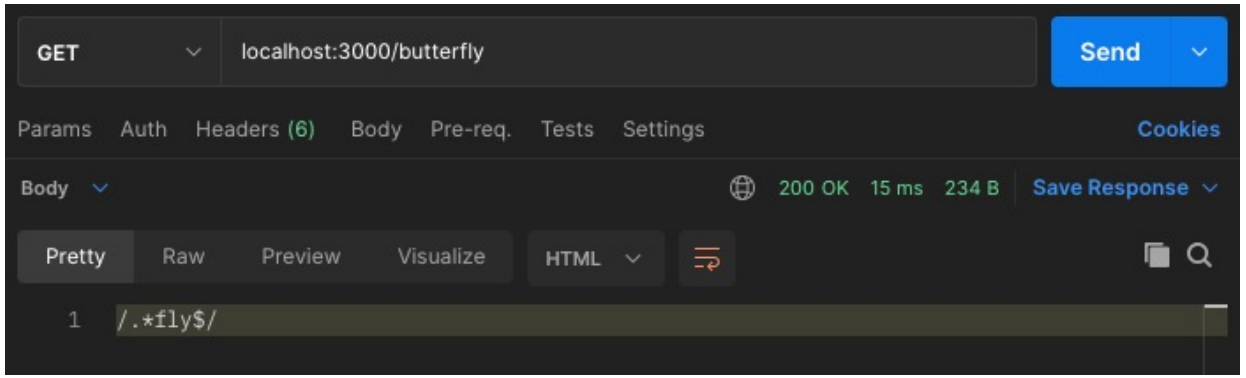
```
app.get(/a/, function (req, res) {
  res.send('/a/')
})
```

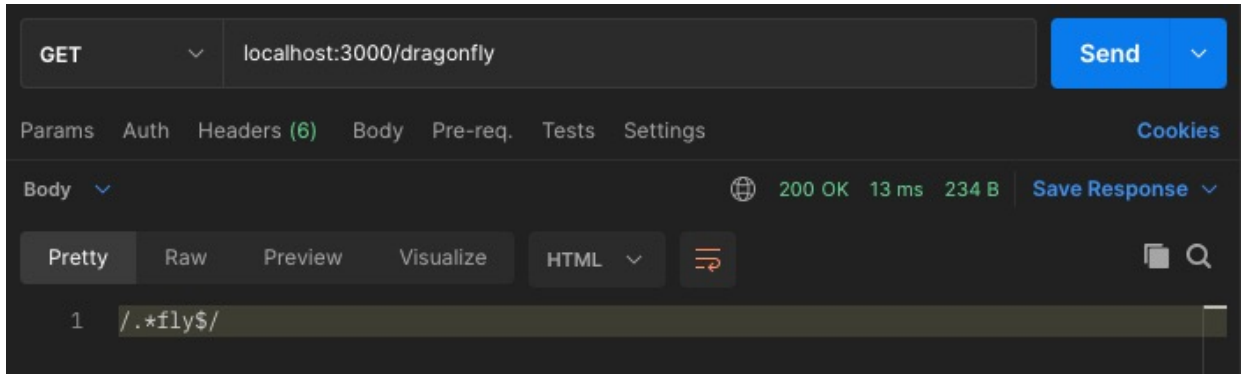


Route path

- Định nghĩa Route path cho endpoint /butterfly hoặc /dragonfly
- Nhưng sẽ ***không*** khớp với /butterfly*man* hoặc /dragonfly*man*

```
app.get(/.*fly$/, function (req, res) {
  res.send('/.*fly$/')
})
```





3. Route parameters

Route parameters

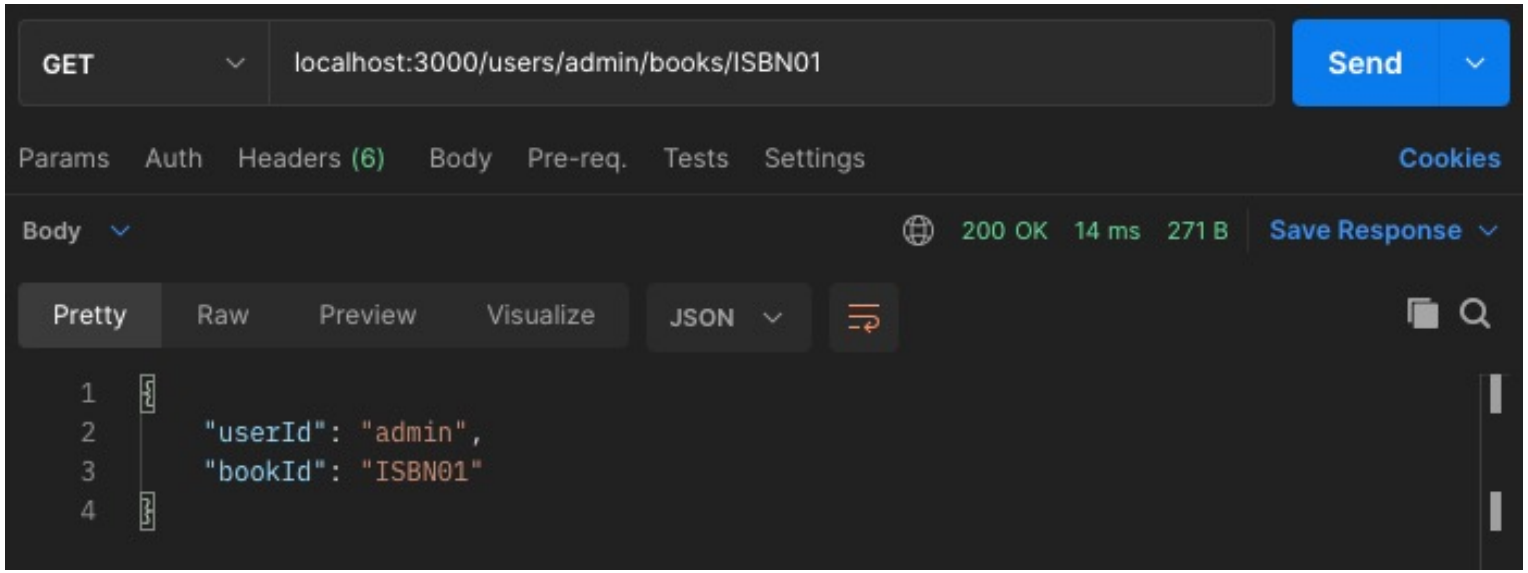
- Route parameters là một đoạn trong URL, nó sẽ chứa giá trị của các parameter (tham số)
- Cách truyền tham số dạng này trên URL còn được gọi là *QueryString*
- Để nhận lại/lấy các giá trị từ các parameter này thì sử dụng đối tượng **req**.*params* kèm theo tên của các object được truyền từ URL

```
Route path: /users/:userId/books/:bookId
Request URL: http://localhost:3000/users/34/books/8989
req.params: { "userId": "34", "bookId": "8989" }
```

Route parameter

- Định nghĩa route với các route parameters đơn giản như sau

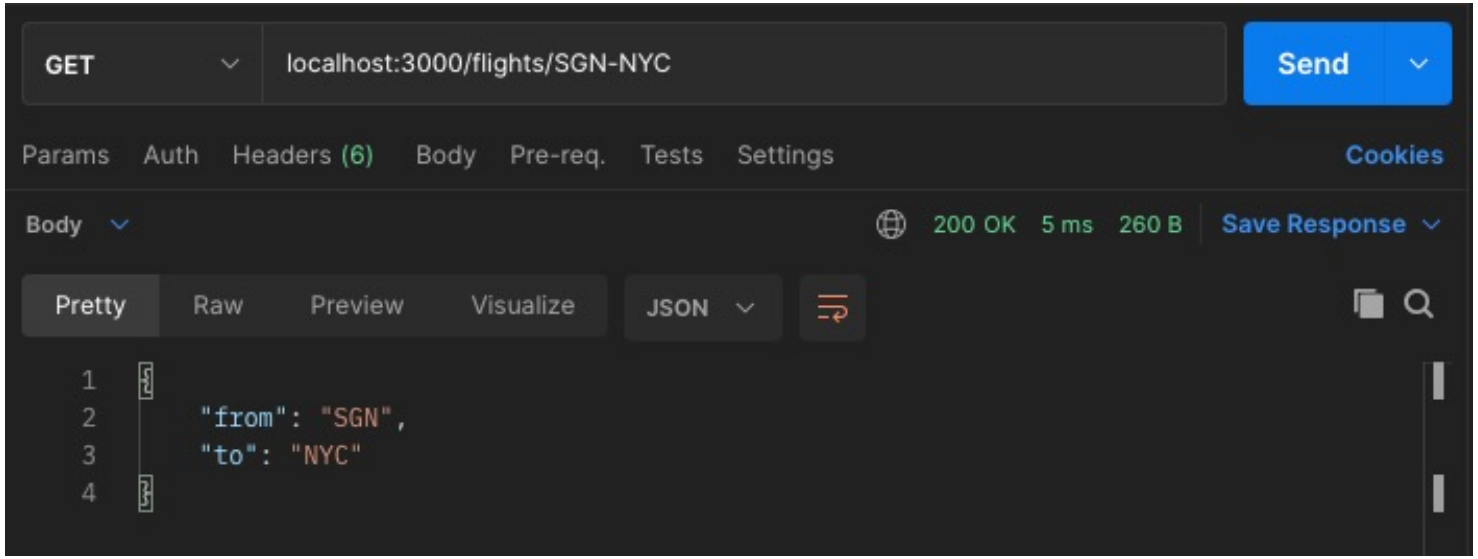
```
app.get('/users/:userId/books/:bookId', function (req, res) {
  res.send(req.params)
})
```



Route paramters

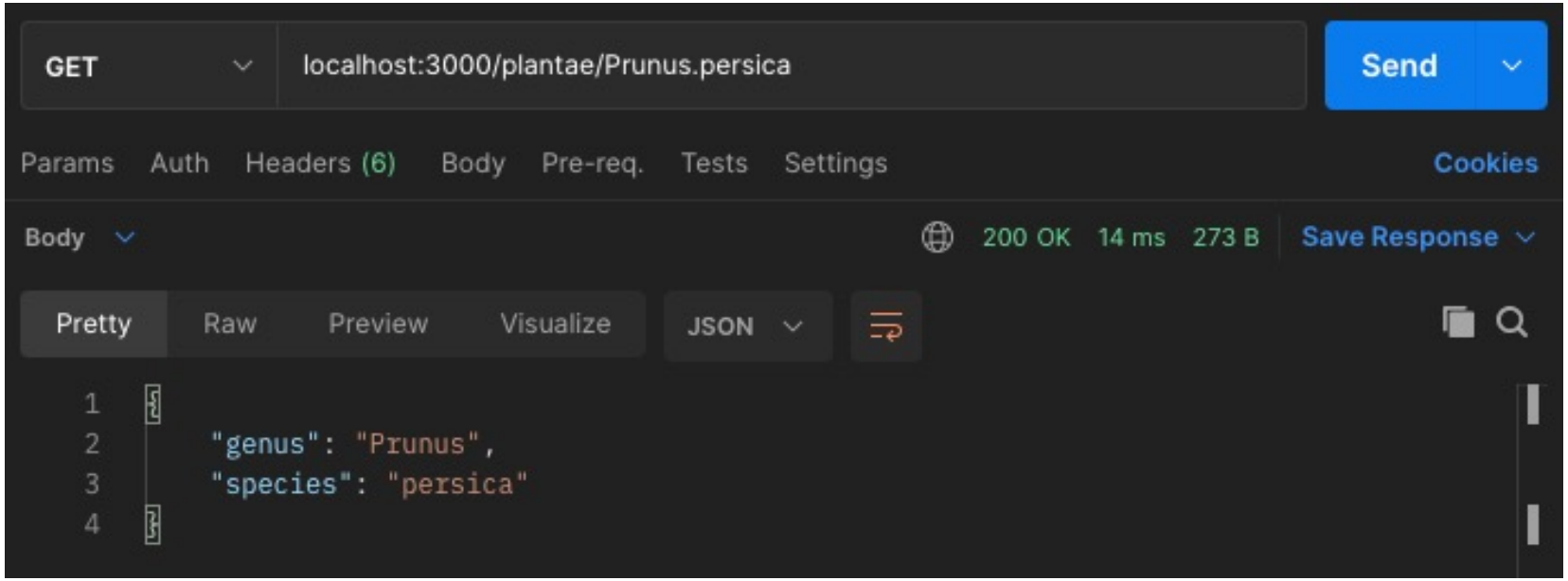
- Sử dụng dấu gạch nối (-) và các dấu chấm (.) để định nghĩa 2 parameters

```
app.get('/flights/:from-:to', function(req, res) {
  res.send(req.params)
})
```



Route paramters

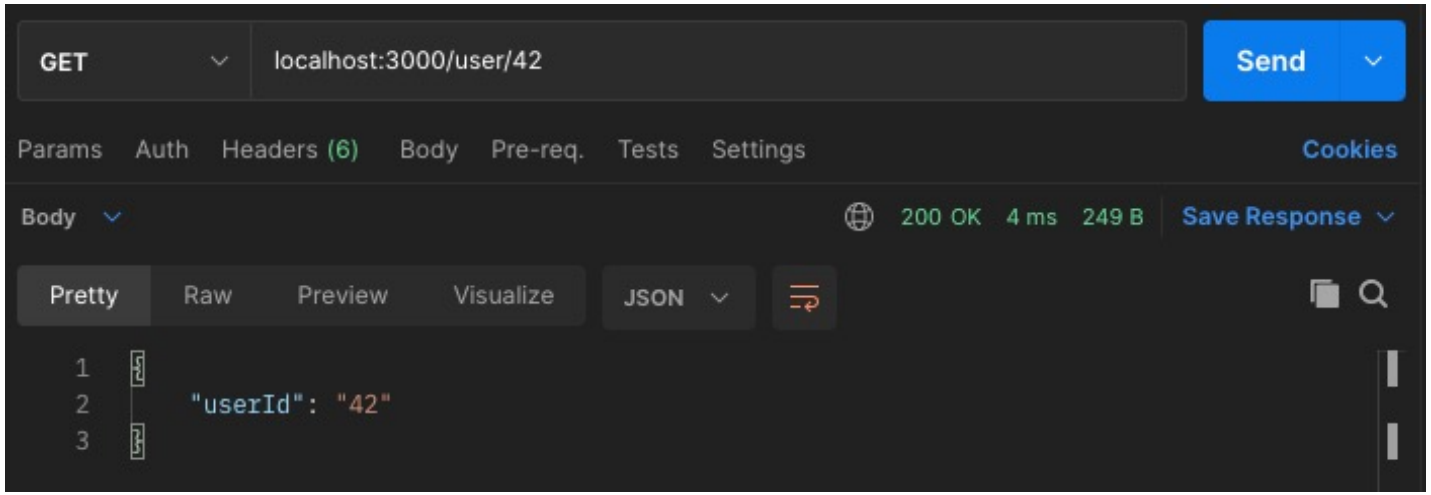
```
app.get('/plantae/:genus.:species', function(req, res) {
  res.send(req.params)
})
```



Route paramters

- Sử dụng dấu ngoặc ( ) để định nghĩa 2 parameters với biểu thức Regular Expression

```
app.get('/user/:userId(\\d+)', function(req, res) {
  res.send(req.params)
})
```



4. Route handlers

Route handler

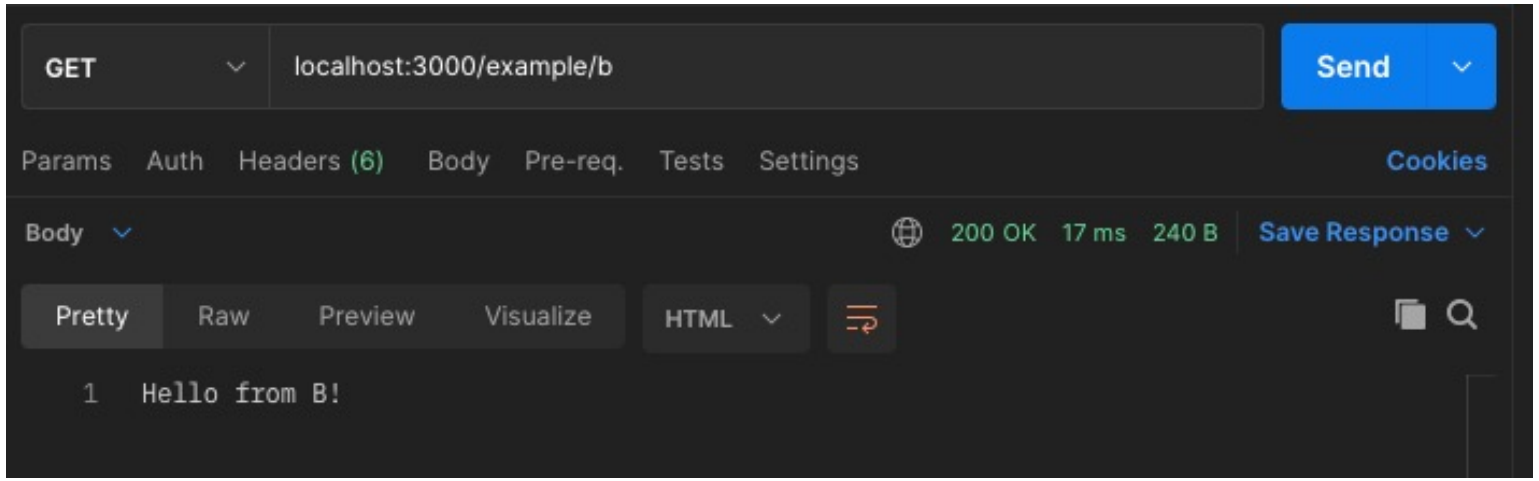
- Chúng ta có thể định nghĩa nhiều hàm callback để xử lý một request, giống như là định nghĩa một middleware
- Để thực hiện truyền việc xử lý cho các hàm callback phía sau thông qua (route) với lệnh ***next***
- Route handler có thể cài đặt một function, một array function, hoặct kết hợp cả 2
- Một callback function xử lý một route

```
app.get('/example/a', function (req, res) {
  res.send('Hello from A!')
})
```

Route handler

- Một hoặc nhiều callback function xử lý một route

```
app.get('/example/b', function (req, res, next) {
  console.log('the response will be sent by the next function ...')
  next()
}, function (req, res) {
  res.send('Hello from B!')
})
```



Route handler

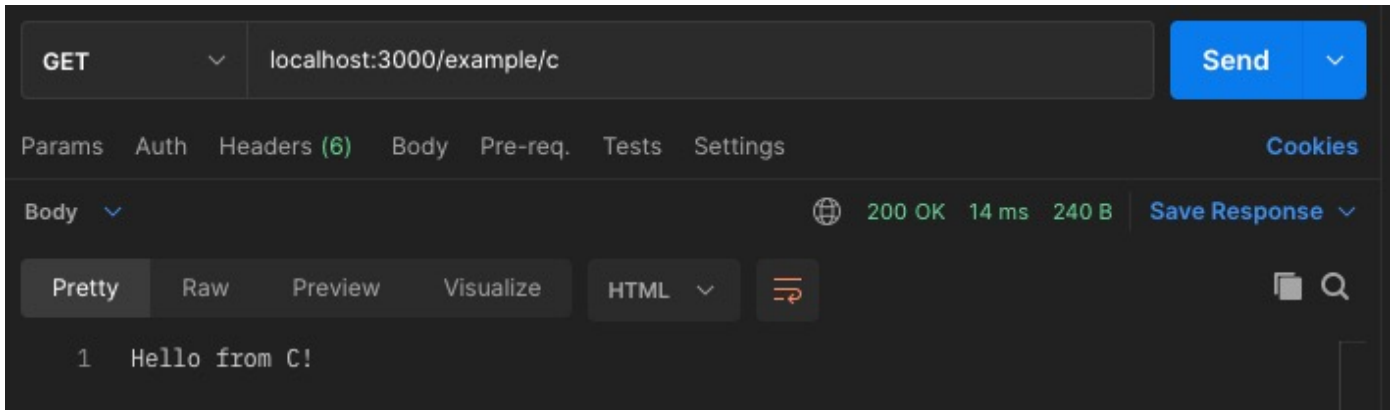
- Mảng các callback function xử lý một route

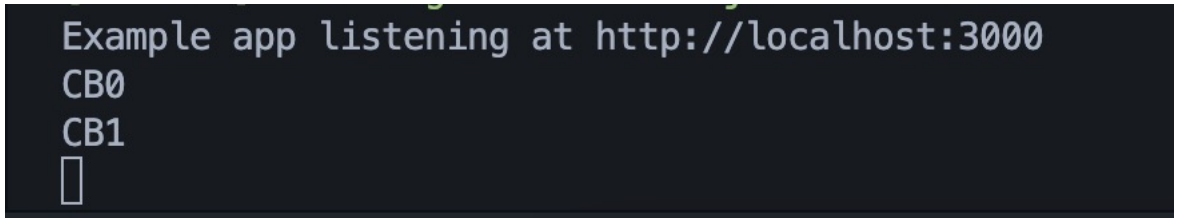
```
var cb0 = function (req, res, next) {
  console.log('CB0')
  next()
}

var cb1 = function (req, res, next) {
  console.log('CB1')
  next()
}

var cb2 = function (req, res) {
  res.send('Hello from C!')
}

app.get('/example/c', [cb0, cb1, cb2])
```





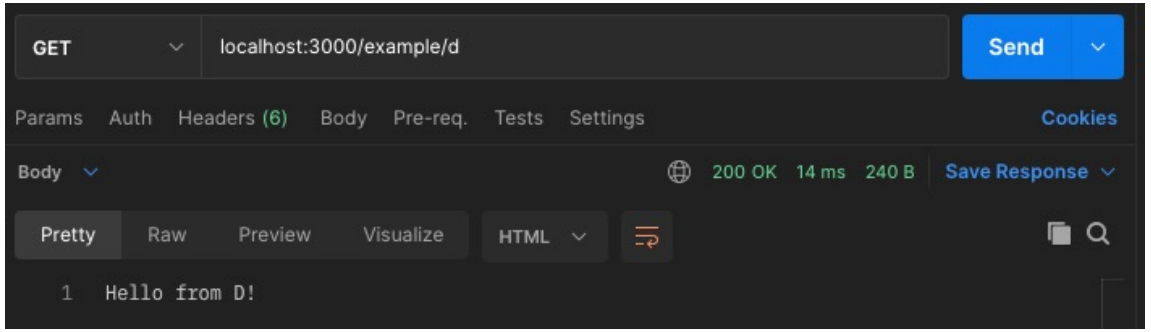
Route handler

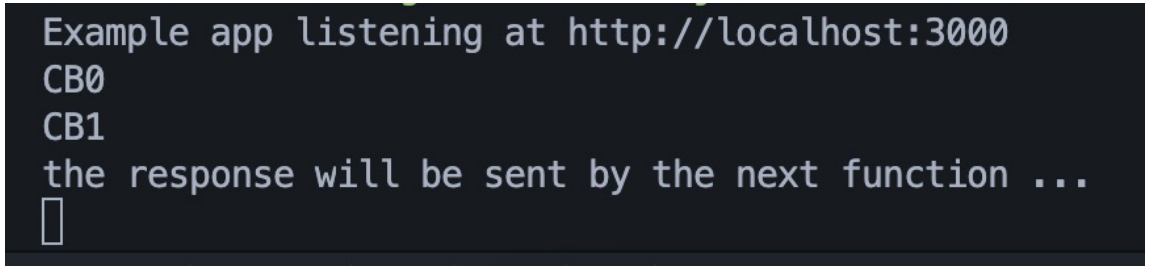
- Hoặc kết hợp giữa một callback function với một array callback function

```
var cb0 = function (req, res, next) {
  console.log('CB0')
  next()
}

var cb1 = function (req, res, next) {
  console.log('CB1')
  next()
}

app.get('/example/d', [cb0, cb1], function (req, res, next) {
  console.log('the response will be sent by the next function ...')
  next()
}, function (req, res) {
  res.send('Hello from D!')
})
```





Response methods

- Các phương thức của response object

Method	Công dụng
res.*download()*	Cho phép download một file
res.*end()*	Kết thúc quá trình xử lý của một response
res.*json()*	Trả về một response dạng JSON
res.*jsonp()*	Trả về một response dạng JSONP
res.*redirect()*	Chuyển request tới một endpoint mới
res.*render()*	Render (hiển thị) một view template (thường được sử dụng Server Side Rendering)
res.*send()*	Trả về một response với nhiều loại khác nhau
res.*sendFile()*	Trả về một file dưới dạng Stream
res.*sendStatus()*	Trả về status code và chuỗi biểu diễn của Status này trong response body

app.route()

- Express hỗ trợ một method là **app**.*route*() để xử lý route

```
app.route('/book')
  .get(function (req, res) {
    res.send('Get a random book')
  })
  .post(function (req, res) {
    res.send('Add a book')
  })
  .put(function (req, res) {
    res.send('Update the book')
  })
```

express.Router

- Sử dụng class express.Router để tạo các module với route handlers
- Một thể hiện Router là một middleware hoàn chỉnh và routing system
- Ví dụ sau đây sẽ
  - Tạo một router như là module
  - Load một hàm middleware
  - Định nghĩa các route
  - Ánh xạ các router module từ đường dẫn của app chính

express.Router – Example

birds.js > ...

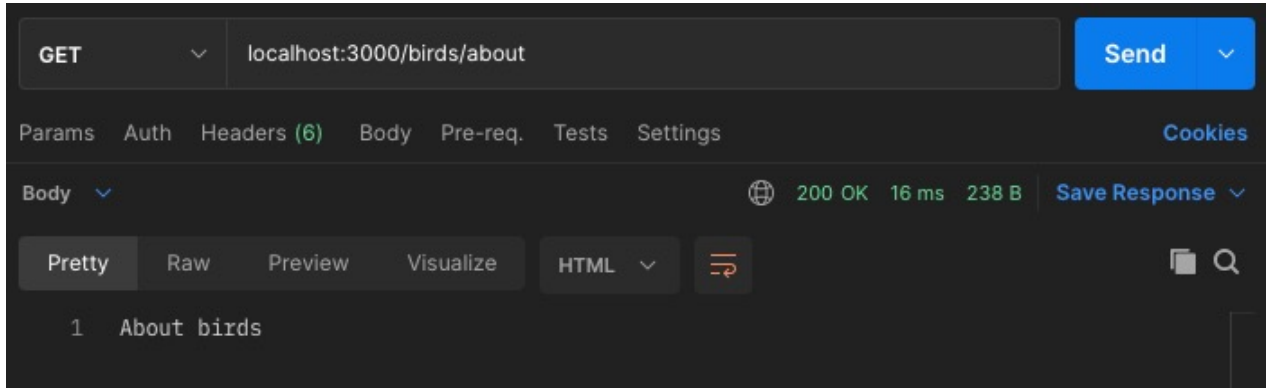
```
1   var express = require('express')
2   var router = express.Router()
3
4   // middleware that is specific to this router
5   router.use(function timeLog (req, res, next) {
6     console.log('Time: ', Date.now())
7     next()
8   })
9   // define the home page route
10  router.get('/', function (req, res) {
11    res.send('Birds home page')
12  })
13  // define the about route
14  router.get('/about', function (req, res) {
15    res.send('About birds')
16  })
17
18  module.exports = router
```

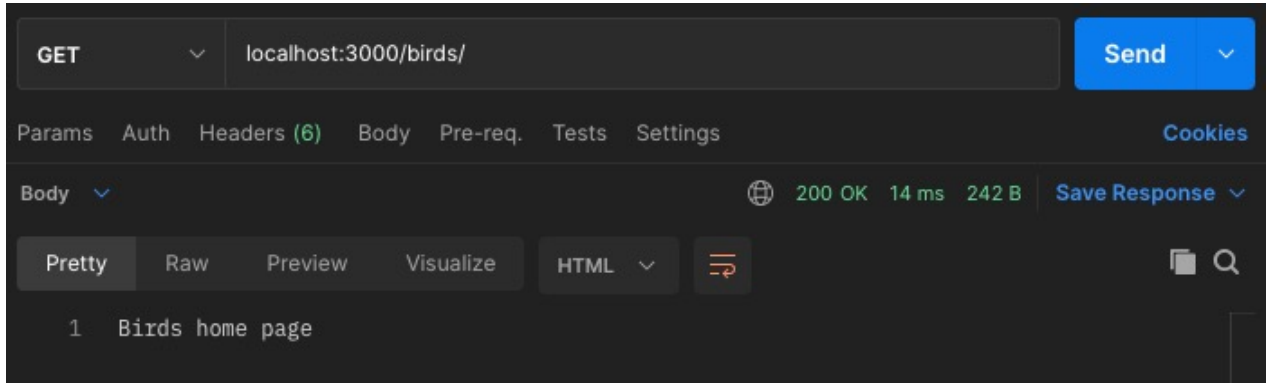
express.Router – Example - Load router module trong App

main.js > ...

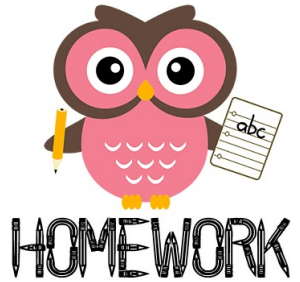
```
1  const express = require('express')
2  const app = express()
3  const port = 3000
4
5  var birds = require('./birds')
6
7  app.use('/birds', birds)
8
9  app.listen(port, () => {
10   console.log(`Example app listening at http://localhost:${port}`)
11 })
```

```
❯ node main.js
Example app listening at http://localhost:3000
Time:  1627334121978
Time:  1627334191003
```





Bài tập về nhà



- Xây dựng một ứng dụng Express đơn giản cho phép thực hiện các yêu cầu sau:
  - Định nghĩa file data.json có cấu trúc tương tự như bài trước
  - Xây dựng Product router để xử thực hiện các xử lý như sau:
    - Liệt kê đầy đủ/tất cả các object có trong data.json (2 điểm)
    - Trả về một object có ID được truyền vào thông qua route parameter (2 điểm)
    - Tạo một một object và ghi vào file data.json (2 điểm)
    - Cập nhập thông cho một object với ID được truyền thông qua route parameter (2 điểm)
    - Xoá một object với ID được truyền vào thông quá route parameter (2 điểm)