

[Github URL](#)

[Github URL for Vercel](#)

[Vercel URL](#)

Video: W11-P1: Implement route /api/shop_xx/:category in server

=> create tables category2_xx and shop2_xx using SQL and set foreign key of cat_id referencing to cid in category2_xx

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows the database schema. A red box highlights the **Columns (6)** section for the **shop2_02** table.
- Query Editor:** Displays a query: `SELECT * FROM public.shop2 ORDER BY pid ASC`.
- Create - Foreign key dialog:** This dialog is open and shows the configuration for creating a foreign key.
 - General tab:** Local column is set to "Select an item..." and References is set to "Select an item...".
 - Columns tab:** A table lists columns: pid [PK] integer and pname character varying (255). Rows 1 through 14 show various jacket names.
 - Referencing tab:** A table defines the foreign key mapping:

Local	Referenced	Referenced Table
cat_id	cid	public.category2_02
- Status Bar:** Shows "Total rows: 35" and "Query complete 00:00:00.085".
- System Tray:** Shows icons for battery, system, and network.
- Bottom Bar:** Includes icons for file operations, search, and other tools.

=> update and delete constraints of the foreign key

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows a tree view of database objects. Under "Tables (2)", "shop2_02" is selected, and under it, "Columns (6)" is expanded.
- Dependents:** A tab showing the dependencies of the current object.
- Create - Foreign key:** A modal dialog for creating a foreign key constraint. It has tabs for General, Definition, Columns, Action, and SQL. The Action tab is active, showing:
 - On update:** CASCADE
 - On delete:** SET NULL
- Query Editor:** Displays a query: `SELECT * FROM public.shop2_02 ORDER BY pid ASC`.
- Data Output:** A table showing data from the shop2_02 table:

pid	pname
1	Brown Brim
2	Blue Beanie
3	Brown Cowboy
4	Grey Brim
5	Green Beanie
6	Palm Tree Cap
7	Red Beanie
8	Wolf Cap
9	Blue Snapback
10	Black Jean Shearling
11	Blue Jean Jacket
12	Grey Jean Jacket
13	Brown Shearling
14	Tan Trench

At the bottom of the pgAdmin window, there is a taskbar with various icons and status information.

=> Chrome, show /api/shop_xx/womens with code (you need to use last two digits of your id)

The screenshot shows a development setup with two main components:

- Left Side (Code Editor):** A code editor window titled "server.js" displays the following Node.js code. A red box highlights the section where the "/api/shop2_02/:category" route is defined.

```
7 app.use(cors())
8
9
10 app.use('/api/blog_02', async (req, res, next) => {
11   const results = await db.query(`select * from blog_02`)
12   console.log(`results`, JSON.stringify(results.rows))
13   res.json(results.rows)
14 }
15
16 app.use('/api/shop2_02/:category', async (req, res, next) => {
17   console.log(`category`, req.params.category)
18   const results = await db.query(
19     `select * from category2_02, shop2_02 where cname = $1 and cid = cat_id`, [req.params.category]
20   )
21   // console.log(`results`, JSON.stringify(results.rows))
22   res.json(results.rows)
23 }
24
25 app.use('/api/shop2_02', async (req, res, next) => {
26   const results = await db.query(`select * from shop2_02`)
27   console.log(`results`, JSON.stringify(results.rows))
28   res.json(results.rows)
29 }
30
31
```

- Right Side (Browser):** A browser window titled "localhost:5001/api/shop2_02/womens" shows a JSON response. A red box highlights the entire response body. The JSON object contains details about a product category named "womens".

```
{
  "cid": 4,
  "cname": "womens",
  "size": "large",
  "image_url": "https://i.ibb.co/GCCdy8t/womens.png",
  "remote_image_url": "/images/midterm/homepage/womens.png",
  "link_url": "/demo/shop_xx/node/womens",
  "pid": 23,
  "pname": "Blue Tanktop",
  "cat_id": 4,
  "price": 25,
  "img_url": "/images/midterm/womens/blue-tank.png",
  "remote_img_url": "https://i.ibb.co/7CQVJNm/blue-tank.png"
},
[ 12 items ],
[ 12 items ]]
```

=> Chrome, show /api/shop_xx/mens with code (you need to use last two digits of your id)

The screenshot shows a development setup with two main windows. On the left is a code editor with a file named `server.js` containing Node.js server code. On the right is a browser window displaying a JSON response from a local host API endpoint.

Code Editor (server.js):

```
JS server.js  TS globals.d.ts  .env
server_02 > JS server.js ...
7
8 app.use(cors())
9
10 app.use('/api/blog_02', async (req, res, next) => {
11   const results = await db.query(`select * from blog_02`)
12   console.log(`results`, JSON.stringify(results.rows))
13   res.json(results.rows)
14 }
15
16 app.use('/api/shop2_02/:category', async (req, res, next) => {
17   console.log(`category`, req.params.category)
18   const results = await db.query(
19     `select * from category2_02, shop2_02 where cname = $1 and cid = cat_id`, [req.params.category]
20   )
21   // console.log(`results`, JSON.stringify(results.rows))
22   res.json(results.rows)
23 }
24
25
26 app.use('/api/shop2_02', async (req, res, next) => {
27   const results = await db.query(`select * from shop2_02`)
28   console.log(`results`, JSON.stringify(results.rows))
29   res.json(results.rows)
30 })
```

Browser Response (localhost:5001/api/shop2_02/mens):

```
[{"cid": 5, "cname": "mens", "size": "large", "image_url": "https://i.ibb.co/R70vBrQ/men.png", "remote_image_url": "/images/midterm/homepage/mens.png", "link_url": "/demo/shop_xx/node/mens", "pid": 30, "pname": "Camo Down Vest", "cat_id": 5, "price": 325, "img_url": "/images/midterm/mens/camo-vest.png", "remote_img_url": "https://i.ibb.co/xJS0T3Y/camo-vest.png"}, {"12 items"}, {"12 items"}, {"12 items"}, {"12 items"}, {"12 items"}]
```

039b62f apple550678

2025-12-01 00:26:57 +0800

Video: W11-P1: Implement route /api/shop

Video: W11-P2: Implement /demo/shop_xx/node and /demo/shop/node/:category in client

=> show how to get category from params

The screenshot shows a developer's environment with the following components:

- Code Editor (VS Code):** The file `page.jsx` is open, showing code related to fetching shop data by category. A red box highlights the line `const [shop_02, setShop_02] = useState([])`. Another red box highlights the line `const params = useParams()`.
- Browser:** A browser window displays a static page titled "Womens". The URL is `localhost:3000/demo/shop_02/no...`. The page content includes "P1_StaticPage_02," and "李國蘋, 213410102". Below this, there is a grid of four product images: "Blue Tanktop 25", "Floral Blouse 20", "Floral Dress 80", and "Red Dress".
- Terminal:** The terminal shows the command `in 42ms (compile: 15ms, render: 27ms)`.
- File Explorer:** Shows the project structure under `W11_NEXT_FULLSTACK_02`, including files like `layout.js`, `page.jsx`, and `page.js`.
- Search Bar:** A search bar at the bottom of the screen.
- System Tray:** Shows battery level (0%), signal strength, and date/time (下午 11:27 2025/11/30).

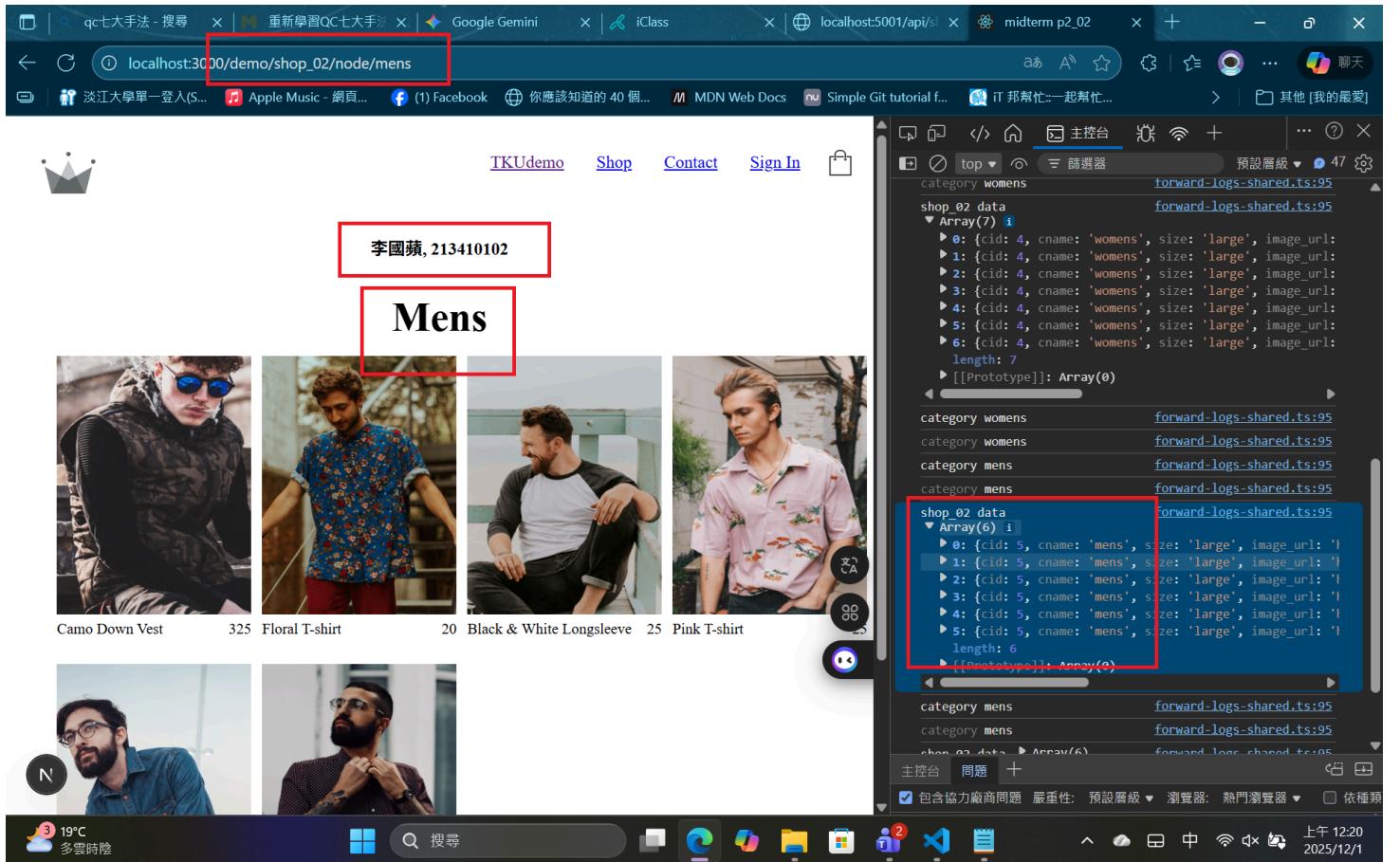
=> show how to fetch category from the category main page

The screenshot displays a developer's environment with several windows open:

- Code Editor:** Shows the file `page.jsx` with code related to fetching shop data by category. A red box highlights the fetch request line:

```
fetch(`http://localhost:5001/api/shop_02/${category}`)
```
- Browser Preview:** Shows a shopping website with a navigation bar for TKUdemo, Shop, Contact, Sign In, and a user profile. Below the navigation is a section titled "P1_StaticPage_02," which displays the text "李國蘋, 213410102". The main content area is titled "Womens" and shows four items: "Blue Tanktop 25", "Floral Blouse 20", "Floral Dress 80", and "Red Dress". Below these are three smaller images.
- Developer Tools:** The right side of the screen features the React DevTools extension for Chrome. It shows the component tree and state for two categories: "womens" and "mens".
 - Category Womens:** Contains 7 items, each with a cid and cname.
 - 0: {cid: 4, cname: 'womens'}
 - 1: {cid: 4, cname: 'womens'}
 - 2: {cid: 4, cname: 'womens'}
 - 3: {cid: 4, cname: 'womens'}
 - 4: {cid: 4, cname: 'womens'}
 - 5: {cid: 4, cname: 'womens'}
 - 6: {cid: 4, cname: 'womens'}
 - Category Mens:** Contains 6 items, each with a cid and cname.
 - 0: {cid: 5, cname: 'mens'}
 - 1: {cid: 5, cname: 'mens'}
 - 2: {cid: 5, cname: 'mens'}
 - 3: {cid: 5, cname: 'mens'}
 - 4: {cid: 5, cname: 'mens'}
 - 5: {cid: 5, cname: 'mens'}

=> Chrome, show FetchShopByCategory_xx.jsx by click Mens



=> relevant code for FetchShopByCategory_xx

The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows the project structure under "W11_NEXT_FULLSTACK_02". A red box highlights the "demo\shop_02\node\[category]" folder, which contains "page.jsx", "layout.jsx", and "page.js".
- Editor:** The "page.jsx" file is open. A red box highlights the code block for "FetchShopByCategory_02".
- Code Block:** The highlighted code is as follows:

```
const FetchShopByCategory_02 = () => {
  const [shop_02, setShop_02] = useState([])
  const params = useParams()
  const category = params.category
  console.log('category', category)

  const fetchShopFromNode = async () => {
    try {
      const response = await fetch(`http://localhost:5001/api/shop2_02/${category}`)
      const data = await response.json()
      console.log('shop_02 data', data)
      if (data.length !== 0) {
        setShop_02(data)
      }
    } catch (err) {
      console.log(err)
    }
  }

  useEffect(() => {
    if (category) {
      fetchShopFromNode()
    }
  })
}

return (
  <Wrapper>
  <div className='shop-page'>
    <div className='section-title'>
      <h4 className='text-center'>李國蘋, 213410102 </h4>
    </div>
    <div className='collection-page'>
      <h1 className='title'>{category}</h1>
      <div className='items'>
        {shop_02?.map((item) => {
          const { pid, pname, price, img_url } = item
          return (
            <Product_02
              key={pid}
              img_url={img_url}
              pname={pname}
              price={price}
            />
          )
        })}
      </div>
    </div>
  </div>
</Wrapper>
)

export default FetchShopByCategory_02
```

The code implements a function component `FetchShopByCategory_02` that uses `useEffect` to fetch data from an API endpoint based on the `category` parameter. It then renders a collection of products using `Product_02` components.

781f563 apple550678

2025-12-01 00:32:04 +0800

Video: W11-P2: Implement /demo/shop_xx/r