

8.3D: Writing a Multi-Route Template Server Application

Tasks

In this task your objective is to create a server application that can serve *dynamic pages* using a template engine within Express, in this case Encapsulated JS (**EJS**).

The server provides a simple interface to access and display data on dKin Cap sales, including details on who the salesperson was, when it was purchased and for how much. You will be provided with sample (very *simple*) data in the form of `JSON` files and will include:

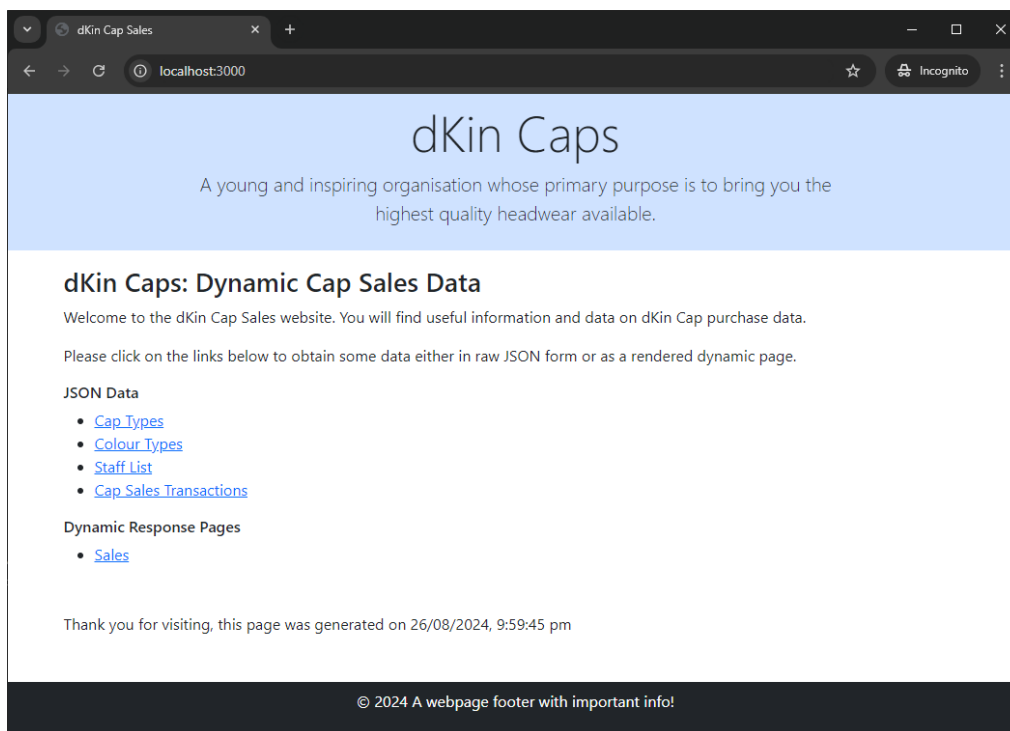
- dKin Cap details
- Cap Colour types
- Sales Staff details
- Sales transaction details
 - Cap type purchased (indexed)
 - Sales staff ID
 - Purchased date
 - A list (array) of items purchased

The routes your server should support include:

- `http://localhost:3000/`
- `http://localhost:3000/captypes` [returns JSON data]
- `http://localhost:3000/colourtypes` [returns JSON data]
- `http://localhost:3000/stafflist` [returns JSON data]
- `http://localhost:3000/transactions` [returns JSON data]
- `http://localhost:3000/sales`

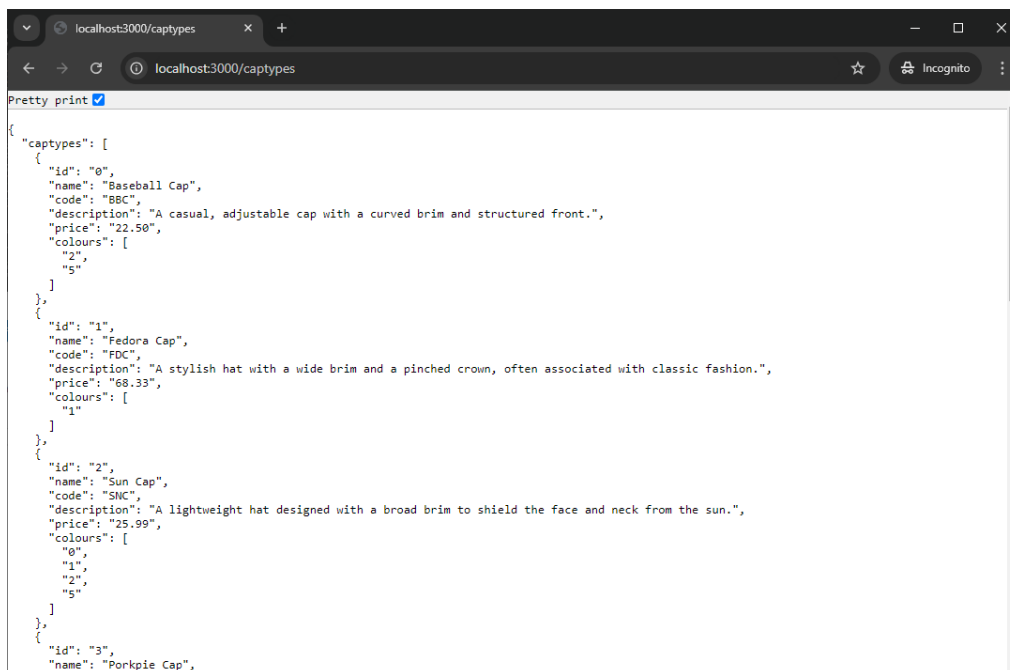
The default `/` (home) and the `/sales` routes should both render dynamic pages, while the remaining routes all return raw `JSON` data for their respective data files.

An example of the output from the default `/` (home) page is shown below (**NOTE:** The date/time shown on the page is dynamic):



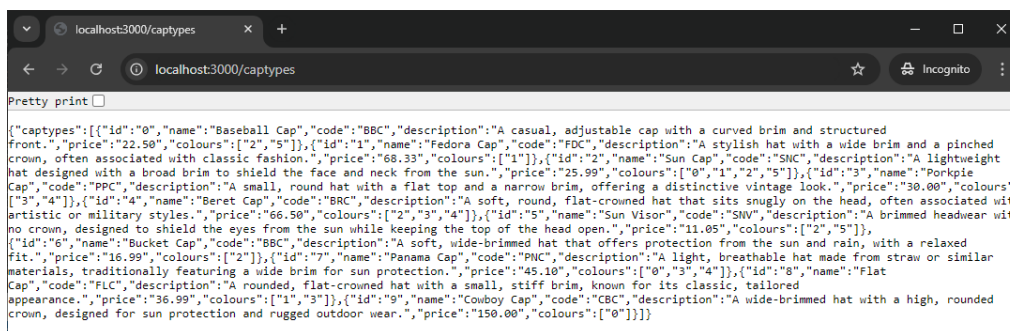
Task8.3.1 Node.js application serving the / route

The routes that return raw JSON data should display output similar to that shown below:



Task8.3.2 JSON response to the /captypes route - formatted output

Although, if your browser doesn't automatically display formatted/indented JSON, it may appear as plain text, as below:



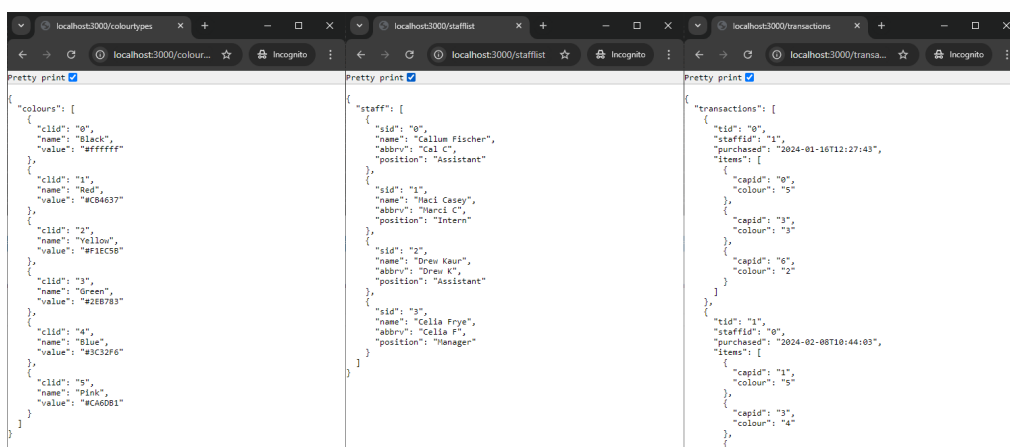
```

{"captypes":[{"id":"0","name":"Baseball Cap","code":"BBC","description":"A casual, adjustable cap with a curved brim and structured front.", "price": "22.50", "colours": ["2", "5"]}, {"id":"1","name":"Fedora Cap","code":"FDC","description":"A stylish hat with a wide brim and a pinched crown, often associated with classic fashion.", "price": "68.33", "colours": ["1"]}, {"id":"2","name":"Sun Cap","code":"SNC","description":"A lightweight hat designed with a broad brim to shield the face and neck from the sun.", "price": "25.99", "colours": ["0", "1", "2", "5"]}, {"id":"3","name":"Porkpie Cap","code":"PPC","description":"A small, round hat with a flat top and a narrow brim, offering a distinctive vintage look.", "price": "30.00", "colours": ["3", "4"]}, {"id":"4","name":"Beret Cap","code":"BRC","description":"A soft, round, flat-crowned hat that sits snugly on the head, often associated with artistic or military styles.", "price": "66.50", "colours": ["2", "3", "4"]}, {"id":"5","name":"Sun Visor","code":"SNV","description":"A brimmed headwear with no crown, designed to shield the eyes from the sun while keeping the top of the head open.", "price": "11.05", "colours": ["2", "5"]}, {"id":"6","name":"Bucket Cap","code":"BBC","description":"A soft, wide-brimmed hat that offers protection from the sun and rain, with a relaxed fit.", "price": "16.99", "colours": ["2"]}, {"id":"7","name":"Panama Cap","code":"PNC","description":"A light, breathable hat made from straw or similar materials, traditionally featuring a wide brim for sun protection.", "price": "45.10", "colours": ["0", "3", "4"]}, {"id":"8","name":"Flat Cap","code":"FLC","description":"A rounded, flat-crowned hat with a small, stiff brim, known for its classic, tailored appearance.", "price": "36.99", "colours": ["1", "3"]}, {"id":"9","name":"Cowboy Cap","code":"CBC","description":"A wide-brimmed hat with a high, rounded crown, designed for sun protection and rugged outdoor wear.", "price": "150.00", "colours": ["0"]}]}

```

Task8.3.2-b JSON response to the /captypes route - unformatted output

Sample output for the two remaining JSON data routes /colourtypes, /stafflist and /transactions are:



```

{"colours": [{"clid": "0", "name": "Black", "value": "ffffff"}, {"clid": "1", "name": "Red", "value": "000000"}, {"clid": "2", "name": "Yellow", "value": "000000"}, {"clid": "3", "name": "Green", "value": "000000"}, {"clid": "4", "name": "Blue", "value": "000000"}, {"clid": "5", "name": "Pink", "value": "000000"}]}

{"staff": [{"sid": "0", "name": "Callum Fischer", "abbrv": "Cel C", "position": "Assistant"}, {"sid": "1", "name": "Haci Casey", "abbrv": "Haci C", "position": "Intern"}, {"sid": "2", "name": "Drew Kaur", "abbrv": "Drew K", "position": "Assistant"}, {"sid": "3", "name": "Celia Frye", "abbrv": "Celia F", "position": "Manager"}]}

{"transactions": [{"tid": "0", "staffid": "1", "purchased": "2024-01-10T12:27:43", "items": [{"capid": "0", "colour": "5"}, {"catid": "3", "colour": "3"}, {"catid": "6", "colour": "2"}]}, {"tid": "1", "staffid": "0", "purchased": "2024-02-08T10:44:03", "items": [{"catid": "1", "colour": "5"}, {"catid": "3", "colour": "4"}]}]}

```

Task8.3.3 JSON response to the /colourtypes, /stafflist and /transactions - formatted output

The final route /sales should render a dynamic page that combines all four data resources into a list of purchase transactions and present the details of these neatly on the page. A sample of how this may be presented is shown in the screenshot below:

dKin Cap Sales

localhost:3000/sales

Incognito

dKin Caps

A young and inspiring organisation whose primary purpose is to bring you the highest quality headwear available.

Purchases Made

Transaction #:	Date/Time:	Salesperson:
0	16/01/2024 at 12:27:43 pm	Marci C (Intern)
Item #:	Description:	Price:
1	Baseball Cap (BBC, Pink) A casual, adjustable cap with a curved brim and structured front.	\$22.50
2	Porkpie Cap (PPC, Green) A small, round hat with a flat top and a narrow brim, offering a distinctive vintage look.	\$30.00
3	Bucket Cap (BBC, Yellow) A soft, wide-brimmed hat that offers protection from the sun and rain, with a relaxed fit.	\$16.99
		\$69.49

Transaction #:	Date/Time:	Salesperson:
1	08/02/2024 at 10:44:03 am	Cal C (Assistant)
Item #:	Description:	Price:
1	Fedora Cap (FDC, Pink) A stylish hat with a wide brim and a pinched crown, often associated with classic fashion.	\$68.33
2	Porkpie Cap (PPC, Blue) A small, round hat with a flat top and a narrow brim, offering a distinctive vintage look.	\$30.00
3	Beret Cap (BRC, Yellow) A soft, round, flat-crowned hat that sits snugly on the head, often associated with artistic or military styles.	\$66.50
		\$164.83

Transaction #:	Date/Time:	Salesperson:
2	20/03/2024 at 2:43:22 pm	Cal C (Assistant)
Item #:	Description:	Price:
1	Fedora Cap (FDC, Red) A stylish hat with a wide brim and a pinched crown, often associated with classic fashion.	\$68.33
2	Porkpie Cap (PPC, Green) A small, round hat with a flat top and a narrow brim, offering a distinctive vintage look.	\$30.00

Task8.3.4 Dynamically rendered page to the /sales route

Transaction #:	Date/Time:	Salesperson:
2	20/03/2024 at 2:43:22 pm	Cal C (Assistant)
Item ?:	Description:	Price:
1	Fedora Cap (FDC, Red) A stylish hat with a wide brim and a pinched crown, often associated with classic fashion.	\$68.33
2	Porkpie Cap (PPC, Green) A small, round hat with a flat top and a narrow brim, offering a distinctive vintage look.	\$30.00
		\$98.33
3	20/03/2024 at 1:00:25 pm	Drew K (Assistant)
Item ?:	Description:	Price:
1	Cowboy Cap (CBC, Black) A wide-brimmed hat with a high, rounded crown, designed for sun protection and rugged outdoor wear.	\$150.00
		\$150.00
4	15/06/2024 at 1:04:25 pm	Marci C (Intern)
Item ?:	Description:	Price:
1	Sun Visor (SNV, Pink) A brimmed headwear with no crown, designed to shield the eyes from the sun while keeping the top of the head open.	\$11.05
2	Panama Cap (PNC, Black) A light, breathable hat made from straw or similar materials, traditionally featuring a wide brim for sun protection.	\$45.10
		\$56.15

© 2024 A webpage footer with important info!

Task8.3.5 Dynamically rendered page to the /sales route (page 2)

Resources Provided

The following resources are provided for this task:

- The EJS template files used to render the default / (home) route:
 - index.ejs
 - header.ejs - The *include* files used in index.ejs
 - footer.ejs - The *include* files used in index.ejs
 - error.ejs - Skeleton error template file
- The raw JSON data file for:
 - captypes.json - The Cap details
 - colours.json - Details of the cap colours
 - staff.json - Sales staff details
 - trasnactions.json - List of the transactions

Steps

Follow the steps below to complete this task:

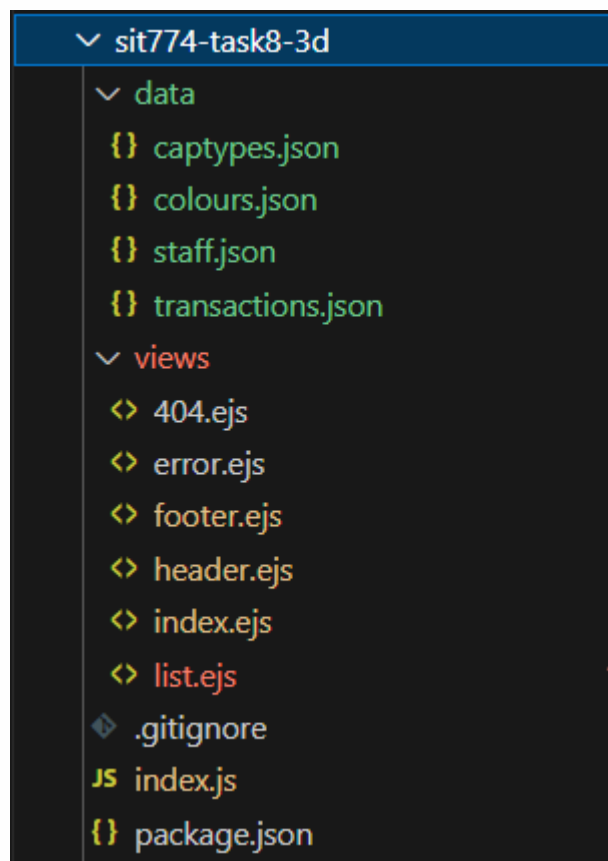
1. Create your own local directory (or you can use the same local directory

created in Task 8.1).

2. Add a new Node Module package to install the template engine for express supporting *Encapsulated JS* or *ejs*

- `npm install ej`

3. Create a new folder in your project called `data`, to hold the `json` data files
4. Create a new folder in your project called `views`, to hold the `ejs` template file to render your output pages
5. Your directory structure should be as shown below:



Task8.3.6 Project directory structure

6. Use express module to create a Node.js application (i.e., `index.js`) that will set a local web server. The server listens to the port **3000**.
7. Add to your `index.js` server the access to the `path` module (already installed along with the `express` module):

```
const path = require('path'); // Added to support access to file system paths
```

8. Add variables (constants) to access the `json` data provided for the cap types, colours, staff and transactions:

```
const jsonCapTypeData = require(path.join(__dirname, 'data/captypes'));  
const jsonColourData = require(path.join(__dirname, 'data/colours'));  
const jsonStaffData = require(path.join(__dirname, 'data/staff'));  
const jsonTransactionData = require(path.join(__dirname, 'data/transactions'));
```

9. Add and configure the EJS template engine:

```
// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');
```

10. Add a route handler for a GET request on the path / to respond with a rendered page from the *EJS template* index.ejs using the res.render() command. Note the arguments provided include the name of the template to use and the data to use inside the template as the second parameter, such as:

```
app.get('/', (req, res, next) => {
  res.render('index', { title: 'dkin Cap Sales' });
});
```

11. Add the route handlers to respond with the JSON data files using the res.json()

```
app.get('/captertypes', (req, res, next) => {
  res.json(jsonCapTypeData);
});

... // add others for '/colourtypes', '/stafflist' and '/transactions'
```

12. Create and build a new template for the page to display the list of feedback comments showing the users and their ratings. This should be held in a list.ejs file.

13. Add the route handler for /sales to render a template list.ejs using the data from the JSON files. It could be called with multiple parameters, such as:

```
app.get('/sales', (req, res, next) => {
  res.render('list', {
    title: 'dkin Cap Sales',
    transactions: jsonTransactionData.transactions,
    types: jsonCapTypeData.captertypes,
    colours: jsonColourData.colours,
    staff: jsonStaffData.staff
  });
});
```

14. Run your project npm run start:dev in a Command Prompt (Windows) or Terminal (Mac OS) within your local directory to start the server.

15. Open a web browser and use the address to test the routes you have implemented:

- http://localhost:3000/
 - Should return a rendered web page
- http://localhost:3000/captertypes
 - Should return JSON data
- http://localhost:3000/colourtypes

- Should return JSON data
 - `http://localhost:3000/stafflist`
- Should return JSON data
 - `http://localhost:3000/transactions`
- Should return JSON data
 - `http://localhost:3000/sales`
- Should return a rendered web page

Hints

To complete this task, review the **EJS Documentation** (<https://www.npmjs.com/package/ejs>) on using template elements to display content; especially that on `loops`.

What will you submit?

You should submit:

- Source code of the **server** file `index.js`
- Source code of the **template** file `list.ejs`
 - This needs to be in PDF format as Ontrack doesn't accept files of `.ejs` type
- 6 x Screenshots of the responses to:
 - `http://localhost:3000/`
 - `http://localhost:3000/captypes`
 - `http://localhost:3000/colourtypes`
 - `http://localhost:3000/stafflist`
 - `http://localhost:3000/transactions`
 - `http://localhost:3000/sales`