



</talentlabs>

Express Lecture 2

Introduction to Express



</talentlabs>

Agenda

- What is ExpressJS
- First Express Application
- Observe the HTTP messages
- Express App Project Exploration

What is ExpressJS

</talentlabs>



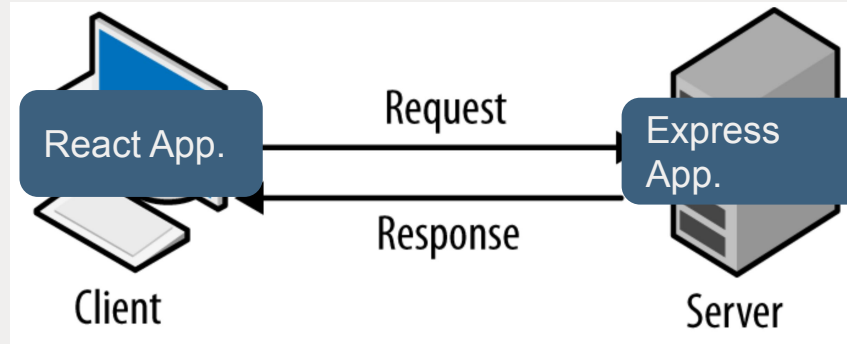
What is Express

Express is a minimal and flexible Node.js web application framework. It allows us to create a robust API quickly.

<https://expressjs.com/>

Express Application

- With the Express framework, we can write Backend Application easily.
- Server is the machine/virtual machine.
- Application is the code running on the that machine.



First Express Application

</talentlabs>



Express Application Generator

Use the application generator tool, express-generator, to quickly create an application skeleton.

Steps:

1. Execute the command: **npx express-generator --ejs first-express-app**
2. Change directory into that new folder: **cd first-express-app**
3. Install all npm packages: **npm install**

Running an Express Application

Start the Express web server by running one of the following commands:

Windows PowerShell:

```
$env:DEBUG='first-express-app:*'; npm start
```

Windows Command Prompt:

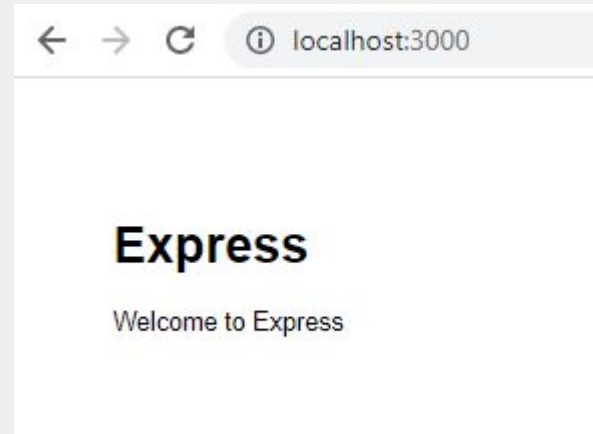
```
set DEBUG=first-express-app:* & npm start
```

MacOs or Linux:

```
SET DEBUG=first-express-app:* & npm start
```

Access the Express application by loading **http://localhost:3000/**

```
first-express-app:server Listening on port 3000 +0ms
```

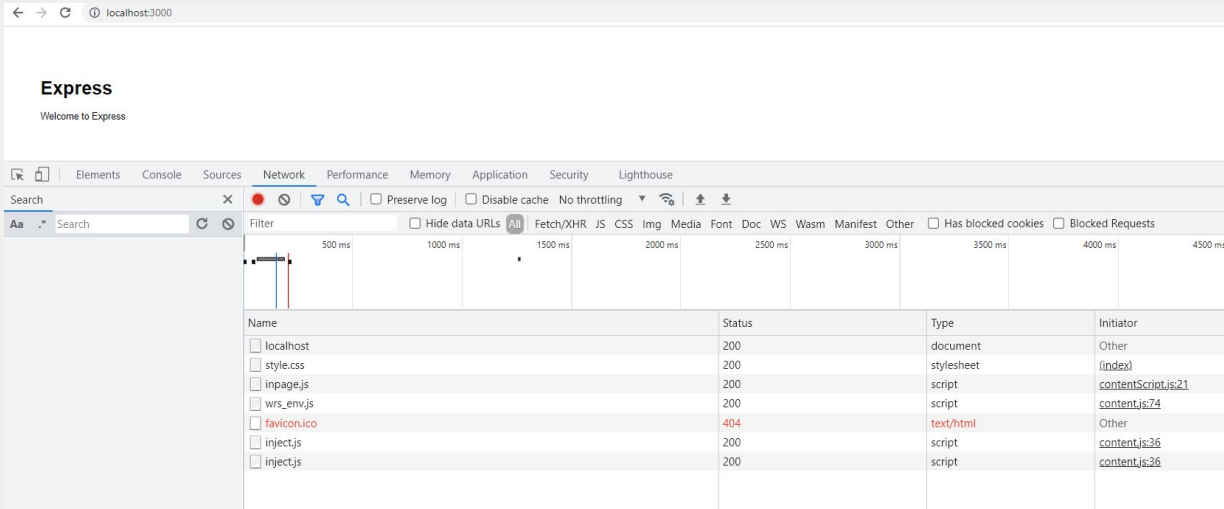


Observe the HTTP messages

</talentlabs>



Observe the HTTP messages with Chrome



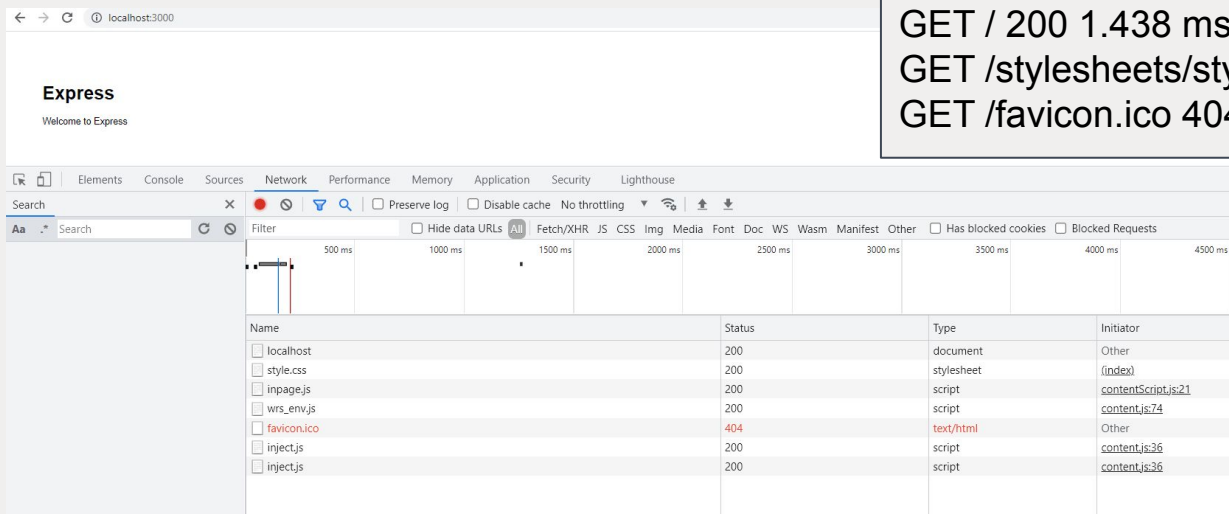
The screenshot shows the Chrome DevTools Network tab. The top panel displays the Express application running on localhost:3000. The bottom panel shows the Network tab with a list of requests. The 'favicon.ico' request is highlighted in red, indicating a 404 status. The table below lists the requests:

Name	Status	Type	Initiator
localhost	200	document	Other
style.css	200	stylesheet	(index)
inpage.js	200	script	contentScript.js:21
wrs_env.js	200	script	content.js:74
favicon.ico	404	text/html	Other
inject.js	200	script	content.js:36
inject.js	200	script	content.js:36

HTML file + CSS File

Observe the HTTP messages with the server log

GET / 200 1.438 ms - 207
GET /stylesheets/style.css 200 1.226 ms - 111
GET /favicon.ico 404 2.531 ms - 1193



- Only **3 files are sending from the server!**
- The other request sent from Chrome is from my Chrome Extension.
- A frontend website can talk to multiple servers...

The HTML Response

Headers Preview Response Initiator Timing Cookies

▼ General

Request URL: http://localhost:3000/

Request Method: GET

Status Code: 200 OK

Remote Address: [::1]:3000

Referrer Policy: strict-origin-when-cross-origin

▼ Response Headers View source

Connection: keep-alive

Content-Length: 207

Content-Type: text/html; charset=utf-8

Date: Fri, 30 Jul 2021 08:59:03 GMT

ETag: W/"cf-sMq3uu/Hzh7Qc54TveG8Dx1BA2U"

Keep-Alive: timeout=5

X-Powered-By: Express

500 ms 1000 ms 1500 ms 2000 ms 2500 ms 3000 ms 3500 ms 4000 ms

Name

- localhost
- style.css
- inpage.js
- wrs_env.js
- favicon.ico
- inject.js
- inject.js

× Headers Preview Response Initiator Timing Cookies

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Express</title>
5     <link rel='stylesheet' href="/stylesheets/style.css' />
6   </head>
7   <body>
8     <h1>Express</h1>
9     <p>Welcome to Express</p>
10  </body>
11 </html>
12
```

GET / 200 1.438 ms - 207

The CSS Response

× Headers Preview Response Initiator Timing Cookies

▼ General

Request URL: http://localhost:3000/stylesheets/style.css

Request Method: GET

Status Code: 🟢 200 OK

Remote Address: [::1]:3000

Referrer Policy: strict-origin-when-cross-origin

▼ Response Headers View source

Accept-Ranges: bytes

Cache-Control: public, max-age=0

Connection: keep-alive

× Headers Preview Response Initiator Timing Cookies

Name

- localhost
- style.css
- inpage.js
- wrs_env.js
- favicon.ico
- inject.js
- inject.js

1 body {
2 padding: 50px;
3 font: 14px "Lucida Grande", Helvetica, Arial, sans-serif;
4 }
5
6 a {
7 color: #00B7FF;
8 }
9

GET /stylesheets/style.css 200 1.226 ms - 111

The .ico file Response

The screenshot displays the browser's developer tools with the 'Response' tab selected. The left sidebar shows a list of files, with 'favicon.ico' highlighted. The main pane shows the response body, which is an HTML document containing a 'Not Found' error message. Below the main pane, the 'Headers' tab is selected, showing the 'General' section with the following details:

- Request URL:** http://localhost:3000/favicon.ico
- Request Method:** GET
- Status Code:** 404 Not Found
- Remote Address:** [::1]:3000
- Referrer Policy:** strict-origin-when-cross-origin

The 'Response Headers' section is also visible, showing 'View source'.

Cannot find this file!!

GET /favicon.ico 404 2.531 ms - 1193

Other Responses (Not from the Express app)

The screenshot displays the Chrome DevTools Network tab. On the left, a list of resources is shown, with 'inpage.js' selected. The main panel on the right shows the details for this request. The 'General' tab is active, displaying the following information:

- Request URL:** chrome-extension://aiifbnfbobpmeeikipheeijmdpn1pgpp/inpage.js
- Request Method:** GET
- Status Code:** 200 OK
- Referrer Policy:** strict-origin-when-cross-origin

Below the 'General' tab, the 'Response Headers' section is expanded, showing:

- Access-Control-Allow-Origin:** *
- cache-control:** no-cache
- Content-Security-Policy:** script-src 'self' 'unsafe-eval'; object-src 'self';
- Content-Type:** text/javascript
- Cross-Origin-Resource-Policy:** cross-origin
- ETag:** "Q9fc6H7otsP7I/9N9ZKiex0q1jM="

The 'Request Headers' section is also expanded, showing a warning that 'Provisional headers are shown' and the following headers:

- Referer:** http://localhost:3000/
- User-Agent:** Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212 Safari/537.36

Express App Project Exploration

</talentlabs>



Project Exploration - app.js

```
...
var app = express();

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

app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());

// static files setup
app.use(express.static(path.join(__dirname, 'public')));
...
```

“app.js” is the **entry point** of the whole application. It mainly contains the **settings of the Express application**:

Project Exploration - /views folder

```
<!DOCTYPE html>
<html>
  <head>
    <title><%= title %></title>
    <link rel='stylesheet'
href='/stylesheets/style.css' />
  </head>
  <body>
    <h1><%= title %></h1>
    <p>Welcome to <%= title %></p>
  </body>
</html>
```

The template file for the index page:

/views/index.ejs:

<%= title %> here is a variable or placeholder that to be replaced at runtime / rendering time.

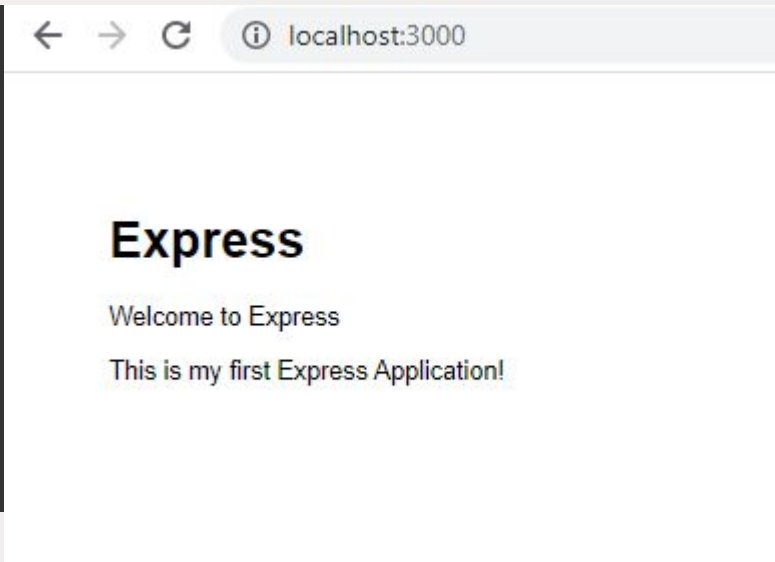
Project Exploration - /views folder

Let's try to update the content of the
/views/index.ejs:

```
<!DOCTYPE html>
<html>
  <head>
    <title><%= title %></title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
  </head>
  <body>
    <h1><%= title %></h1>
    <p>Welcome to <%= title %></p>
    <p>This is my first Express Application!</p>
  </body>
</html>
```

You will need to rerun the Express application.

1. Ctrl-C in the terminal.
2. Run the run server command again.



Project Exploration - /routes folder

The route file for the index route:

/routes/index.js:

```
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res,
next) {
  res.render('index', { title:
'Express' });
});

module.exports = router;
```

“/routes”/ folder stores all the **route files**. In a route file, we **define routing using methods of the Express app object** that correspond to HTTP methods;

for example, app.get() to handle GET requests and app.post to handle POST requests.

Project Exploration - /routes folder

The route file for the index route:

/routes/index.js:

```
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res,
next) {
  res.render('index', { title:
'Express' });
});

module.exports = router;
```

These routing methods **specify a callback function** (sometimes called “handler functions”) called when the application receives a request to the specified route (endpoint) and HTTP method.

In other words, the application “listens” for requests that match the specified route(s) and method(s), and when it detects a match, it calls the specified callback function.

We will talk more about this in the future lectures.

Project Exploration - /routes folder

The route file for the index route:

/routes/index.js:

```
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res,
next) {
  res.render('index', { title:
'Express' });
});

module.exports = router;
```

In the above example, When the Express app receives a GET request to the endpoint “/”, it will render the **template file named “index”** with the **context / state { title: 'Express' }**.

What is the template file named “index”?
It is “/views/index.ejs”.

What is a **rendering context / state**?
It is a JavaScript object that **provides the actual values for the variables / placeholders** in the template file.

Project Exploration - /public folder

“/public” folder stores all the **static files** such as images, CSS files, and JavaScript files.

This is for the contents that don't need logic to process, therefore **no need to go through a route handler function**.

They can be accessed via:

```
http://localhost:3000/stylesheets/style.css
```