



Full-stack Application Development

ExpressJS

Where to Find The Code and Materials?

<https://github.com/iproduct/fullstack-typescript-react>



What Is Express?

- [Express](#) is fast, unopinionated, minimalist web framework for [Node.js](#)

`npm install express --save`

- Allows to build:
 - **Web Applications** - Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.
 - **APIs** - with a myriad of HTTP utility methods and middleware at your disposal, creating a robust API is quick and easy.
- **Performance** - Express provides a layer of fundamental web application features, without obscuring Node.js features that you know and love.
- **Frameworks** - many popular frameworks are based on Express.

Express with TypeScript Simple HTTP Server Example

```
import * as express from 'express';  
import { Request, Response } from 'express';
```

```
const app = express();  
app.get("/", (req: Request, res: Response) => {  
    res.send("Hello World")  
})
```

```
const PORT = process.env.PORT || 3000;  
app.listen(PORT, () => {  
    console.log(`Server is running in http://localhost:${PORT}`)  
})
```

Basic Routing

- **Routing** refers to determining how an application responds to a client request to a particular endpoint, which is a URI (or path) and a specific HTTP request method (GET, POST, and so on).
- Each route can have one or more handler functions, which are executed when the route is matched.
- **Route** definition takes the following structure:
`app.METHOD(PATH, HANDLER)`, where:
 - `app` is an instance of `express`.
 - `METHOD` is an [HTTP request method](#), in lowercase.
 - `PATH` is a path on the server.
 - `HANDLER` is the function executed when the route is matched.
- Ex: Respond to POST request on the root route (`/`), the application's home page:
`app.post('/', function (req, res) { res.send('Got a POST request') })`

Serving Static Files

- To serve static files such as images, CSS files, and JavaScript files, use the `express.static` built-in middleware function in Express:

```
express.static(root, [options])
```

- The `root` argument specifies the root directory from which to serve static assets. For more information on the `options` argument, see [express.static](https://expressjs.com/en/4.x/api.html#express.static).
- Example – serve images, CSS/JS/HTML/etc. files from a directory named `public`:

```
app.use(express.static('public'))
```
- To create a virtual path prefix (where the path does not actually exist in the file system) for files that are served by the `express.static` function, specify a `mount path` for the static directory, as shown below:

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


Route Parameters

- Route parameters are named URL segments that are used to capture the values specified at their position in the URL.
- The captured values are populated in the `req.params` object, with the name of the route parameter specified in the path as their respective keys - Ex:
Route path: `/users/:userId/books/:bookId` Request URL: `http://localhost:3000/users/34/books/8989`
`req.params`: `{ "userId": "34", "bookId": "8989" }`

- To define routes with route parameters, simply specify the route parameters in path:
`app.get('/users/:userId/books/:bookId', function (req, res) {
 res.send(req.params) })`
- The name of route parameters must be made up of “word characters” (`[A-Za-z0-9_]`). Since the hyphen (`-`) and the dot (`.`) are interpreted literally, they can be used along with route parameters for useful purposes.

Route path: `/flights/:from-:to` Request URL: `http://localhost:3000/flights/LAX-SFO` `req.params`: `{ "from": "LAX", "to": "SFO" }`

Route path: `/plantae/:genus.:species` Request URL: `http://localhost:3000/plantae/Prunus.persica`
`req.params`: `{ "genus": "Prunus", "species": "persica" }`

Route Parameters - I

- Route parameters are named URL segments that are used to capture the values specified at their position in the URL.
- The captured values are populated in the req.params object, with the name of the route parameter specified in the path as their respective keys - Ex:

Route path: /users/:userId/books/:bookId

Request URL: http://localhost:3000/users/34/books/8989

req.params: { "userId": "34", "bookId": "8989" }

- To define routes with route parameters, simply specify the route parameters in path:

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


Route Parameters - II

- The name of route parameters must be made up of “word characters” ([A-Za-z0-9_]). Since the hyphen (-) and the dot (.) are interpreted literally, they can be used along with route parameters for useful purposes.

Route path: /flights/:from-:to

Request URL: <http://localhost:3000/flights/LAX-SFO>

req.params: { "from": "LAX", "to": "SFO" }

Route path: /plantae/:genus.:species

Request URL: <http://localhost:3000/plantae/Prunus.persica>

req.params: { "genus": "Prunus", "species": "persica" }

- To have more control over the exact string that can be matched by a route parameter, you can append a regular expression in parentheses (()):

Route path: /user/:userId(\d+)

Request URL: <http://localhost:3000/user/42>

req.params: {"userId": "42"}

Response methods

Method	Description
<code>res.download()</code>	Prompt a file to be downloaded.
<code>res.end()</code>	End the response process.
<code>res.json()</code>	Send a JSON response.
<code>res.jsonp()</code>	Send a JSON response with JSONP support.
<code>res.redirect()</code>	Redirect a request.
<code>res.render()</code>	Render a view template.
<code>res.send()</code>	Send a response of various types.
<code>res.sendFile()</code>	Send a file as an octet stream.
<code>res.sendStatus()</code>	Set the response status code and send its string representation as the response body.

app.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

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

// middleware that is specific to this router
router.use(function timeLog (req, res, next) {
  console.log('Time: ', Date.now())
  next()
})
// define the home page route
router.get('/', function (req, res) {
  res.send('Birds home page')
})
// define the about route
router.get('/about', function (req, res) {
  res.send('About birds')
})
module.exports = router
```

Thank's for Your Attention!



Trayan Iliev

IPT – Intellectual Products & Technologies

<http://iproduct.org/>

<http://robolearn.org/>

<https://github.com/iproduct>

<https://twitter.com/trayaniliev>

<https://www.facebook.com/IPT.EACAD>