## What is Express.js ?

After the big rise of Node.js, there are plenty of frameworks and library that have been created. The most used one for backend rendering is Express. That will be the topic of this Super Skill. So during this chapter we are going to explore:

- What is Express?
- How to setup an Express environment?
- What is routing via Express?
- What are middlewares and what is their use?
- How to work with a template engine?

## What is Express.js ?

The Express website describes Express as a "minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications." What does that really mean, exactly?

Express provides basic tools that facilitate the creation of Node.js applications without obscuring Node.js features that you know and love.

It provides a set of methods for processing HTTP requests and provides a middleware system that extends its functionality.

It also makes it easier to manage the path (URL) of your application and not to mention that it uses templates.

## What is Express.js ?

If you have wrote any serious apps only using the core Node.js modules, you must have likely found yourself reinventing the wheel by repeatedly writing the same code for the same tasks such as:

- Parsing of HTTP request bodies.
- Parsing of cookies.
- Managing sessions.

- Organizing routes with a chain of **if** conditions based on URL paths and HTTP request methods.
- Determining proper response headers based on data types.

## How Does Express Work ?

Express.js usually has an entry point or in other words, a main file. In that file, we can perform the following steps:

1. Include third party dependencies.
2. Configure Express.js app settings such as the template engine and its files’ extensions.
3. Define middlewaress such as error handlers, static files folder, cookies and other parsers.
4. Define routes.
5. Connect to databases such as MongoDB, Redis or MySQL.
6. Start the app.

When Express.js app is running, it listens to requests. Each incoming request is processed through a defined chain of middlewares and routes starting from top to bottom. For example, we can have multiple functions handling each request and some of those functions will be in the middle (hence the name middleware):

1. Parse cookie information and go to the next step when done.
2. Parse parameters from the URL and go to the next step when done.
3. Get the information from the database based on the parameter's value, if a user is authorized (cookie/session), and go to the next step if there is a match.
4. Display the data and end the response.

Previous next