Agenda

- Express
- Mysql

Express

- Developed by many developers around the world
- It is a fast,minimilistic web framework for nodejs
- It is a lightweight and flexible routing framework with minimal core features meant to be augmented through the use of Express middleware modules.

```
# to start using express application install express using npm or add using yarn
npm install express
# OR
yarn add express
```

Advantage of express over http server

http server

- It's the core built-in module used to create HTTP servers.
- Gives low-level control over requests and responses.
- we manually handle Routing (URLs),Headers, Response codes, Parsing JSON, etc.

express server

- A framework built on top of http module.
- Simplifies routing, middleware, JSON handling, templating, etc.
- Makes your code shorter, cleaner, and easier to maintain.

Routing in express

- In a framework like Express, routing is typically implemented by setting up specific endpoints whith combination of paths and methods (like GET, POST, etc.) that will trigger certain functions to handle those requests.
- These specific endpoints are created called as `Routes`
- Each route can have one or more handler functions, which are executed when the route is matched.
- A route is essentially a combination of a URL and a method, and it is where the logic of handling requests resides.
- In express Route definition takes the following structure:

```
app.METHOD(PATH, HANDLER)
```

- Where:

1. app is an instance of express.
2. METHOD is an HTTP request method, in lowercase.
3. PATH is a path on the server.
4. HANDLER is the function executed when the route is matched.

- 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.
- 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.

```
const express = require("express");
const app = express();

// Route for homepage
app.get("/", (req, res) => {
  res.send("Welcome to Home Page");
});

// Route for /about
app.get("/about", (req, res) => {
  res.send("About Us Page");
});

// Route for POST request to /login
app.post("/login", (req, res) => {
  res.send("Login data received");
});

app.listen(3000,'localhost', () => console.log("Server running on port 3000"));
```

express.Router

- express.Router() is a mini Express app, just for handling routes.
- used to create modular, mountable route handlers.
- It lets us break our app into modules so instead of defining all routes in server.js, we can separate them into different files (like users.js,products.js,etc).
- A Router instance is a complete middleware and routing system.

```
// In user.js file

const express = require("express");
const router = express.Router();
// GET /users/
router.get("/", (req, res) => {
  res.send("User List");
});
```

```
// GET /users/:id
router.get("/:id", (req, res) => {
  res.send(`User ID: ${req.params.id}`);
});
module.exports = router;
```

```
// In server.js
const express = require("express");
const app = express();
const userRouter = require("./user");

app.use("/users", userRouter); // Mount user routes at /users

app.listen(3000, () => console.log("Server running on port 3000"));
```

Middleware

- Middleware is a function that runs between the request and the response in a web application.
- It is commonly used in backend frameworks like Express.js to modify, process, or control HTTP requests before sending a response.
- It Access Request (req) and Response (res) Objects
- It Modify Requests/Responses → Example: Adding headers, parsing JSON
- Run Code Before Sending a Response → Example: Authentication, Logging
- Call next() to Continue Execution → Passes control to the next middleware
- It is commonly used in express for

1. Handling CORS (cors package)
2. Parsing JSON data (express.json())
3. Authentication & authorization

Connecting with mysql database

1. add/install the mysql module

```
npm install mysql2
```

2. Create the pool object

```
const pool = mysql.createPool({
    host: 'localhost',
    user: 'root',
    password: 'root',
    database: 'kdac_db'
})
```

3. Call the query method

```
const sql = `SELECT * FROM product`
pool.query(sql, (error, data) => {
    if(data)
        response.send(data)
    else
        response.send(error)
})
```