Nice 🔥 Now we're at **6. Express.js Framework** – the **most widely used framework in Node.js**, and for you, it's like the **Laravel of the Node.js world** (but lighter and unopinionated).

---

6. Express.js Framework

📌 What is Express.js?

- A **minimal and flexible Node.js web framework**.
- Built on top of Node's **http module**.
- Provides powerful features like **routing, middleware, error handling, templating support**.
- Used for **REST APIs, web apps, microservices**.

6.1 Installing Express

```
npm install express
```

6.2 Basic Express Server

```
const express = require('express');
const app = express();
const PORT = 3000;

app.get('/', (req, res) => {
  res.send('Hello Express!');
});

app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

👉 Similar to Laravel routes → `/` maps to a controller-like function.

6.3 Routing in Express

Express provides **HTTP method-specific routes**.

```
// GET
app.get('/users', (req, res) => {
  res.send('Get all users');
});

// POST
```

```
app.post('/users', (req, res) => {
  res.send('Create new user');
});

// PUT
app.put('/users/:id', (req, res) => {
  res.send(`Update user ${req.params.id}`);
});

// DELETE
app.delete('/users/:id', (req, res) => {
  res.send(`Delete user ${req.params.id}`);
});
```

👉 Equivalent to Laravel's `Route::get()`, `Route::post()` etc.

6.4 Middleware in Express

- Middleware = Functions that **execute in request/response cycle**.
- Use cases: logging, auth, body parsing, error handling.

Example:

```
// Global middleware
app.use((req, res, next) => {
  console.log(`${req.method} ${req.url}`);
  next(); // pass control to next middleware
});

// Route-level middleware
function authMiddleware(req, res, next) {
  if (req.headers['authorization']) {
    next();
  } else {
    res.status(401).send('Unauthorized');
  }
}

app.get('/secure', authMiddleware, (req, res) => {
  res.send('Secure Data');
});
```

👉 Similar to Laravel middleware (`Kernel.php`, `auth`, `csrf`).

6.5 Request & Response

Express extends Node's `req` and `res` objects.

```
app.get('/profile/:id', (req, res) => {
  console.log(req.params.id);   // URL param
  console.log(req.query.name);  // Query string
  console.log(req.body);        // Body (needs body-parser/json middleware)

  res.status(200).json({ success: true });
});
```

6.6 Error Handling in Express

- Express has built-in error-handling middleware.
- Error middleware must have **4 arguments** `(err, req, res, next)`.

Example:

```
app.use((err, req, res, next) => {
  console.error(err.stack);
  res.status(500).json({ error: 'Something went wrong!' });
});
```

👉 Similar to Laravel's Exception Handler.

6.7 Express Router

Used to **organize routes** (like Laravel route groups/controllers).

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

router.get('/', (req, res) => res.send('User list'));
router.post('/', (req, res) => res.send('Create user'));

module.exports = router;
```

In main `app.js`:

```
const userRoutes = require('./routes/users');
app.use('/users', userRoutes);
```

6.8 Express with Middleware Libraries

- **Body Parsing**:

```
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
```

- **CORS**:

```
const cors = require('cors');
app.use(cors());
```

- **Helmet (security)**:

```
const helmet = require('helmet');
app.use(helmet());
```

- **Morgan (logging)**:

```
const morgan = require('morgan');
app.use(morgan('dev'));
```

6.9 Advanced Concepts

- **Template Engines**: e.g., Pug, EJS → like Blade in Laravel.
- **Static Files**:

```
app.use(express.static('public'));
```

- **Versioned APIs**:

```
app.use('/api/v1', v1Routes);
```

📌 Interview Q&A

☑ **Q: What is middleware in Express?**
👉 Functions that run during the request/response lifecycle, used for logging, auth, parsing, error handling.

☑ **Q: How is error handling done in Express?**
👉 Using middleware with **4 params** `(err, req, res, next)`.

☑ **Q: Difference between `app.use()` and `app.get()`?**
👉 `app.use()` mounts middleware for all routes, `app.get()` defines a route handler for GET requests.

☑ **Q: How does Express differ from frameworks like Laravel?**
👉 Express is **unopinionated & lightweight** (you add what you need), while Laravel is **full-featured & opinionated** with built-in ORM, migrations, etc.

☑ **Q: How do you organize large Express apps?**
👉 Using **Routers, middleware layers, services, MVC-like structure**.

📌 Key Takeaways

- Express = **lightweight framework** on top of Node's http module.
- Provides **routing, middleware, error handling**.
- Middleware = core concept (similar to Laravel middleware).
- Use `express.Router` for modular routes.
- Use libraries like **CORS, Helmet, Morgan** for production apps.

👉 Next, we'll cover **7. Database Integration** (MySQL/PostgreSQL, MongoDB with Mongoose, ORM/ODM).

Do you want me to explain **SQL (MySQL/Postgres) + NoSQL (MongoDB)** side by side for comparison, or focus only on **MongoDB with Mongoose (most common in Node world)**?