Task Manager API – Express Server (Line-by-Line Explanation)

Generated on 2025-09-03 23:08:36

Full Source Code (numbered)

```
01  import dotenv from "dotenv";
02  dotenv.config();
03
04  import express from "express";
05  import cors from "cors";
06  import morgan from "morgan";
07  import { connectDB } from "./db.js";
08  import tasksRouter from "./routes/tasks.js";
09
10  const app = express();
11
12  app.use(
13    cors({
14      origin: (process.env.CORS_ORIGIN || "http://localhost:3000").split(","),
15    })
16  );
17  app.use(express.json());
18  app.use(morgan("dev"));
19
20  app.get("/", (_, res) => res.json({ ok: true, name: "Task Manager API" }));
21  app.use("/tasks", tasksRouter);
22
23  // 404
24  app.use((req, res) => res.status(404).json({ error: "Not found" }));
25  // Error handler
26  app.use((err, req, res, next) => {
27    console.error(err);
28    res.status(err.status || 500).json({ error: err.message || "Server error" });
29  });
30
31  const PORT = process.env.PORT || 4000;
32
33  connectDB().then(() => {
34    app.listen(PORT, () => console.log(`■ API listening on http://localhost:${PORT}`));
35  });
```

Line-by-Line Explanations

```
01  import dotenv from "dotenv";
```
Imports the `dotenv` package using ES modules. `dotenv` loads environment variables from a `.env` file into `process.env`.

```
02  dotenv.config();
```
Immediately runs `dotenv.config()` so your `.env` values become available to the process before other imports use them.

```
03
```
Blank line for readability; separates environment setup from the rest of the imports.

```
04  import express from "express";
```
Imports the `express` web framework used to create the HTTP server and define routes/middleware.

```
05  import cors from "cors";
```
Imports the `cors` middleware to control which origins (websites) can access this API from the browser.

```
06  import morgan from "morgan";
```
Imports `morgan`, a request logger that prints concise request information during development.

```
07  import { connectDB } from "./db.js";
```
Imports `connectDB` (your own module) which connects to MongoDB. It returns a Promise that resolves when the DB is ready.

```
08  import tasksRouter from "./routes/tasks.js";
```
Imports the tasks router which contains all `/tasks` CRUD routes (separated by concern).

```
09
```
Blank line for readability; separates imports from application setup.

```
10  const app = express();
```
Creates an Express application instance; `app` is the central object for registering middleware and routes.

```
11
```
Blank line for readability.

```
12  app.use(
```
Registers the CORS middleware with custom options using `app.use(...)`. Middleware runs for every request in order.

```
13    cors({
```
Calls `cors({...})` to configure CORS:

```
14      origin: (process.env.CORS_ORIGIN || "http://localhost:3000").split(","),
```
Sets `origin` to an **array** of allowed origins. Reads `CORS_ORIGIN` from env (comma-separated origins) or defaults to `http://localhost:3000`; `.split(',')` turns it into an array.

```
15    })
```
Closes the CORS options object.

```
16  );
```
Applies the configured CORS middleware to the app.

```
17  app.use(express.json());
```
Registers the JSON body parser so `req.body` contains parsed JSON for `Content-Type: application/json` requests.

```
18  app.use(morgan("dev"));
```
Registers `morgan('dev')` to log each incoming request (method, URL, status, response time) to the console.

```
19
```
Blank line for readability.

```
20  app.get("/", (_, res) => res.json({ ok: true, name: "Task Manager API" }));
```
Defines a health/root endpoint `GET /`. It ignores the first parameter (request) using `_` and responds with JSON `{ ok: true, name: 'Task Manager API' }`.

```
21  app.use("/tasks", tasksRouter);
```
Mounts the `tasksRouter` on the `/tasks` path so all routes inside it become `/tasks/*`.

```
22
```
Blank line for readability.

```
23  // 404
```
Comment: marks the start of the 404 handler section.

```
24  app.use((req, res) => res.status(404).json({ error: "Not found" }));
```
Registers a catch-all 404 handler. If no previous route matched, this middleware runs and returns `{ error: 'Not found' }` with status 404.

```
25  // Error handler
```
Comment: marks the start of the centralized error handler.

```
26  app.use((err, req, res, next) => {
```
Registers an error-handling middleware. **Important:** it has 4 parameters `(err, req, res, next)`, which is how Express recognizes it as an error handler.

```
27    console.error(err);
```
Logs the error to the server console to aid debugging.

```
28    res.status(err.status || 500).json({ error: err.message || "Server error" });
```
Sends a JSON error response. Uses `err.status` if provided, otherwise 500. The body contains `err.message` or a generic 'Server error'.

```
29  });
```
Closes the error-handling middleware function.

```
30
```
Blank line for readability.

```
31  const PORT = process.env.PORT || 4000;
```
Reads the listening port from `process.env.PORT`; defaults to `4000` if not set.

```
32
```
Blank line for readability.

```
33  connectDB().then(() => {
```
Initiates the database connection. Only after a successful connection (Promise resolved) do we start the HTTP server.

```
34    app.listen(PORT, () => console.log(`■ API listening on http://localhost:${PORT}`));
```
Starts the Express server on `PORT`. The callback logs a friendly message with the URL to the console.

```
35  });
```
Closes the `then` callback for `connectDB()`.