NestExpressApplication

August 18, 2026

NestExpressApplication

Kind: Interface

Source: `packages/platform-express/interfaces/nest-express-application.interface.ts`

Part of: Platform Express

Interface describing methods on NestExpressApplication.

`NestExpressApplication` extends Nest's standard application contract with Express-specific configuration APIs. Use it when an application needs Express features such as template engines, static asset serving, custom body parsers, and Express locals.

Diagram

```
graph LR
  A[NestFactory.create] --> B[NestExpressApplication]
  B --> C[Configure views]
  B --> D[Serve static assets]
  B --> E[Configure body parsers]
  B --> F[Set Express locals]
  B --> G[listen()]
  G --> H[Express HTTP server]
```

Usage

```
import { join } from 'node:path';
import { NestFactory } from '@nestjs/core';
import { NestExpressApplication } from '@nestjs/platform-express';
import { AppModule } from './app.module';

async function bootstrap() {
  const app = await NestFactory.create<NestExpressApplication>(AppModule);

  app.setBaseViewsDir(join(__dirname, '..', 'views'));
  app.setViewEngine('hbs');
```

```
  app.setStaticAssets(join(__dirname, '..', 'public'));
  app.setLocal('appName', 'My Nest App');

  app.useBodyParser('json', { limit: '2mb' });

  await app.listen(3000);
}

bootstrap();
```

AI Coding Instructions

- Create the application with `NestFactory.create<NestExpressApplication>()` when Express-specific methods are required.
- Configure view directories and the template engine before starting the server with `listen()`.
- Use `setStaticAssets()` for public files instead of implementing static-file routes manually.
- Prefer `useBodyParser()` when custom parser limits or parser options are needed; ensure the selected parser matches the incoming content type.
- Keep Express-specific configuration in the bootstrap layer so application modules remain transport-agnostic.

Relationships

- IMPORTS → `HttpServer`
- IMPORTS → `INestApplication`