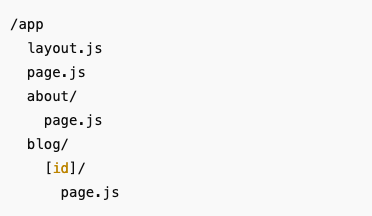
# next.js两种模式page router vs app router

| **特性 / 维度** | **Pages Router（传统）** | **App Router（现代推荐）** |
| --- | --- | --- |
| 📁 路由目录结构 | /pages | /app |
| 🧭 路由方式 | 文件即路由（如：/pages/about.js → /about） | 文件夹+page.tsx 组合（如：/app/about/page.tsx → /about） |
| 🎬 页面组件写法 | 默认导出 React 组件 | 必须导出 Page 组件，支持嵌套路由 |
| 🧱 布局（Layout）系统 | 仅支持全局 \_app.js、\_document.js | **支持嵌套布局** layout.tsx**，每级路由独立布局** |
| ⌛ 中间件（Middleware）支持 | 支持（较弱，仅配合 Pages Router 的 URL 结构） | **原生支持、自动识别** /middleware.ts**，功能更强更灵活** |
| 🔁 动态路由 | [id].js, [slug].js | [id]/page.tsx, [slug]/page.tsx |
| 🔌 数据获取（静态） | getStaticProps | generateStaticParams + async 组件 |
| 🔄 数据获取（服务端） | getServerSideProps | async 页面组件直接处理 |
| 🔄 数据获取（初始 props） | getInitialProps（已不推荐） | ❌ 不支持（被替代） |
| 🌐 SEO 支持 | 手动 <Head> | generateMetadata() 方法自动生成 |
| 🧩 模块复用性 | 只能在 \_app.js 控制全局 | **模块化强，布局嵌套灵活可复用** |
| 📦 API 路由支持 | /pages/api/\* | /app/api/\*（结构更清晰） |
| ⚡ 性能优化（懒加载） | 手动优化 | 默认开启模块懒加载 |
| 💾 缓存控制 | 搭配 HTTP 响应头控制 | 原生支持 fetch() 缓存策略（如 force-cache） |
| 🧰 Middleware 支持强度 | ✅（基本支持） | ✅✅✅ **原生集成，权限控制/AB测试首选** |
| 📅 首次引入版本 | Next.js 初始版本（长期支持） | **自 Next.js 13 开始引入 App Router（**app/ **目录）** |
| 🧠 元信息定义 | <Head> 内部手动编写 | generateMetadata() 函数式定义 |
| 📄 页面预加载支持 | ✅ 支持 | ✅ 支持 |
| 🧩 页面模块粒度 | 页面为单位 | **布局 + 页面 + loading + error 为粒度单位，组件化更彻底** |
| 🧪 使用复杂度 | 简单，开发者熟悉 | 稍复杂，但更强大、灵活 |
| 🧩 推荐场景 | 老项目维护、轻量网站 | **新项目开发、复杂布局、动态权限、国际化** |

## Pages Router 示例结构



## App Router 示例结构



# 推荐选择指南

| **项目类型/需求** | **推荐模式** | **理由** |
| --- | --- | --- |
| 快速搭建小项目、老项目迁移 | Pages Router | 简单稳定，学习曲线低 |
| 新项目、追求性能、组件复用 | App Router | 支持 RSC、嵌套布局、功能更强 |
| 多语言、多主题、多 Layout | App Router | 更易组织，原生支持 layout |
| SEO、SSR、自定义 Head 支持 | 二者皆可 | 两种模式都支持，但 App Router 方式更灵活 |