

Development Setup Guide - Payload CMS Integration

This guide explains how to use the development-only Payload CMS integration in your Next.js project.

Overview

Step 3 of the Payload CMS integration has been completed, adding development-only startup logic that allows you to run Payload CMS alongside your Next.js application during development without affecting production builds.

Key Features

- ✓ **Development-Only:** CMS only runs in development mode (`NODE_ENV=development`)
- ✓ **Production Safe:** Zero impact on production builds or deployments
- ✓ **Port Management:** CMS runs on port 3001, Next.js on port 3000
- ✓ **Easy Scripts:** Multiple ways to start development servers
- ✓ **Graceful Shutdown:** Proper process management with SIGINT/SIGTERM handling

Available Commands

1. Run Both Servers Together (Recommended)

```
./scripts/dev-with-cms.sh
```

This starts both Next.js (port 3000) and Payload CMS (port 3001) simultaneously.

2. Run Only Payload CMS

```
./scripts/start-cms-only.sh
```

Useful when you only need to work on CMS content or configuration.

3. Run Only Next.js (Standard)

```
npm run dev
```

Standard Next.js development - no CMS running.

4. Run Payload CMS Directly

```
NODE_ENV=development npx tsx scripts/start-payload-dev.ts
```


Direct execution of the Payload startup script.



URLs When Running

Service	URL	Purpose
Next.js App	http://localhost:3000	Main application
Payload Admin	http://localhost:3001/admin	CMS admin interface
Payload API	http://localhost:3001/api	CMS API endpoints

Environment Safety

The development startup script includes built-in safety checks:

```
#  WORKS - Development mode
NODE_ENV=development npx tsx scripts/start-payload-dev.ts

#  SKIPS - Production mode
NODE_ENV=production npx tsx scripts/start-payload-dev.ts
# Output: " Payload CMS startup skipped (not in development mode)"
```

Production Build Verification

Production builds are completely unaffected:

```
npm run build #  Still works perfectly
npm start    #  Production mode - no CMS
```

Files Created

Core Scripts

- `scripts/start-payload-dev.ts` - Main development startup logic
- `scripts/dev-with-cms.sh` - Helper to run both servers
- `scripts/start-cms-only.sh` - Helper to run only CMS

Configuration Updates

- `tsconfig.json` - Excludes `scripts/cms` folders from production builds
- Dependencies added: `payload`, `@payloadcms/db-sqlite`, `express`, `@types/express`

Development Workflow

First Time Setup

1. Ensure all dependencies are installed: `npm install`
2. Make scripts executable: `chmod +x scripts/*.sh`
3. Start development: `./scripts/dev-with-cms.sh`

Daily Development

1. Start both servers: `./scripts/dev-with-cms.sh`
2. Access your Next.js app: `http://localhost:3000`
3. Access Payload admin: `http://localhost:3001/admin`
4. Work on your project normally
5. Stop with `Ctrl+C`

CMS-Only Development

If you only need to work on CMS configuration or content:

```
./scripts/start-cms-only.sh
```

Troubleshooting

Port Conflicts

If port 3001 is busy, set a different port:

```
PAYLOAD_PORT=3002 ./scripts/start-cms-only.sh
```

Script Permissions

If scripts don't run, make them executable:

```
chmod +x scripts/*.sh
```

TypeScript Errors

Scripts are excluded from production builds but you can check them:

```
npm run tsc --noEmit scripts/start-payload-dev.ts
```

Next Steps

This completes Step 3 of the Payload CMS integration. The next steps will be:


1. **Step 4:** Set Up Payload Admin Interface
2. **Step 5:** Define Content Collections (Blog, Partners, Products)
3. **Step 6:** Create Data Migration Scripts
4. **Step 7:** Integrate Payload with Next.js API Routes
5. **Step 8:** Update Next.js Components

Support

For issues with the development setup:

1. Check that all dependencies are installed
2. Verify Node.js version compatibility

3. Ensure scripts have executable permissions
 4. Check for port conflicts
-

Status:  Step 3 Complete - Development-only startup logic fully implemented and tested.