

PLAYBOOK

Ecommerce Migration with AI Tooling

A WordPress-to-MedusaJS migration pattern from real projects: AI-assisted content inventory, URL and data mapping, and staged cutover with a verified project diary.

Why migrations go wrong

Commerce migrations fail quietly. The new storefront ships, redirects mostly work, and three months later somebody notices a product attribute family never moved. The failure mode is rarely technical - it is undocumented scope. The discipline that prevents it is a written inventory before any code, and a public diary that distinguishes 'merged in source' from 'live in production'.

The four-phase pattern

01 Inventory

Crawl the legacy site and export the CMS: pages, products, categories, media, redirects, plugins in active use. AI tooling is genuinely good here - summarising crawl output and classifying page types turns a week of manual spreadsheet work into a day.

02 Mapping

Every legacy entity gets a target: Medusa product, category, CMS page, redirect, or deliberate drop. The drop list is signed off by the customer - nothing disappears silently.

03 Rebuild

The new theme is owned code, not a builder export. Commerce data moves into MedusaJS entities with import scripts that can be re-run idempotently while content keeps changing on the old site.

04 Cutover

Staged: new storefront on a private development address, verified against a checklist, then DNS switch. The diary records each state separately - source merged, staging verified, production accepted.

Where AI tooling earns its keep

- Content inventory: classifying hundreds of legacy URLs into products, categories, editorial and dead pages
- URL mapping: proposing redirect tables from crawl data - always reviewed, never applied blind
- Data transformation drafts: turning WordPress export shapes into Medusa import shapes for engineer review
- Regression sweeps: diffing rendered output before and after theme rebuilds

Development addresses stay private. Staging URLs are for the team and the customer - public project pages show the work, never the preview hostname.

The honest diary discipline

In-progress project pages work because they say exactly where things stand. Each entry names a verified event - a merged change, a staging delivery, an acceptance note - and the page header states plainly that production migration is ongoing. Customers reading their own project page should recognise their project; strangers should learn what working with us looks like.

- ☐ Written inventory exists before migration code starts
- ☐ Every legacy entity has a mapped target or an approved drop
- ☐ Import scripts are idempotent and re-runnable during the transition
- ☐ Staging verification completed and recorded before DNS cutover
- ☐ Post-launch redirect coverage measured, not assumed

This document is part of the UHL resources collection. Direct download and related work: <https://uhl-services.ch/en/resources/>

