Pixel & Tonic

Craft CMS Site Audit - Optimization

Findings & Recommendations for CretaLive.gr
by Top Shelf Craft / reviewed by Pixel & Tonic, Inc.

Never before in my career has the phrase "*It's all Greek to me*" been so applicable! Thank you for giving me an opportunity to review and audit the CretaLive site. I enjoyed digging into this project, and I believe my findings will be valuable for you. I look forward our continued work together.

[illegible]

Executive Summary

I performed a cursory evaluation of the overall site setup and then proceeded through an in-depth analysis of the homepage templates, according to the priorities we laid out in our initial discussions.

The overall architecture of the site is straightforward and sensible. I've provided a few recommendations for resolving potential issues in the Craft CMS setup and providing a better author experience, but overall you're doing well in this area.

The majority of the audit focused on optimizing the homepage, with special attention towards reducing the query load. The most impactful recommendation here is to make use of eager loading to eliminate 'n+1' queries for Elements relations and Asset Transforms. Doing so, I was able to reduce the query load on the homepage by about 50%, and I have provided my code changes for you to use as a reference. The same 'best-practices' I tested on the homepage will, in most cases, also be applicable throughout the site.

This audit was reviewed by Pixel & Tonic. The results, findings, and recommendations are within the constraints of the Agreement.

Additional Details

When a site makes extensive use of macros or includes, as you have done in your templates, it can be tricky to tell where it makes sense to use optimizations like eager-loading. This is because the parts of your code that access individual attributes of objects are separated from the places where you fetch the objects initially. (For example, you may fetch a set of Entries in one template and feed them into an include template, which feeds them into a macro, and *then* some Assets are accessed from each Entry.) However, by isolating one block of code at a time, we can identify the template elements that generate the most database activity, and work down through the layers from there.

In addition to eager loading, some general refactoring will help eliminate repeated operations, and I have provided suggestions for this wherever I saw obvious examples.

Server Setup Notes

- PHP7 is definitely the right choice. (We typically see 1.5x-2x speed improvements using PHP7 over PHP5.6 with Craft.)

- Consider Nginx w/ PHP-FPM in place of Apache. (Nginx uses an asynchronous, event-driven approach to handling connections, so it can handle many connections in a single worker process. This usually brings much better performance on sites with many concurrent users and frequent traffic spikes.)

Overall Patterns & Misc. Suggestions

- Site templates can use the .twig file extension, which enables Twig syntax support in many IDEs.

- Make sure to add a redirect so that `/index` doesn't serve up a duplicate page as `/` — Duplicate content can cause SEO trouble, especially on a homepage.

- Make use of the `{{ url() }}` helper to generate fully qualified URLs (based on the siteName in each environment) rather than using relative paths.

- For placeholder elements (e.g. Asset 219) — Query these objects once and set it as a global variable once (i.e. at the beginning of a base/master template) and use your master variable throughout subsequent templates, rather than querying the asset by ID in each instance where you use it. While Craft is smart enough to only run the query once, and cache its result for later use, it will be easier to adjust a value in the future if you only define it once.

- Use Craft's eager-loading abilities wherever possible. Doing so is singularly responsible for almost all of the query elimination I was able to find in my analysis. *Any time you are loading multiple elements and later intend to use element-relation attributes of those elements*, eager-loading is your best friend.

- Eager loading *doesn't* usually benefit you in cases where you are only fetching one parent element. This comes up a lot in sections where you are only accessing the `.first` item in a set of Entries, and then accessing the `.first` Asset connected to that Entry. I call these cases "first-on-first queries." Eager loading takes 3-4 queries per instance (to load the parent element, to load the related elements, and to form the relationship map). By contrast, a first-on-first query takes only 2 database hits: one to load the parent element, and one to load the related element. So, in cases where you are loading several parent elements, eager loading is helpful. However, where you are only accessing one related item on one parent entry, use the normal lazy-loading methods.

- Use Craft's native caching, especially for content that doesn't change frequently. I recommend exploring a "Russian-doll style" caching scheme for more granular caching and cache-breaking.

Homepage Layout Analysis

blank-template benchmark	8 queries	0.20s
current site	57 queries	1.13s
w/ optimization	39 queries	0.60s

- **developerSettings** — *6 queries*
  - Unless the Developer Settings values need to be repeatable (which doesn't seem to be the case), consider moving them out of the Matrix block and assigning them directly on the element. Grouping them into a Matrix block adds query overhead.

- **_includes/head** — *0 queries*
  - No issues here. ☺

- **_includes/header** — *6 queries*
  - Consider using fully qualified URLs in your nav, as this often provides SEO benefit.
  - Namedays
    - Comparison of a native DateTime object with a formatted is shaky. Either format both value and criterion dates as strings, or compare them as DateTime objects.
    - Or, even better: Use a query criteria to select only the one desired item, instead of fetching all possible items and comparing them to see whether each one matches. (This will result in a lot of memory overhead as more dates are added to the system.)
    - Consider refactoring into a separate channel rather than a Matrix block. As the number of Namedays in the system grows, editing these as Matrix blocks will be cumbersome both for performance and for your authors/UX.
    - You can cache this block aggressively using Craft's native caching, since it presumably only changes once per day.
  - The contents of the header block may not change very frequently. Consider caching the entire header, even for 30-60 minutes.

- **_includes/sidebar** — *benchmark: 34 queries*
    - Newsroom (uses the helpers.listFlow macro) — *16 queries*
        - You don’t need to format the dates as timestamps in order to compare them — PHP/Twig allows you to compare DateTime objects directly.
        - Eager-loading the `newsCategory` attribute saves 12 queries!
    - “Read More” — *18 queries*
        - Set the default image once, outside the loop, rather than in each loop iteration (saves several queries, depending on how often the fallback is used).
        - Save several queries by eager-loading `mainPhoto` and `newsCategory`

- **_includes/footer** — *10 queries*
    - Don’t eager-load related elements for a first-on-first query (i.e. `homeMidArticles.politesMain`
    - Consider caching this section natively, as it is unlikely to change too frequently.

Index Template Analysis

blank-template benchmark	8 queries	0.2s
current site	232 queries	2.3s
w/ optimization	99 queries	1.6s

- **Main Article Block** — *5 queries*
    - Don’t eager-load first-on-first queries — eliminates 1 query

- **Sub Article Block** — *8 queries*
    - Don’t eager-load first-on-first queries — eliminates 1 query

- **Highlights Block** — *38 queries – YIKES!*
    - Eager load `mainPhoto` with transforms — eliminates 26 queries!
    - Eager load `newsCategory` — eliminates 5 queries

- **Top Stories Block** — *8 queries*
    - No big issues here.

- **I Saw I Hear** — *16 queries*
    - Don’t eager-load first-on-first queries — eliminates 2 queries

- **Main History + Opinion block** — *31 queries*
    - Don’t eager-load first-on-first for `mainPhoto` — eliminates 1 query

    - Load all Opinions at once and slice them in memory, rather than loading the first, and then the rest, in separate operations — eliminates 2 queries
    - If a helper macro is going to retrieve an image transform for an Asset field on every entry, go ahead an eager-load that transform. — Save 3 queries

- **Sections** — *127 queries — big opportunity for improvement here!*

    - \`alternativeSection\` variant — *33 queries each*
        - Pre-load \`mainImage\` with transforms — eliminates 18 queries
        - Pre-load \`newsCategory\` — eliminates 3 queries
        - Pre-load \`newsAuthor\` — eliminates 3 queries

    - \`defaultSection\` variant — *16 queries each*
        - Eager-loading saves 6 queries

Search Implementation Recommendations

- Don't worry if you see empty values in the search index table — The search index table will contain many rows for fields that don't actually have searchable values. Blank rows/columns in the search index isn't usually concerning.

- Mixed Greek and Latin characters in the index values are likely a result of the \`customAsciiCharMappings\` config setting you have provided, i.e. this is *probably* to be expected. Consider tweaking the \`defaultSearchTermOptions\` config item — specifically, set the \`exact\` key to false — to make the results a bit more tolerant to variance in search queries.

- Pixel & Tonic support will be able to help you resolve any issues with erroneous search results.

- Consider implementing elastic search (i.e. via Algolia/SearchPlus) for faster, more relevant results and more flexibility in index faceting.

Analytics Recommendations

- It's generally advisable to perform analytics functions client-side wherever possible. (This allows you to use the most aggressive server-side caching possible.) Therefore, the Entry Count plugin is probably not serving you very well. Focusing your attention towards adding a robust Google Analytics setup is probably best. Dukt Analytics is a good plugin for making analytics data available in the control panel. In addition to tracking per-page visits, you can also use custom events to track how many times a story appears on the homepage, clicks from various traffic sources, etc.

Craft CMS Setup Recommendations

- You currently use Global sets to manage the selection of entries to feature in the various Homepage sections. It's unclear to me whether most of that content is actually *global* in nature — i.e. whether those selections need to be accessible on other pages besides the homepage. Global elements are loaded on *every* page request. Consider making the homepage an Entry in the CMS (perhaps implemented as a Single) and attaching homepage-specific content to that entry instead of using Globals.

- A few places in the CMS seem to be using Matrix blocks to provide visual grouping for related fields. (For example, the global 'Dev Settings' fields are grouped using a single Matrix block.) Matrix fields are designed for *repeating* content. In cases where content doesn't actually need to be *repeatable*, use of Matrix introduces unnecessary overhead and can be simplified.

Additional Recommendations

- **Continuing code clean-up** — This audit focused on the homepage layout and index template, with the stated goal of reducing the query load on the homepage. As you go forward and apply our notes across the entire site, I believe you'll see a significant overall performance improvement. While your existing architecture is good, a more extensive investigation and refactoring of the helper macros, includes, and template structure, may produce additional gains in both performance and maintainability.

- **Improving Control Panel AX for Homepage featured-stories selection** — The author experience of selecting featured entries could be improved if the selection fields for each homepage 'section' only showed entries that would potentially appear in that section — i.e. if the field for selecting featured Lifestyle entries only displayed entries belonging to the Lifestyle category. Additionally, in the example you provided, you showed how it would be useful to display metadata with each selection, such as who made the selection and when. This customization would require some custom development. However, such a custom field type would extend the existing Entries field type almost entirely, with only incremental additions required. I agree with you that customization in this area would be very useful for improving your editors' experience.

- **Custom cache-breaking logic —** Since there are so many atomic pieces of content changing so frequently, developing a more granular cache invalidation/regeneration process would yield significant benefits both to the site's performance and to your sanity as a developer. You identified this as a significant concern in our initial discussions, but unfortunately, I believe you've "maxed out" the currently available toolset. Additional work in this area would require some custom development/consultation.

- **Analytics** — I imagine you can accomplish what you need for entry analytics without needing to do any custom plugin development. If you need assistance customizing your analytics setup to meet specific business/reporting requirements, we're happy to help.

- **Implementing Accelerated Mobile Pages** — *Accelerated Mobile Pages* is a client-side technical specification for ensuring that web pages load extremely quickly, especially on mobile devices. The project is developed by Google, specifically oriented towards publishers, and backed by Twitter, NYT, Vox, and others. Once you're comfortable with the state of the project on the server side, I recommend giving additional attention to client-side page rendering, as improvement in this area has the potential to yield big improvements in user satisfaction, engagement, and SERP.

Contact

We look forward to continuing to support you.

Please contact us any time we may be of service!

Michael Rog
Top Shelf Craft
michael@topshelfcraft.com
+1 (713) 581-4764

Leslie Camacho, Chief Customer Officer
Pixel & Tonic, Inc.
leslie@pixelandtonic.com
+1 (541) 414-6253