effective UI

The case for headless CMS

Steve Clement
Technical Director, EffectiveUI

At EffectiveUI, we are seeing continued and increasing interest in decoupled, or "headless" CMS. The decoupled approach to software is nothing new, and has always been the cornerstone of solid software architecture. And now as Content as a Service (CaaS) vendors are poised to transform how we think about content management systems, it is important to research the pros and cons of headless CMS to determine whether or not this approach is right for your business and your digital product strategy.

Anyone looking to bring innovative digital products to market should be aware of current thinking across their technology stack, and how the CMS plays an important part of many digital strategies. However, to remain competitive, companies need to keep up with current architectural styles such as microservices while acknowledging that the old ways of bringing a product to market may no longer be sufficient.

How can decoupling and microservices use a headless approach to minimize your total cost of ownership (TCO)? Let's focus on a few things to show how you can lower your long-term costs:

- What is headless CMS and how is it priced?
- Microservices and rapid product development
- Technology and talent
- Microservices and LeanUX

Headless CMS

WHAT IS IT?

Very simply, a headless CMS differs from traditional CMS platforms like Sitecore, Drupal and WordPress by managing your content without providing a presentation layer, or "head." Instead, the system exposes your content through APIs, and any application presenting that content can do so in whatever way it chooses. The advantages of this approach are obvious when you consider the variety of different digital channels you may wish to target. Each channel—from Web and mobile to Smart TV and wearables—has its own presentation needs and corresponding presentation technologies. When content is simply another API service providing data, front-end developers can style and create rich experiences around the data just as they would with any other service call.

Very simply, a headless CMS differs from traditional CMS platforms like Sitecore, Drupal and WordPress by managing your content without providing a presentation layer, or "head."

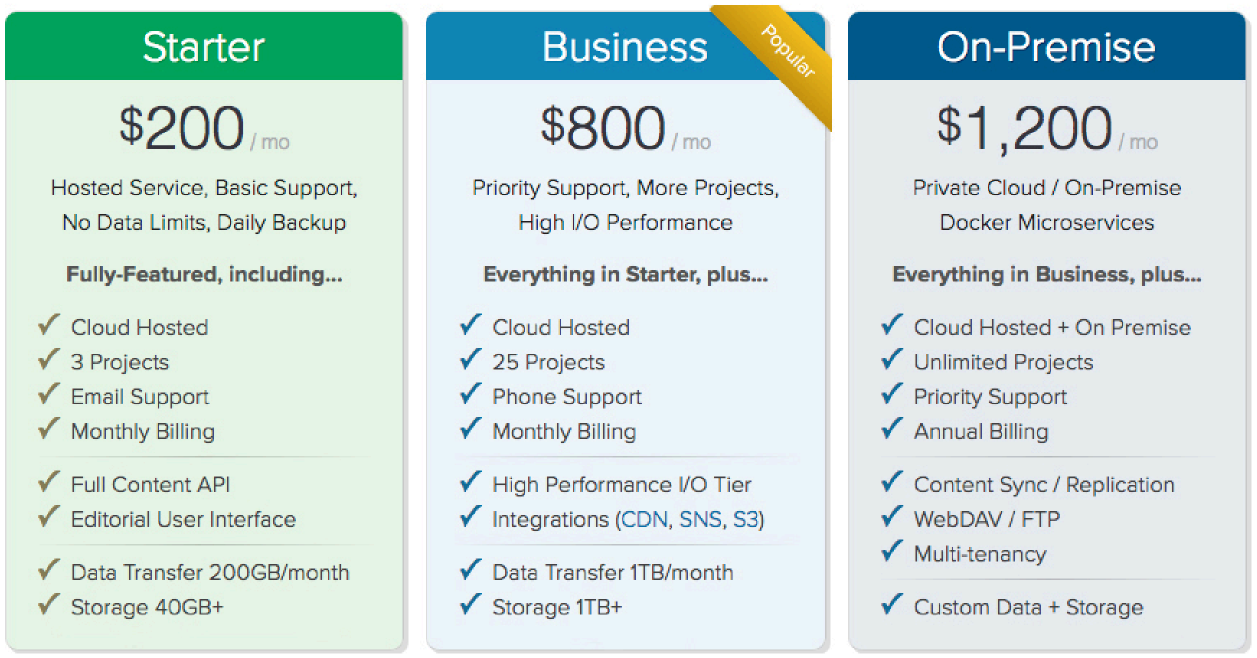
PRICING

The price of traditional CMS platforms increases with functionality and levels of customization. CMS platforms with basic functions are available for a modest price; some like Drupal or WordPress are free (not counting hosting costs). As additional features and the ability to customize increase, the price can go up dramatically. Platforms like Sitecore and Adobe Experience Manager (AEM) offer additional features and customization options, making them top-of-the-line in regards to capabilities and price.

Cloud-based CMS services offer a much lower subscription price, on average, than traditional CMS licenses. While it's true that many don't offer the feature parity of traditional CMS for things like search, scheduled publications, workflow, and built-in analytics, those that do are still much more reasonably priced.

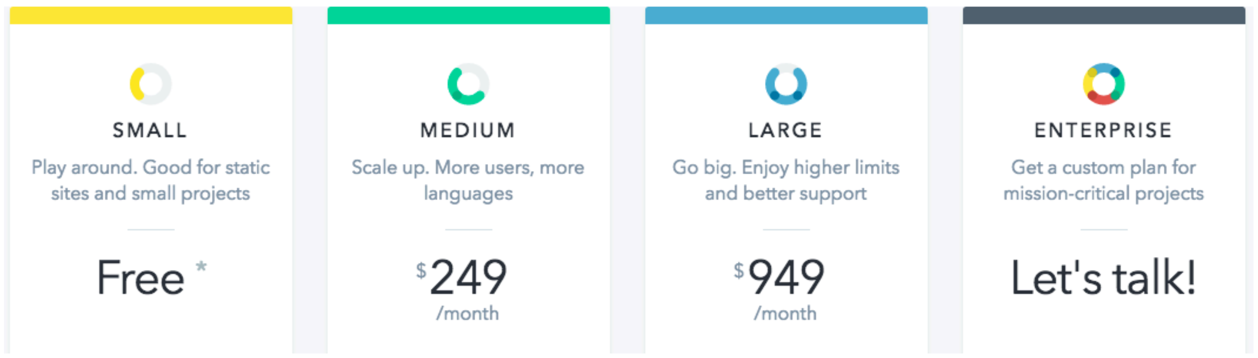
A Sitecore license can run $200K depending on the number of servers needed, not including the $35K annual support and maintenance fee. Kentico, another CMS built on the .NET platform, has a single server license cost between $17K and $40K depending on the features, with an annual renewal fee of $5-12K.

In contrast, the pricing plans of some of the headless options such as Cloud CMS and Contentful as of August 2016 are much less cost prohibitive:



Starter	Business	On-Premise
$200 / mo	$800 / mo	$1,200 / mo
Hosted Service, Basic Support, No Data Limits, Daily Backup	Priority Support, More Projects, High I/O Performance	Private Cloud / On-Premise Docker Microservices
Fully-Featured, including...	**Everything in Starter, plus...**	**Everything in Business, plus...**
✓ Cloud Hosted	✓ Cloud Hosted	✓ Cloud Hosted + On Premise
✓ 3 Projects	✓ 25 Projects	✓ Unlimited Projects
✓ Email Support	✓ Phone Support	✓ Priority Support
✓ Monthly Billing	✓ Monthly Billing	✓ Annual Billing
✓ Full Content API	✓ High Performance I/O Tier	✓ Content Sync / Replication
✓ Editorial User Interface	✓ Integrations (CDN, SNS, S3)	✓ WebDAV / FTP
		✓ Multi-tenancy
✓ Data Transfer 200GB/month	✓ Data Transfer 1TB/month	✓ Custom Data + Storage
✓ Storage 40GB+	✓ Storage 1TB+	

Cloud CMS



SMALL	MEDIUM	LARGE	ENTERPRISE
Play around. Good for static sites and small projects	Scale up. More users, more languages	Go big. Enjoy higher limits and better support	Get a custom plan for mission-critical projects
Free *	$249 /month	$949 /month	Let's talk!

Contentful

However, when evaluating TCO, it's a mistake to equate *price* with *cost*. To truly understand how your long-term costs come down, let's look at how a headless approach can help power microservice architecture and a rapid to-market digital strategy.

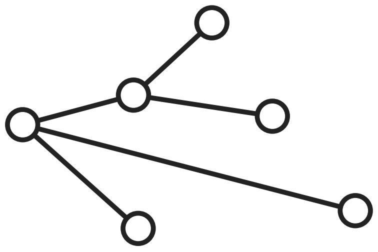
When evaluating TCO, it's a mistake to equate *price* with *cost*.

Microservices and rapid product development

We have seen that enterprises are struggling to adjust to the rapid to-market capabilities of disruptive startups like Uber. One possible solution is adopting what Gartner calls a Bimodal approach. Also known as two-speed IT, the Bimodal approach acknowledges that while our existing IT processes (Mode 1) may be slow, they cannot simply be discarded. They are, in fact, the engine of the current business model, albeit risk-averse and slow to change. A separate parallel effort, Mode 2, can focus on Agile and Lean principles that allow a digital product to be designed, developed and delivered in a fraction of the time, making it nimble and reactive to market forces.

Mode 1

Mode 2



Microservices have become a hot topic recently, as they often provide the core architectural pattern behind Mode 2 initiatives. According to Martin Fowler, microservices "are built around business capabilities and independently deployable by fully automated deployment machinery." If one of these services needs to change, evolve, or pivot based on market forces, the change is isolated to that service.

One of the problems facing traditional CMS platforms is that their monolithic nature doesn't facilitate the kind of rapid, iterative, constantly refined scope that is the hallmark of Mode 2. Once the presentation layer has been decoupled, however, rapid iterations for new products and experiences are achievable. Using a headless CMS exposes your content as data, allowing designers and developers to create digital products without the hassle of battling legacy designs and presentation decisions made for another product or context.

In addition, microservices are generally easier to set up for continuous delivery on platforms like Jenkins or Travis than on monolithic architectures. Bimodal IT benefits from the automated testing and delivery that comes as a part of the DevOps collaboration, and a headless CMS allows other parts of the stack to automate the builds and testing of the "head" (e.g., native mobile, or JavaScript web application) in a way that is just not available to monolithic CMS platforms. If you have business logic in the presentation layer, testing and automated delivery becomes significantly more difficult due to the tight coupling of content and presentation.

Technology and talent

Another thing to consider when looking at CMS platforms is the actual stack on which the CMS is built. Traditional CMS platforms are applications built on top of other technology: Sitecore and Kentico are built on .NET, Adobe Experience Manager (AEM) on Java, Drupal and WordPress on PHP, and so on. As the technologies that provide the underpinning for traditional CMS platforms, these languages and frameworks aggregate functionality and thereby create dependencies.

Headless CMS systems provide a simple alternative: decouple the presentation. This has two key benefits. First, making changes and upgrades to the underlying CMS is easier due to the minimized dependencies. Second, the front-end code can keep pace with the rapidly changing world of front-end frameworks because it is not tied to a monolithic CMS system.

An example: the .NET framework has an established method of processing web requests. The current method is ASP.NET MVC, replacing the old method, WebForms. There is a tipping point when a technology becomes the default for adoption, and MVC reached that tipping point some time ago. As early as 2013, Microsoft was providing an upgrade path from WebForms to MVC.

If your CMS had originally been built on .NET WebForms, changing to ASP.NET MVC would likely not be a simple upgrade. All parts of your CMS that depend on and interact with WebForms would need to be rewritten. Furthermore, there will always be other components of the technical stack with their own product lifecycles and dependencies within your CMS. Because of this, these types of upgrades for the CMS application are always going to happen at an understandably slower pace than any one part of the stack might be if it were decoupled from the other parts.

Headless CMS systems provide a simple alternative: decouple the presentation. This has two key benefits. First, making changes and upgrades to the underlying CMS is easier due to the minimized dependencies. Second, the front-end code can keep pace with the rapidly changing world of front-end frameworks because it is not tied to a monolithic CMS system.

How does this affect TCO? Consider the engineering and developer talent required to manage a CMS site. Because a traditional CMS dictates web presentation creation, being adept at front-end technologies simply isn't enough. Developers need to have a complex understanding of what's going on server-side. Developers acquire specialized knowledge of the platform over time, and that knowledge can be expensive to replace. Hiring managers cannot simply look for good front-end skills (HTML/CSS/JS, IOS/Android, etc.), they also need to look for skills on the particular platform.

When you need to find specialized knowledge, your costs go up. There may not be an ideal candidate who is local, or you may find that additional training is required to get someone up to speed on your platform. Even worse, you may need to settle for average talent, which may lead to lackluster solutions riddled with problems, difficult to maintain and scale.

The headless approach, on the other hand, lets developers work in the technologies they know best. Specialized knowledge of the platform isn't required, and content can be consumed by the application just like any other data. This means your potential hiring pool contains the best talent available for the presentation technology in question (Angular, React, etc.) without needing to filter on knowledge of a specific platform. A larger talent pool increases the odds that you will be able to map the specific design needs to a developer's capability.

Microservices and Lean: time is money

LeanUX is an approach to building products that helps solve business problems. Instead of placing sole focus on the design deliverables, LeanUX dictates that design should only go as far as is absolutely necessary to get feedback from users and/or stakeholders. In lieu of spending valuable time creating high-resolution visual designs, wireframes or even paper sketches lay the groundwork for developers to get to work immediately.

When applied correctly, the Lean philosophy results in prototypes that are rapidly designed, built, and tested so that when the time comes to build the final product, you have data-driven proof that your product will be embraced by the market. It also means getting working software into the hands of end users as soon as possible.

Microservice architecture helps facilitate this Lean approach by providing a backend that can rapidly respond to change as the product begins to take shape. Rapid backend changes can often be somewhat problematic, but they become untenable when working on a monolithic application with code that is intermingled with other business applications. And forget about rapid deployment—the ripple effect of a bad deployment on a monolith affects not only your new product, but other unrelated applications. Your ops team may have something to say about that and may be less than enthusiastic about accommodating your desire for rapid iterations!

Headless CMSs can be the foundation of a microservices architectural style. While not adhering the extreme separation required of strict microservice purists, content is often cross-cutting across multiple digital channels. Your product description, SKU, and price, for example, can be consumed in a mobile app as well as a web experience.

When applied correctly, the Lean philosophy results in prototypes that are rapidly designed, built, and tested so that when the time comes to build the final product, you have data-driven proof that your product will be embraced by the market.

Unfortunately, a traditional CMS doesn't allow for this rapid change paradigm all that well. The tight coupling of presentation with your content means there is much more work to do when trying to rapidly evolve a product. The content itself is probably relatively stable, but your user feedback may require wildly different presentations that can be cumbersome to implement. Of course, the presentation will need to adjust *somewhere* in the delivery pipeline, but the speed at which a good React or Angular developer can make changes will almost always beat out doing it within the templating system of a proprietary CMS.

Concluding thoughts

Granted, it may require a little extra upfront investment to build the presentation layer separate from the content. But, as mentioned earlier, *price* is not the same as *cost*. The TCO benefits of a decoupled system are compelling. Not only can you get to more digital channels in less time, but you also significantly reduce the overall maintenance realities. Your digital product will be able to consume the data from a headless CMS in a way that allows your product team to build the right product, targeted to the right market, and making use of the best development tools and talent. As technologies mature, you are in a better position to adapt. You can stay current with your market needs, and even show some leadership by bringing digital products to market faster than your competitors. ■

About the author

Steve Clement
Technical Director, EffectiveUI

Steve Clement brings 15 years of full-stack enterprise development and architectural experience to his role as Technical Director at EffectiveUI, and is responsible for providing the strategic technical vision for our engagements. He also works closely with his team to make sure they have a deep understanding of the full architectural complexity of our clients' environments and assists with the implementation strategies. He is passionate about staying current with the ever-changing technical landscape in order to be able to provide the best solutions for our clients.

SERVICES

Insight

Strategy

Experience Design

Development

OFFICES

Denver, Rochester, New York City

Our committed, purpose-driven teams empower clients to achieve long-term success.

150+ Industry awards

CONTACT

Scott Esmond
scott.esmond@effectiveui.com

Steve Clement
steve.clement@effectiveui.com

effectiveui.com / 888.310.5327

effective UI

We have a measurable, human-centered design approach to building better experiences. Our teams collaborate and share expertise with you to empower you to achieve long-term success.