The Definitive Guide to Content-as-a-Service

INGENIUX

Content beyond boundaries

Contents

Digital content has grown well beyond the bounds of traditional websites. From new front end development frameworks like Angular JS, Ember, and React, to mobile applications, Internet of Things, print, and other channels, you need to empower business users to manage content beyond traditional websites.

The challenge is that most Web CMS software is not designed to support Web and Mobile applications. A traditional CMS assumes that all of the pages or screens are built and managed in the CMS. *This assumption is a non-starter for most organizations.* Applications are built with custom code, require build and deploy processes, and need to be managed in existing source code control and development operations systems.

It's an evolution from content that was presentation and interaction-oriented to smarter content creation regardless of channel, device, context, etc. It's about proper governance and orchestration of content, and decoupling presentation from management.

What is Content-as-a-Service

Content-as-a-Service, or simply CaaS, is a strategy for delivering CMS managed content to Web applications and other channels.

With CaaS you can manage and edit your content in a CMS. When the content is ready, it can be pushed as a resource file into an external application, or the application can request content using an API-based web service.

Adding content management to your applications empowers your business and marketing teams to update in-app content easily, translate content into multiple languages, personalize content based on business rules or audience, or enable self-service for third-party customers of your applications.

Content-as-a-Service delivers clear business value and is a key strategy to align marketing and IT programs. As such,

Content-as-a-Service delivers clear business value and is a key strategy to align marketing and IT programs.

CaaS is a very popular approach today. However, there is a lot of confusion around content-as-a-service. From “headless” CMS solutions to decoupled or loosely coupled CMS solutions, there are a wide range of approaches to support CaaS, with benefits and liabilities for each.

CMS Applications that Support Content-as-a-Service

The content management systems market is very diverse, with a wide range of solutions from large web experience suites to basic blogging applications. Clearly, not all CMS software is the same, and most CMS software is not designed to support content-as-a-service.

The primary types of CMS applications designed for content-as-a-service are the headless CMS and the decoupled or loosely coupled CMS. We will look at the definition of those terms, but the main point is understanding how a CMS application supports content delivery.

Headless CMS

A headless CMS has the singular focus of enabling the creation and basic management of content and the delivery of that content to external publishing channels typically via a RESTful API. The "headless CMS" only provides the backend content management capabilities; any formatting of the presentation of that content takes place on the front-end delivery tier and is not tied to the CMS in any way.

It's a simple model that works for publishers, designers and developers of highly customized websites or web applications, mobile applications or for highly customized website layouts that a traditional or "tightly coupled" CMS can't support. Headless is also an option for developers who use new JavaScript MVC frameworks such as AngularJS and React.

Decoupled CMS

A decoupled CMS provides the content management capabilities of a full CMS application, but with the flexibility of headless CMS. The term decoupled simply means that the content management application is physically separated from the content delivery environment. A decoupled CMS can either deliver content through an API or by replicating files to a publishing target or remote location.

A decoupled CMS is a technical architecture approach employed by some full CMS platforms. These applications provide a much deeper set of capabilities compared to a basic headless CMS and typically support a file-based delivery model as well as a content API.

Loosely Coupled CMS

Another approach is a loosely coupled CMS application. Loosely coupled means that the CMS software is separate from content delivery, but provides the option of a content delivery framework or application for rendering dynamic content and other services. In some cases, a loosely coupled content delivery tier can be used as middleware for supporting content-as-a-service, providing less dependency on the CMS software and better scaling characteristics, as well as supporting the option of on premise management.

CMS Management, Delivery Freedom

With headless, decoupled or loosely coupled CMS applications, the web presentation layer can easily support web applications and CMS content without complex integration. You can manage any and all content for your website and business systems in your CMS and deliver it on demand to whatever front-end interface requests it, whether it's to a website built on top of the CMS platform or a completely separate web application.

> Not all CMS software is the same, so it's important to understand if and how your CMS supports content delivery

With a CaaS delivery model, you do not have to make changes to legacy applications or data, specific server technology is not needed, and your web experience platform does not dictate what your "stack" looks like. Even better, your customer-facing web applications can consume managed content to deliver a consistent customer experience and content governance process.

Common Use Cases for CaaS

If you are any company doing real business on the web, chances are it's complex and cannot be delivered using a CMS platform alone. When you think about it, there are unlimited use cases where the delivery of content as a service works very well. Let's examine some these.

Native Mobile Applications

Native mobile apps are a prime example of CaaS in use. Native, or hybrid, mobile applications are typically a combination of functionality and content. It could be an interactive application, a game, a store shopping app or something else.

The content in these mobile applications often needs to change frequently (new deals, new information, related content or product). Leveraging CaaS enables developers to update mobile app content continually without having to rebuild or recompile their applications and go through the process of resubmitting them to the App Store and forcing updates on mobile devices.

Multi-Site/Multilingual Websites

Most large enterprises or global organizations manage more than one website or a single website with multiple languages. CaaS is a good approach to provide translations to multilingual websites or to deliver the same content (in different filters or views) to different websites.

Web Applications

Web applications also benefit from CaaS in a few different ways. Web applications are designed to provide business services – banking and financial applications are two great examples. The focus of these applications is the functionality they provide. But they also offer content that needs to be regularly updated.

Here are a few examples of web applications that would benefit from CaaS.

Embedded Help

Most applications have help – tips embedded within the web pages directly and separate help pages, including FAQs, recommended content and so on. This content can be easily managed within a CMS and then fed to the application. If the help is personalized or contextual, it's easier to send a request to the CMS and have the content quickly curated and sent back. If the help is not contextual, maintaining it separately in a CMS enables you to update it regularly and provide it in different languages without having to rebuild the application for each content change.

Labeling

The ability to manage the text of labels in a web application is useful, especially when your application is available in different languages. You can manage the text of the labels within a CMS and then feed them into the web application based on the language selected.

CTAs in a Shopping Cart

Some shopping carts offer CTAs (calls to action). The CTA may be to look at additional products, read reviews on items in the shopping cart, or to provide links to supporting content that encourages the shopper to complete the transaction. Manage the content used in these CTAs in the CMS and then pull it into the shopping cart based on the items the cart contains.

Blogs and Other Supporting Content

Many of today's applications provide a range of supporting content in the form of blogs, downloadable resources and so on. If the web application is completely custom built, this content can be managed within a CMS and then pulled into the application on demand. With major content assets, such as a blog, it's much simpler to create a separate publishing target in the CMS, but with small content, such as marketing text on a page, or widgets that offer recommended content or tips, CaaS is a useful approach. Organizations can keep the content updated and fresh, and personalized to the user and their context.

Working with Angular JS and other JavaScript Frameworks

The Content-as-a-service model works well for developers building apps using modern client-side JavaScript frameworks such as AngularJS, Ember, and React. Developers can create highly responsive web experiences that leverage the content management capabilities of the CMS to store and manage content.

For example, with an AngularJS app directives (parts of the web page) are mapped directly to components managed within the CMS back-end. The CMS provides a visual representation of the app's content for non-coders so they can easily and quickly update content in the web app, such as messages, labels or help text.

With CaaS, you can ensure a consistent customer experience regardless of delivery channel.

Highly Customized Web Layouts

It's often not easy to implement custom layouts using a CMS templating framework without a great deal of development effort. When the front-end visual design is highly customized, it may make more sense to create the website structure outside the CMS and leverage CaaS to serve its content.

Third Party Content Editing

There are occasions when you have an application that contains content that requires third-party content editors or contributors. You want to provide an easy, intuitive way for them to contribute the required content.

Enabling third-party content editing in the CMS allows you to follow a strict review and approval workflow process before you publish the contributed content to the application or website.

What are the Options for Content-as-a-Service

Content as a Service is not exactly new; some organizations have been implementing versions of it for a few years. Here's a look at three ways you can deliver content to an application.

Publishing Resource Files

Content is stored in resource files written in XML, JSON or another proprietary format. These resources files are included at the application's run-time, and there is often a visual tool for changing the content. Publishing resource files is a typical approach to managing help content or labels, or any content that you don't need to change frequently. It is a limited approach however in that it requires the code (resource files) to be part of the application deployment. When updated, the server must be recycled to recognize the changes in the resource files.

Resource file publishing should only be an option when there is no opportunity to change the existing application at all.

File-based Content Delivery

When content is stored in files using XML, the application is updated to load the files like a supporting resource. When loaded in this manner, you can update the files and have the changes reflected immediately without having to recycle the server.

File-based delivery is a "push" approach to delivering content and is often the preferred method by many organizations when CaaS is used internally. This approach is often much better than API deployment (the third approach) because it has zero backend dependency and better performance and scalability. You achieve content personalization by structuring the XML content with metadata that supports the personalization.

When delivering content via a file-based method, you have to pay particular attention to timing and caching, as well as setting up the proper connections and enabling the delivery of files across the firewall.

RESTful API

With a RESTful API, you can deliver real-time content updates to an application or native mobile app. The content API is the design approach for modern web and mobile applications. The application requests the content through the Content API which in turn pulls the right content from the CMS where it is managed and stored.

As with file-based content delivery, the API approach enables content editors and contributors to create and manage content following proper review and approval workflows and ensures that only those allowed to modify the content can do so through the CMS. It also supports the ability to deliver personalized, contextual content based on the request coming from the application.

The Content API does have disadvantages. If there are a lot of API calls you may need to increase the amount of servers supporting the requests. Also, if a server crashes, the API doesn't work and as a result, the application doesn't work. The API method is popular with some CMS vendors because it's easier to scale from a business perspective (just add more servers), but it's less flexible than the file-based method in terms of deciding when to push out new content. If you work with JSON, the content API is often a popular approach to CaaS.

> File-based content delivery works well for internal web applications, while a content API works best when you need to deliver personalized, contextual content.

Key CMS Features Required to Support Content-as-a-Service

Several key components need to be in place in a content management platform to deliver Content as a Service.

Structured Content is Critical

No CMS CaaS model will work correctly without a structured content model. A structured content model, also known as an intelligent content model, is a way to create and manage content completely separate from how it is presented in any application or website.

You store structured content in a format that both defines it using content types and relationships and describes it using metadata. This semantic definition enables the CMS to adapt the content for multiple outputs and formats.

This approach requires a completely new way for many organizations to develop content. Typically they think about content based on the channel. So, for example, when you design a new website, there is a content identification and development phase; the same approach is followed for a web application. In both instances, the content is only defined for its use within a single channel.

It's important to bring the modeling of content a step up, outside of its delivery to any particular channel. Take some time to understand all the content your organization creates and manages. Define your organization's taxonomy including content types, their relationships, and associated metadata. Defining and managing content in this manner ensures that it can be reused across all your channels, both offline and digital. It also supports federated and faceted search.

Your CMS must support the ability to create and deliver structured content. Structured content includes XML, JSON and other formats that provide a rich content definition.

Creating the Decoupled Architecture

In a decoupled architecture the delivery tier is completely separate from the backend content management system. The CMS doesn't dictate the stack or set of technologies required to deliver the content or the website/web application. The delivery tier can be located on completely separate servers and in completely different environments, ensuring that your content is not accessible before being published.

With a decoupled architecture, you do not have the overhead of the CMS application on every web server. Because content is not delivered from a database in the "run-time" environment, it is much more scalable, enabling you to deliver content anywhere – a website on another service, a cloud-based application, a kiosk in a store, or a content delivery network (CDN).

When content is delivered as a web service, typically using a RESTful API and JSON or XML for file-based, it can also be written in a completely different technology from the backend CMS. Also, the requesting application does not have access to the content in the CMS, reducing the number of people with direct access to the content and the risk of denial of service attacks and SQL injections.

Agile Deployment Model

An agile CMS solution offers multiple deployment options to fit how the organization needs to deliver its content now or in the future. These include: dynamic delivery using a server technology like ASP.NET or Java; multi-format delivery using mixed or different server technologies; Web services delivery using a REST or SOAP-based API; device-targeted delivery using a mobile detection system; push-based delivery such as XML, JSON or into an external database so it can be consumed by a remote application; and plain old HTML delivery for static Web content. All of these deployment models can leverage CaaS to request the content from the CMS.

Managing Content Permissions

In a pull based API model, the requesting delivery tier sends a request to the CMS and includes the proper credentials to show they can access content. In a push based model, only someone with the correct permissions on the CMS backend can push new or updated content to the delivery tier.

A CMS might manage content for a number of applications and websites, so

it's important to ensure permissions are properly applied for each application. The CaaS management tier must request the credentials of the calling application and apply permissions appropriately.

Also, by providing a read-only content API, no one can send requests to the CMS to modify the content.

Integrating DevOps Processes and Builds

As you develop your CaaS CMS model, you may slowly update how the content API or the file-based delivery works, and like any other development process, you want to include DevOps processes and builds.

GitHub and Mercurial are two approaches to version control that can help you manage the development and continued updates of your CaaS delivery models. Both offer versioning that enables you to point your delivery tiers to different versions of your content repository.

Also, when you are building your website, native mobile app or web application, and you manage development using one of these version control repositories you can point different versions of your code to pull different versions of the content API or resource files. In this way, you can point applications in development at a development version of the content API or files, and the production versions of your apps at the production version of the content API or files.

Using version control tools you can build a branch for each stage of your content API – development, staging, master – in the same way you create versions of a website or web application.

Remote Preview

One drawback of the content as a service model is not being able to preview what your content looks like in the delivery tier. Because the content is created separately in a CMS, there's often no way to see how it will look when it is applied to a presentation.

Remote or external preview empowers marketing users to make in-context edits and view layouts on pages that are not entirely managed by the CMS. An agile CMS should provide a web services-based preview system that can emulate applications and content in the design-time CMS environment.

Conclusion

The only constant on the Internet is change. When evaluating a CMS you do not only need to plan for the requirements you have today, but also for what's to come. While none of us have a crystal ball, it's pretty clear that the changes coming to the Internet deal with devices, applications, cloud networks, new interfaces like voice and touch, and new channels like in-car telematics or the Internet-of-things. A CMS that supports CaaS will provide the agility to meet the challenges of the changing Internet.

A CMS that supports CaaS provides the agility to meet the challenges of the changing Internet.

The traditional website is still an important channel in your web strategy, but it's no longer the only channel. Most organizations deal with many delivery channels, from websites to key web applications to mobile and enterprise portals. Often you use the same content in more than one of these channels, so it's critical to ensure that content updates are applied consistently across channels. The ability to support content-as-a-service should be part of any modern Web CMS evaluation, whether CaaS is the primary deployment strategy or a capability you will need to support future projects.

Content-as-a-service will enable your organization to move beyond traditional Web Content Management and embrace the third generation of digital content management, delivering engaging customer experiences across all channels and devices whether managed in your CMS, or across external applications and networks.

When applied correctly, CaaS can connect and support all of your sites and applications with content that is updated, governed, and relevant, providing richer customer experiences while dramatically lowering the time and cost required to support digital programs.

About Ingeniux

Ingeniux is the leading provider of web content management and digital experience software. We enable organizations to orchestrate the entire customer experience from acquisition through to sales to support and service, across any device, application, or website.

We build content management software with an unparalleled focus on the content itself. The Ingeniux CMS is designed to manage and deliver modern websites, customer support portals, online communities, and other customer touchpoints.

We believe in intelligent “structured” content. We design our software to enable content reuse, enable true mobile and multi-channel content delivery, and insightful content discovery. Our unique content-as-a-service capabilities deliver content into web and mobile applications, and other key channels.

Ingeniux software is available as a fully managed software service or an on premise application. Ingeniux delivers unparalleled service and support to customers worldwide.

To learn more, visit us at http://www.ingeniux.com.

INGENIUX

1601 2nd Avenue, Suite 1010
Seattle, WA 98101

info@ingeniux.com
877 445 8228