2019

dotCMS White Paper

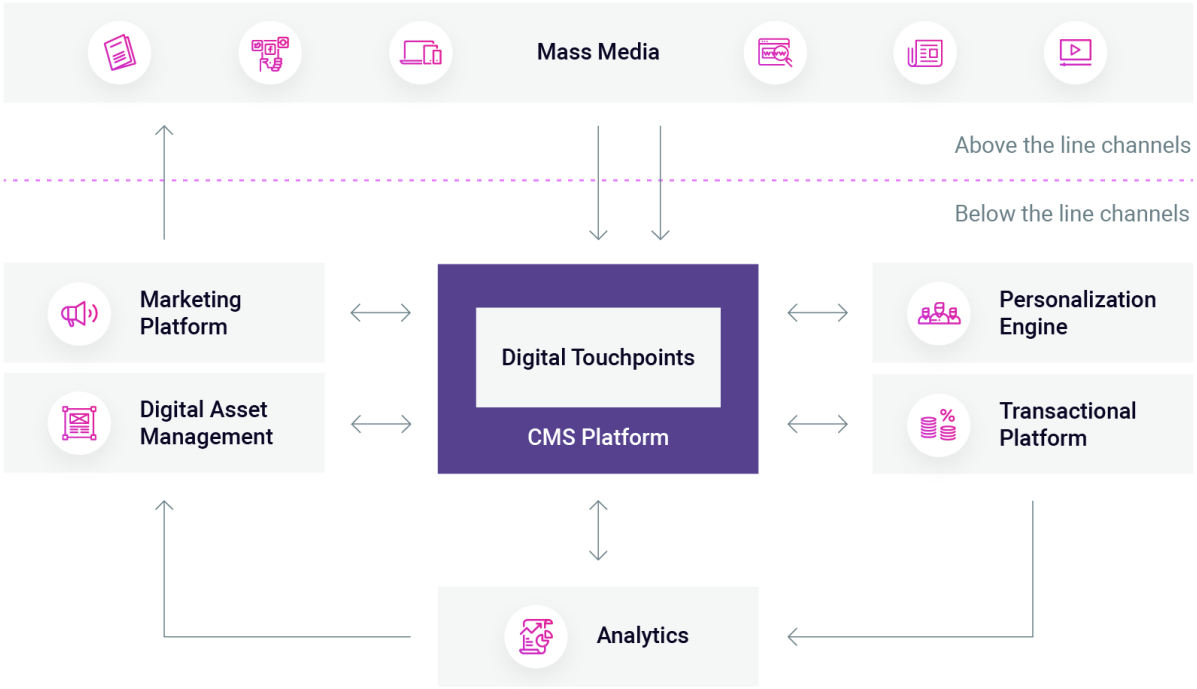
How dotCMS Enables Interoperability & Extensibility

dotcms

Inside the White Paper

Building a digital experience platform (DXP) is expensive. And for the majority of implementations, **the greatest costs of a project can be attributed to systems integration and development.** That's because one software solution will never completely meet the needs of a business out of the box. Enterprises, therefore, often turn to a best-of-breed approach that requires connecting a variety of applications & tailoring them to fit the DXP requirements of their business.

That's why a highly interoperable and extensible platform like dotCMS is critical to lowering the total cost of ownership of a CMS implementation while increasing the platform's overall ROI. With the best-of-breed approach to a DXP, integrated systems that avoid data silos is essential. Software that enables deep integrations, therefore, can lay the foundation for a highly effective DXP for the future.



Let's look at the

multitude of ways that dotCMS allows for interoperability and extensibility

by not only developers but also by business users.

In particular, we'll cover:

- Interoperability vs. Extensibility
- The OSGi Implementation
- The API-Driven Architecture
- The NoCode & LowCode Features
- Summary

Interoperability vs. Extensibility

When it comes to building out a DXP ecosystem, there are two key software architecture characteristics that dictate a successful outcome: interoperability and extensibility.

Interoperability

Refers to how straightforward it is for software to integrate with external systems.

Usually, the most significant challenge developers face when it comes to interoperability is the exchange of data. If the systems can't communicate at all, or the way systems connect isn't robust enough, then data silos will form. That's why developers rely on **communication methods like APIs and standardized frameworks for tightly integrating software.**

With a CMS, for example, interoperability is **crucial when building out a DXP ecosystem because sharing data and services amongst systems** is vital for building a seamless digital experience across touchpoints. If a CMS isn't highly interoperable, data silos could exist within external systems that limit the overall effectiveness of your DXP. If marketers can't access sales information from a CRM within the CMS, for example, they'll lose out on vital information about their customers and target audience. Interoperable software, therefore, **is critical for enabling a best-of-breed approach to a DXP.**

Extensibility

Refers to how easily software can be extended to have additional functionality.

Some software enables this through a modularized core system, accessible data, standardized interfaces, and detailed documentation. The most extensible software, however, is **open-source software that lets developers have full control over the source code** and modify it as necessary.

For CMSs, extensibility is **critical for customizing the software to meet a company's specific DXP needs.** If a CMS doesn't have the ability to create new modules, companies will be stuck with generic functionality that meets the needs of most companies but **isn't tailored to delivering digital experiences for a particular company's audience.** Extensible software, therefore, enables developers to better meet business requirements.

Interoperability	Extensibility
01. Refers to how straightforward it is for software to **integrate with external systems.**	**01.** Refers to how easily software can be **extended to have additional functionality.**
02. APIs & standardized frameworks are used to integrate software.	**02. Open-Source software** that lets developers have full control over the source code.
03. Critical for enabling a **best-of-breed approach to a DXP.** To build a seamless digital experience across touchpoints.	**03.** Critical for customizing the software **to meet a company's DXP needs.** If not it will be stuck with generic functionalities.

Different, But Equally Important

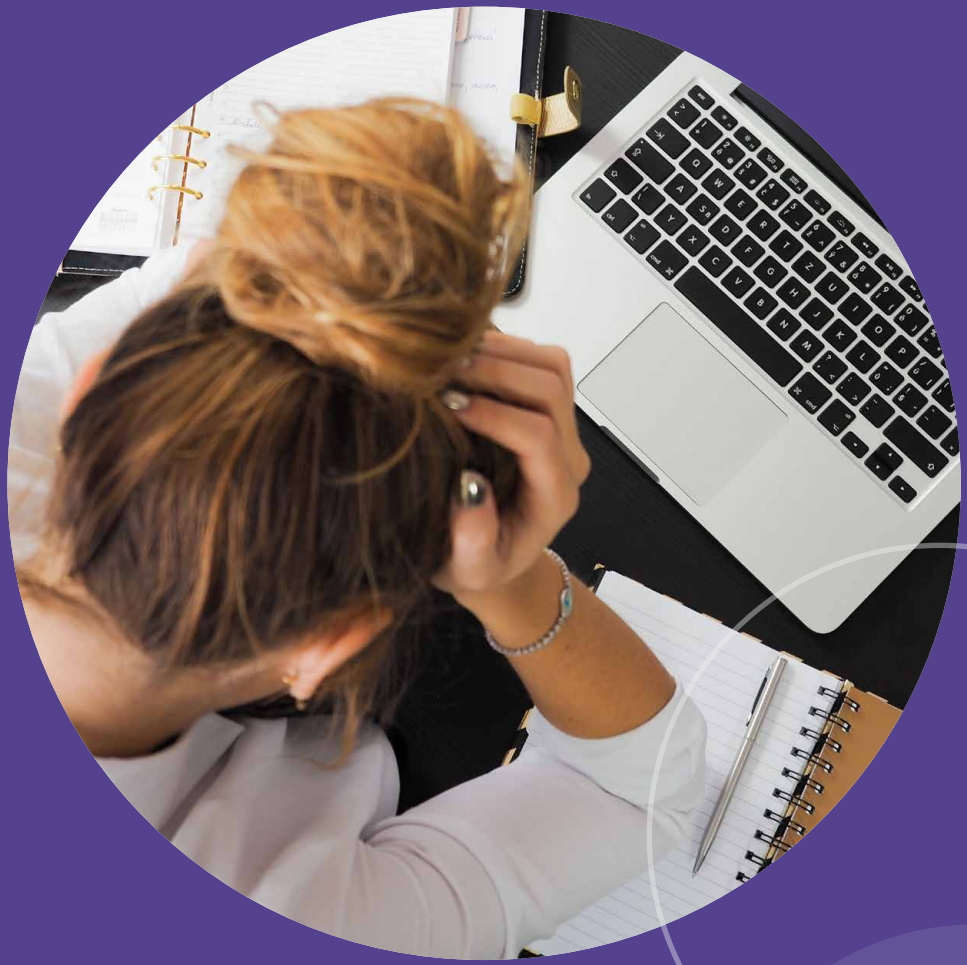
Both interoperability and extensibility are **essential characteristics when it comes to choosing a CMS & implementing it within your DXP ecosystem.**

These are key factors in enabling a best-of-breed approach and reducing the overall total cost of ownership of implementation projects. As we'll see in the following sections,

dotCMS has a multitude of features that make business-specific integrations & customizations possible.

The OSGi Implementation

In a Java programming environment, one of the **biggest challenges is modularization.** While component-based software makes the development of individual functionality easier, as software gets larger it can become more complex. It thewn becomes difficult for developers to **maintain the communication between various services without running into incompatibilities.** With this in mind, let's take a closer look at OSGi and how it's a solution to many of these issues.



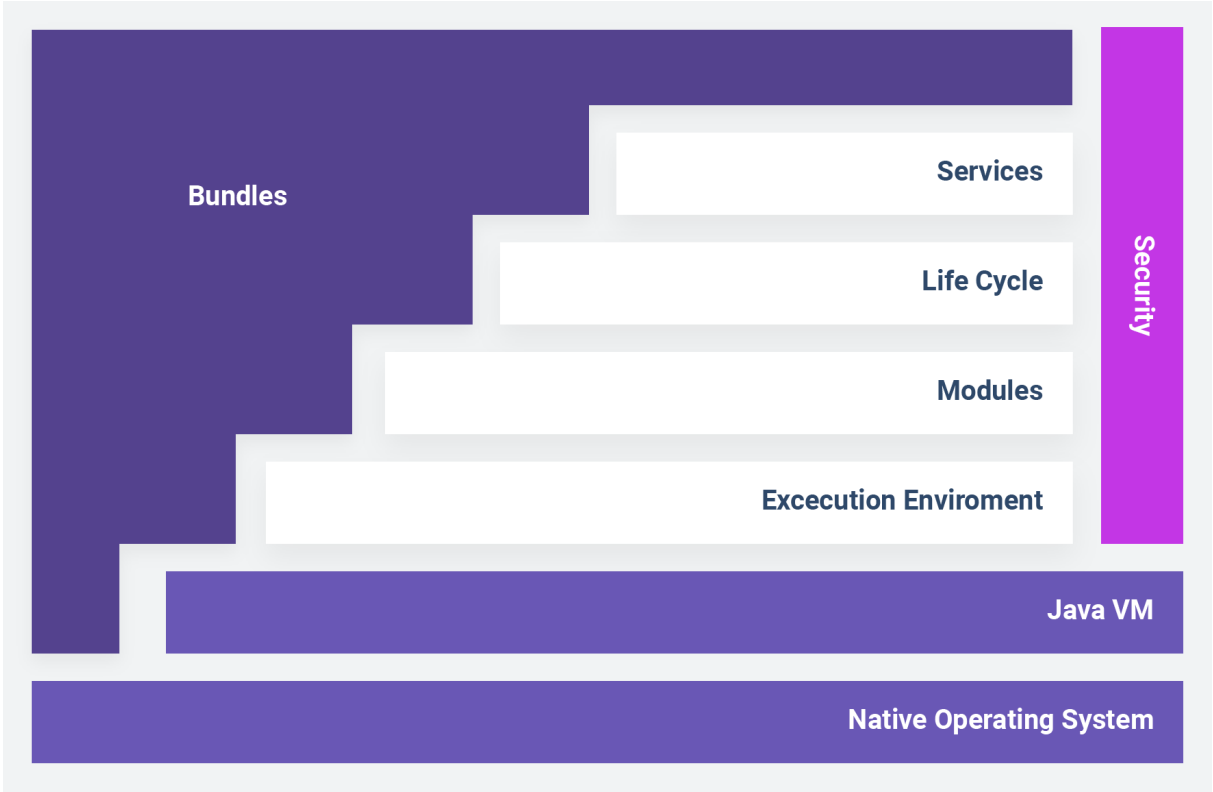
What is OSGi?

The Open Service Gateway Initiative (OSGi) defines a **Java-based framework that makes it easier to manage many different independent software components.** The core tenants of the framework are encapsulation and loose coupling of components.

01. Encapsulation: means the internal workings of components and their dependencies are self-contained.

02. Loose coupling: means that components can run independently while relying on other components as little as possible. These attributes make highly modularized software much easier to maintain.

With OSGi, independent software components are known as bundles, which run in an OSGi container. Once a container is deployed, bundles can be installed and managed without requiring the platform to reboot because the OSGi service registry automatically detects bundle changes. These bundles run in isolation with all of their own dependencies included.



This means developers avoid the challenge of resolving dependency conflicts — different versions of libraries or Jar files running in the same Java Runtime Environment. The bundles also hide their internals from each other and communicate through limited interfaces. That's why running bundles in isolation has the added benefit of reducing the impact that additional code can have on core source code.

How dotCMS Implements OSGi

dotCMS implements OSGi using Apache Felix, so that bundle plugins can be hot-swapped while the CMS is running. These plugins include:

- **Viewtools**
- **Actionlets**
- **Service**
- **Custom Admin Portlets**
- **Servlets & more**

OSGi is enabled by default, so new plugins can be added at any time manually or using the Dynamic Plugins Portlet web interface.

Beyond managing bundles, dotCMS gives you the tools necessary to develop your own plugins. Using an Eclipse development environment, you can write an entirely new plugin in Java by following the examples dotCMS provides.

You can quickly install your new plugin while ensuring the custom code is isolated from the core dotCMS code.

Further Reading:
OSGI Explained: Extending Your Software to Embed an OSGi Framework >>

Why OSGi Improves Interoperability & Extensibility

When it comes to extensibility, OSGi lets developers **quickly create custom functionality and integrate the new code with dotCMS** without worrying about versioning conflicts. The code is completely isolated, so major or minor software upgrades of the core platform won't be impacted going forward.

This means developers are free to create business-specific capabilities & deploy them without impacting the platform's durability.

Extended functionality can include deep integrations with third-party applications, or additional features within the dotCMS interface.

OSGi also enables **interoperability with other applications because each bundle is entirely independent.** This means the dependencies one component needs won't conflict with those necessary for other applications or components when deployed within the OSGi container. Dependency conflicts are a common issue many developers face when deploying applications to web containers. **dotCMS has many ready-made OSGi plugins for common use cases,** which dramatically reduces the time to market and cost for integrating with the most popular external applications.

OSGi reduces the complexity of adding additional functionality and integrating with third-party systems

by leveraging a transparent component-based framework for building your DXP. For these reasons, OSGi leads to a faster time to market when implementing dotCMS.

The API-Driven Architecture

Along with the Java-based OSGi framework, there are situations where developers want language-agnostic integrations. OSGi modules aren't practical for integrating with systems developed in languages other than Java. That's where APIs come into play.

There are many types of APIs from REST to SOAP, but what they have in common is enabling programs written in different languages to communicate with standardized data formats like JSON or XML. The ability to exchange information has become crucial with the rise of microservices, and the best-of-breed approach. That's why there's been a **30% increase in new APIs released in 2019 than the previous four years.**

What is REST?

REST is an API design style that mimics the internet itself. This means a

RESTful API follows the HyperText Transfer Protocol (HTTP) methods like GET, PUT, PATCH, and DELETE.

They're also stateless, which means **each API request and response is independent and self-contained.** Statelessness enables the loose-coupling of systems when it comes to integration.

When it comes to the actual data transferred, **most REST APIs use JSON content for formatting because it's easier to read than formats like XML.** In more recent years, REST API standards have become more popular than traditional SOAP web services because they're usually more straightforward to work with and often use less bandwidth. **There's been a proliferation of APIs, and most SaaS applications now expose an APIs with a range of functionalities.**

What is GraphQL?

GraphQL is a querying language for APIs that

lets developers describe the data they want and get all of it back from one endpoint.

The standardized querying language hides the implementation details of the underlying APIs from developers, so **they don't need to know where specific resources are located or the structure of the backend server.** GraphQL is introspective, which means the API can be queried for the data types it supports, so there are fewer challenges with versioning and code changes on the backend.

Since GraphQL requests retrieve exactly what's been described, overfetching and underfetching of data is virtually eliminated. This means **fewer API calls and less bandwidth usage in the long run.** Many development teams have been moving to GraphQL because of its ease of use and potential performance increases.

Further Reading:
How dotCMS & GraphQL Combine to Simplify Headless Content Management >>

...objects and content can be created, modified, and accessed using the APIs.

How dotCMS Implements APIs

dotCMS has out of the box support for both REST APIs and GraphQL. These dotCMS APIs have thorough documentation and follow industry standards for security, performance, and control.

When it comes to security, dotCMS APIs support a multitude of **authentication methods like Basic Authentication and JSON Web Tokens (JWT) to validate the identity of users.** Beyond that, dotCMS APIs support authorization down to the granular-level, which lets administrators set permission rules on specific pieces of content.

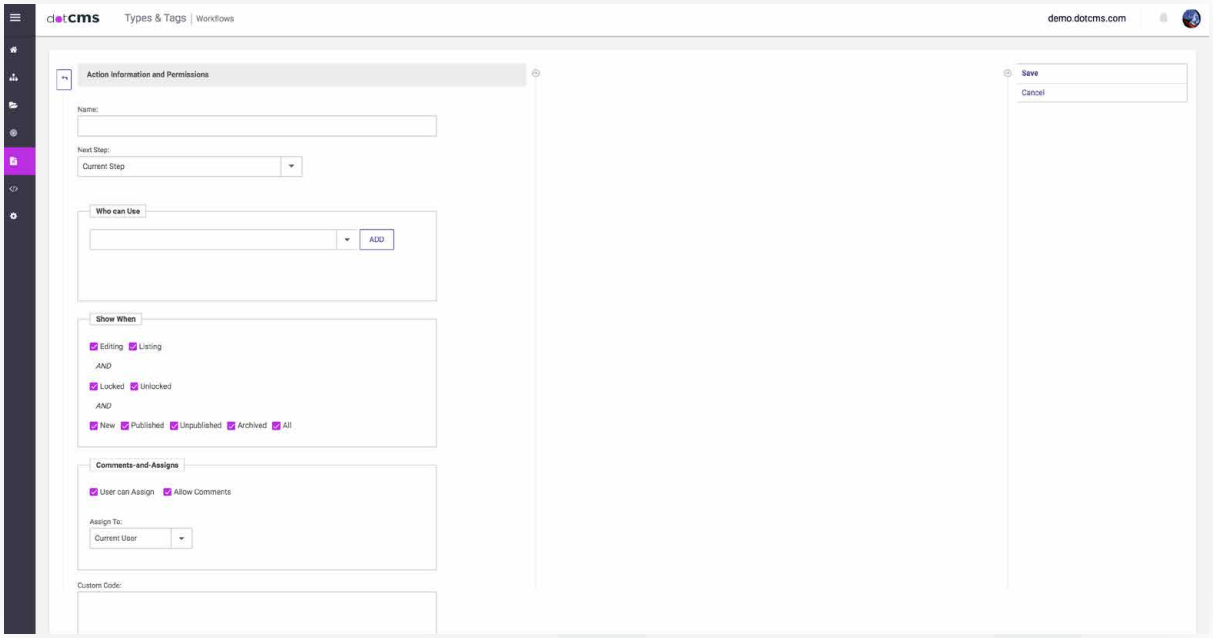
With the dotCMS REST APIs, you can perform nearly any function — which we call Everything as a Service.

There are hundreds of REST endpoints to work with, which are **fully documented here.** Using the APIs, **objects and content can be created, modified, and accessed using the APIs.** The dotCMS configurations can be modified, and workflow actions like publishing can be triggered through APIs as well. Here are more details on some of the most commonly used APIs:

01. Content API:

Gives developers control over content and assets. This includes full CRUD — **create, read, update, and delete** — capabilities for any content type. The Content API is the most common way to work with content. This API can be used to ingest content from third-party systems or pull content for frontend applications.

Read More >>



02. Content Type API:

Lets developers **manage content types.** When integrating with third-party systems, this API is useful for retrieving the structure of various content types before implementing data manipulation processes.

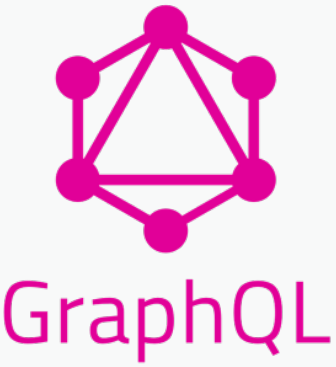
Read more >>

03. Elasticsearch API:

Allows developers to search & retrieve content quickly by leveraging an Elasticsearch-index content repository.

The API uses the Lucene syntax for basic queries and full Elasticsearch JSON syntax for more advanced features like aggregation and geolocation searches.

Read more >>



04. Workflow API:

Lets developers create, modify, and execute workflows and workflow actions. Through this API, **third-party applications can trigger workflow actions and move content** through to the next step. External translation services, for example, can receive content from dotCMS, translate it, and use the Workflow API to notify dotCMS that the translated content is ready.

Read more >>

05. Layout API:

Allows developers to **retrieve everything necessary for a web page in a single API call** from the page layout to its theme, content, and widget components. This Layout as a Service approach **gives marketers control over web page layouts** while still enabling developers to integrate with the **frontend technologies** of their choice.

Read more >>

06. Navigation API:

Lets developers **retrieve navigation and menu information for a website based on the dotCMS file and folder tree structure.** This means frontend applications can have dynamic site navigation.

Read more >>

The GraphQL APIs that dotCMS gives developers access to the entire content repository

using the querying language. This means developers have added flexibility in shaping the responses they get back to better fit with the data formats required for new or legacy applications.

You can find out more about the dotCMS GraphQL implementation here >>

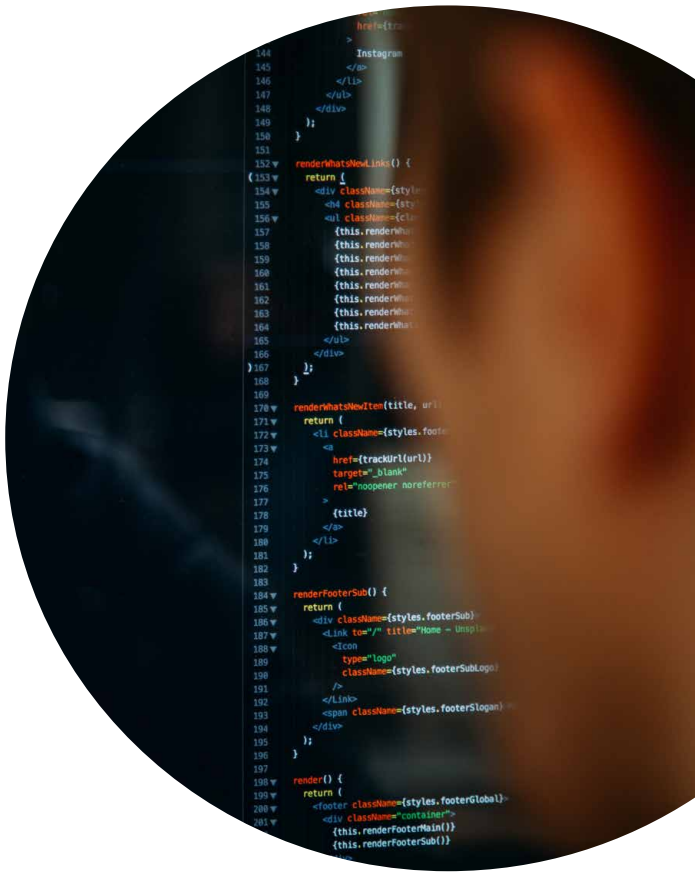
Why an API-Driven Architecture Enables Agile Development

The API-driven nature of dotCMS means companies can

access content at whichever level they want, from whole pages down to layout components or individual content.

This means developers can use the traditional CMS approach by delivering static HTML pages or they can provide fragments of pages in JSON format to integrate with SPA frameworks like React or Angular. Developers have complete flexibility when it comes to content, workflows, and most other dotCMS functionality.
Everything as a Service streamlines the integration with front-end frameworks and enables developers to deliver web apps faster than ever.

Besides integrating with nearly any front-end, the dotCMS APIs **enable developers to connect with a variety of third-party applications.** Data can easily be pulled from external systems or actions within external applications can trigger functionality within dotCMS. Most modern web apps expose APIs, so the integration options are truly endless. With dotCMS, building a best-of-breed DXP is straightforward.



The NoCode & LowCode Features

As enterprises look to deliver web apps to market faster, at a much lower cost, they've been increasingly **turning to NoCode and LowCode solutions.** That's because **enterprise IT teams are spending 60% of their team maintaining existing application,** which leaves little room for digital innovation. For many IT implementation projects, software development is a slow and costly endeavor. Reducing heavy development with a lighter scripting language like Velocity or even drag-and-drop interfaces can dramatically reduce the costs of building a DXP ecosystem.

With that in mind, here's a number of dotCMS features that streamline development, and even enable non-technical staff to get involved in the implementation process.

Scriptable API Builder

While dotCMS has powerful native APIs, some companies may wish to **create their own custom endpoints** to meet unique business requirements. With the scriptable API builder, this is now possible without requiring Java code.

Using low code Velocity scripting, developers can create **new endpoints to pull, manipulate, and render content** however they deem necessary.

Lightweight velocity endpoints have full access to requests, responses, & authentication information just like REST endpoint.

In addition, there are a variety of Viewtools — Java classes with functionality that's accessible from Velocity code.

Velocity scripting requires much less technical knowledge, so it's no longer necessary for developers to build custom endpoint using heavy Java programming. In many cases, business users can even learn enough Velocity to perform some tasks like creating templates themselves.

Scripting as a Service

Scripting as a Service takes Velocity-based API endpoints a step further. Developers can now dynamically **generate API responses at runtime by including Velocity code with API requests.** This means external applications can define APIs that fits exactly what they need, which makes integrating with legacy applications much easier than in the past.

Rendering an endpoint dynamically with dotCMS is straightforward. **Developers can make an API request to the "dynamic" REST endpoint and include a "velocity" field in the JSON content that contains a Velocity script.** This Velocity code would be executed on the fly as if it was already a defined custom lightweight endpoint.

Scripting as a service lets

front-end developers create customized endpoints for web apps without relying on backend Java developers.

This can reduce the time to market for integrating with legacy applications as well.

[Watch Scripting as a Service Demo >>]

NoCode Every Step of the Way

While not directly implementation features, there is a multitude of **NoCode tools for marketers** that give developers some breathing room. Creating content types and workflows are tasks that should be done by business users, but most platforms require developer assistance.

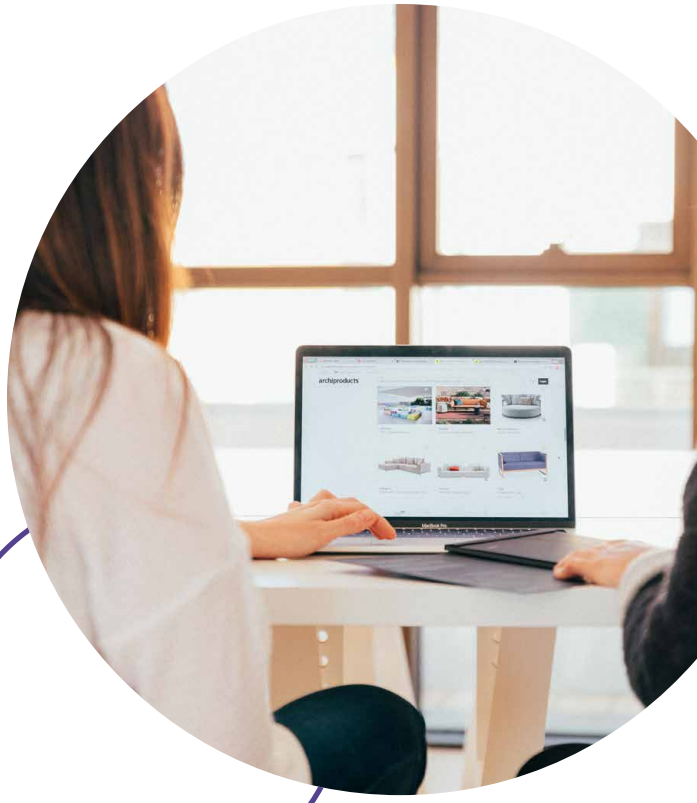
With dotCMS, non-technical users are empowered to fulfill many tasks that free up IT staff

to work on more complex integration tasks. These NoCode features, therefore, indirectly contribute the interoperability and extensibility of dotCMS, and reduce the overall ROI of building a best-of-breed DXP.

Benefits of NoCode & LowCode

And it's not just non-technical users that benefit from NoCode and LowCode, but developers as well. dotCMS streamlines the development process when it comes to API integrations with its API Builder and Scripting as a Service capabilities. NoCode shifts development teams from low-level tasks to building unique business logic. In most situations, **Velocity scripting is significantly faster than Java development**, so developers can complete integration projects faster than ever before.

The benefits for organizations are enormous when it comes to reducing software development requirements. **Reduced development needs increases productivity for developers and marketers, which increases the ROI** of implementation projects. That's why dotCMS continues to develop features that fit its NoCode philosophy.



Why Interoperability is Key to DXP Success

When building a best-of-breed DXP solution for your organization, interoperability and extensibility are critical factors to consider. That's why **open source software like dotCMS can act as a content hub and foundation for enabling a seamless DX ecosystem.**

A best-of-breed approach is not only possible with dotCMS but a guarantee. With dotCMS, you're choosing a low total cost of ownership for your DXP ecosystem.

dotCMS is built from the ground up for integration and customization. It's API-driven architecture and OSGi support enable developers to choose the best approach for each integration point — whether it's a front-end customer-facing app or back-end system like a CRM or marketing analytics tool. If further customization is needed, developers can look towards LowCode API tooling to streamline the process further.

In the dotCMS 5.x Series, dotCMS doubles down on integration capabilities with GraphQL & Scripting as a Service.

The multitude of integration options reduces the development cycle and leads to better adaptability for the future.

Altogether, dotCMS has the capabilities necessary for implementing the platform fast and reducing the overall costs of IT and development. A best-of-breed approach is not only possible with dotCMS but a guarantee. With dotCMS, you're choosing a low total cost of ownership for your DXP ecosystem.

Complimentary Evaluation Support

dotCMS offers a variety of tactics to test-drive and proof out your key use-cases around your personalization strategy. It is our investment and helps you to evaluate dotCMS effectively, way beyond shiny product demos and slick sales presentation.

More on our evaluation support
Here>>>

About dotCMS

dotCMS is a leading, open source content and customer experience management platform for companies that want innovation and performance driving their websites and other content-driven applications. Extensible and massively scalable, both small and large organizations can rapidly deliver personalized and engaging content across browsers, mobile devices, channels, second screens and endpoints -- all from a single system.

Founded in 2003, dotCMS is a privately owned US company with offices in Miami, Florida; Boston, Massachusetts and San Jose, Costa Rica. With a global network of certified development partners and an active open source community, dotCMS has generated more than a half-million downloads and thousands of implementations and integration projects worldwide. **Notable dotCMS customers include:** Telus, Standard & Poors, Hospital Corporation of America, Royal Bank of Canada, DirecTV, Thomson Reuters, China Mobile, Aon, and DriveTest Ontario.

Miami
3059 Grand Av.
Miami,FL,33133
U.S.A

Boston
200 Portland St.
Boston,MA,02114
U.S.A

Heredia, Costa Rica
Eurocenter
Primera Etapa, 2nd Floor
106 Heredia, Costa Rica

ON-DEMAND DEMO

dotcms.com

+1-305-900-2001

sales@dotcms.com