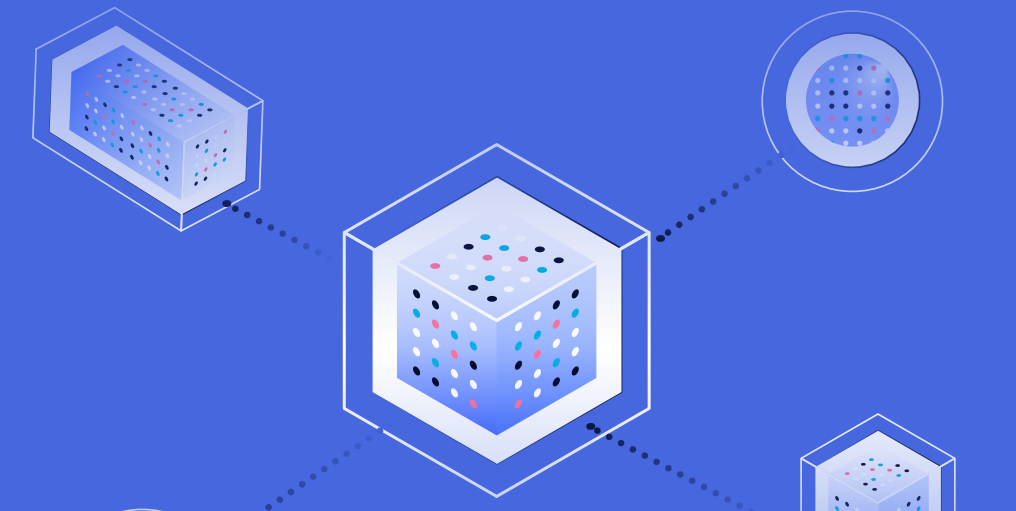
Updated May 2019

altis
{white-papers}

Headless WordPress: The Future DXP

Omnichannel and progressive solutions for enterprise, featuring case studies from:



By Tom Willmot & Joe Hoyle

CONTENTS

About the authors

Tom Willmot is the CEO of Human Made, where he is responsible for company direction, project management, and client relationships. When not working with clients he contributes back to the open source community.

Joe Hoyle is the CTO of Human Made, where he leads the development team and the overall technical direction of the company. He is a member of the WordPress REST API development team.

Contributors:

- **Siobhan McKeown**
- **Ryan McCue** — *WordPress REST API co-lead*
- **Daniel Bachhuber** — *WordPress REST API team*
- **Noel Tock**
- **Petya Raykovska**
- **Ant Miller**
- **Robert O'Rourke**
- **Barbara Marcantonio**
- **Nevena Tomovic**
- **Ana Silva**

Human Made & WordPress

Human Made are one of the leading contributors to the WordPress project, and have been actively involved in developing the software since version 3.0. Our involvement in WordPress spans years of modifications and evolutions and we have seen dramatic changes to the way WordPress has been adopted across a range of industries and enterprises.

Executive Summary

WordPress is an open-source content management system which is used to power millions of websites and blogs, and an increasing number of web applications. It currently powers more than 33% of the top 10 million websites on the Internet. WordPress' usability, extensibility, and mature developer community make it a popular choice for websites of all sizes.

WordPress, like many other CMSs, is monolithic. It provides everything you need to run a website, and can be extended further with third-party plugins and themes. But on today's web, we are moving beyond the monolithic CMS. WordPress, along with others, is leading the charge to a future where the CMS acts as a central hub, consuming and aggregating content and data from other tools and services and in-turn exposing its own content and data via APIs.

The WordPress REST API is a huge step towards this future. Exposing WordPress content and data as JSON via a standardised RESTful API unlocks your data and will enable an explosion in the number and complexity of integrations.

By embracing the WordPress REST API, you can more easily: separate your frontend delivery from the CMS, power multiple frontends from the same content (think a website, app, Apple News, etc.), and use WordPress as part of complex multi-service workflows (like pushing content to a separate service for translation before pulling those translations back into WordPress).

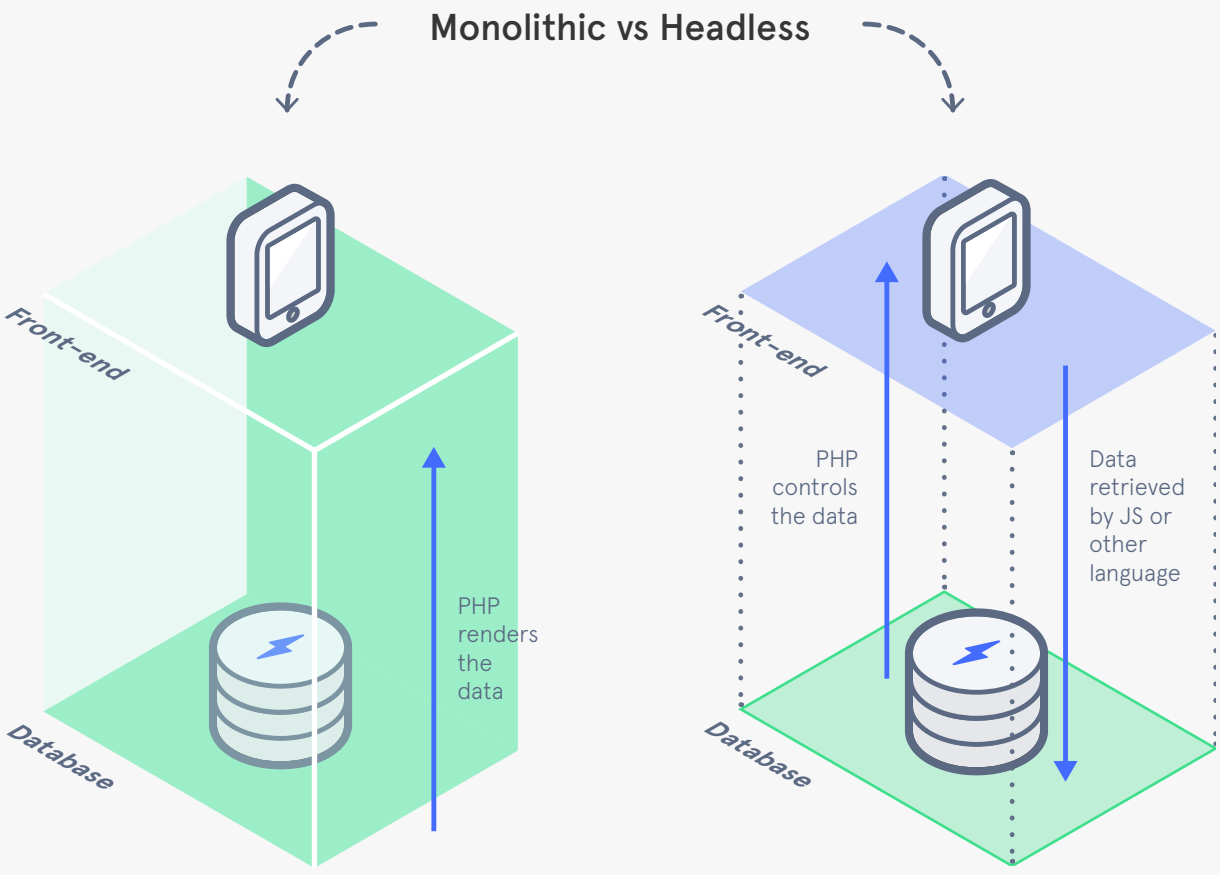
The potential implications for your business are far-reaching, particularly for large custom builds and applications. WordPress provides content management and content capture, while making the data available to other frontend technologies. This permits engineering teams to work independently on discrete parts of a larger project, and allows for more stable third-party integrations. With the REST API, WordPress stops being a web development tool used in isolation. It is one module that is available in a web developer's toolkit; a building block to be used in many kinds of applications.

01 The Headless CMS

A headless CMS is used only for data capture, storage, and delivery, making it frontend agnostic. Its data can be displayed using any frontend technology, whether in a browser, mobile application, syndication, or elsewhere.

A traditional CMS deals with data collection, delivery, and display. WordPress, for example, has a backend where users can enter data. This data is stored in a MySQL database, retrieved from the database using PHP, and then displayed in the browser using the Theme system.

A headless CMS decouples the Theme system, allowing you to replace it with the frontend technologies of your choice. What's left is the data storage method and web application for authors and editors, while the data is delivered to the frontend using an API.



Decoupling content management from frontend display

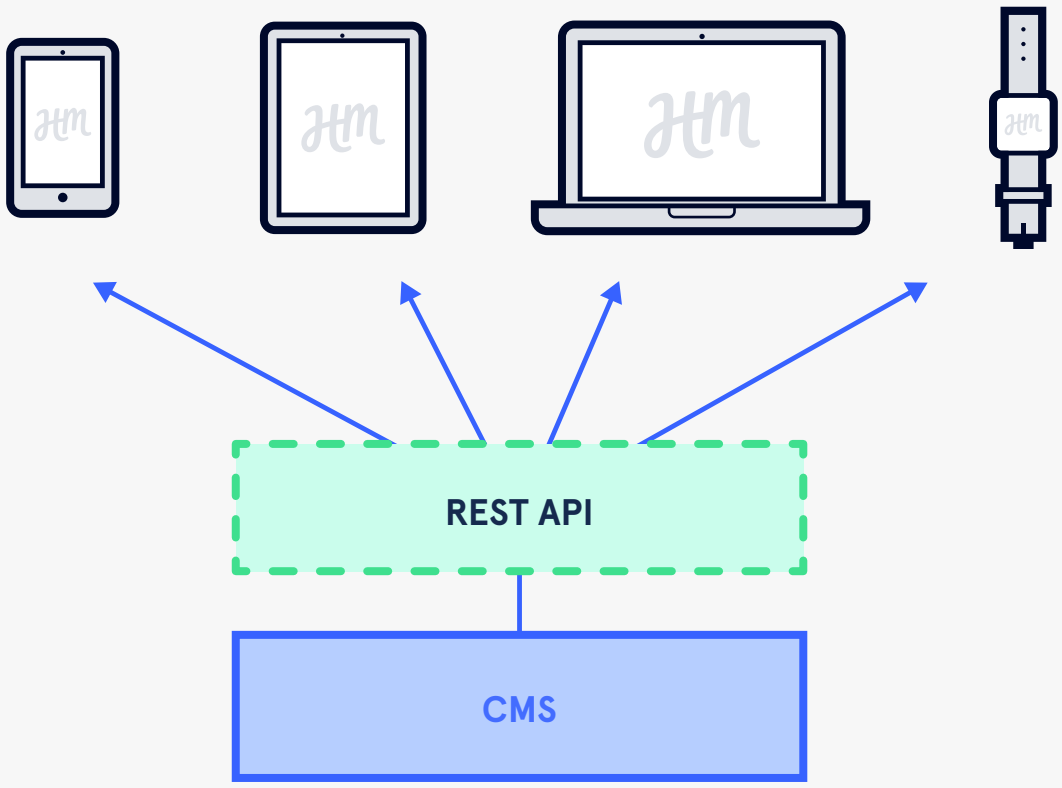
By decoupling content management from frontend display, a headless CMS allows developers to use any technology to display content. Developers are not locked into the templating engine provided by the CMS. The CMS might be written in PHP, but developers working in languages like JavaScript, React, React Native, and Vue.js, can use an API to retrieve, store, and display data. A frontend developer has complete control over the website or application's markup and user experience, using client-side technologies to create smooth interactive experiences. It also means that if the frontend needs to be displayed in a new way (for example a redesign or to display content on a new device) the CMS can still hold the data, removing the need for complex migrations.

Fast, interactive experiences

When you use a headless CMS there are two components: the CMS itself and the frontend display. By splitting up your website or application in this manner, you can improve performance and provide a superior user experience. The CMS focuses only on content management, without having to assemble formatted responses, while the client-side technology can quickly display that data in the browser. Using client-side technologies for display means that in-browser experiences are fast, acting in real-time, without having to wait for PHP to generate entire pages. There is a significant increase in performance when using JavaScript vs PHP: Node.js, for example, can handle many more requests than PHP due to its asynchronous event driven nature. This can be especially useful when an application requires many connections open simultaneously.

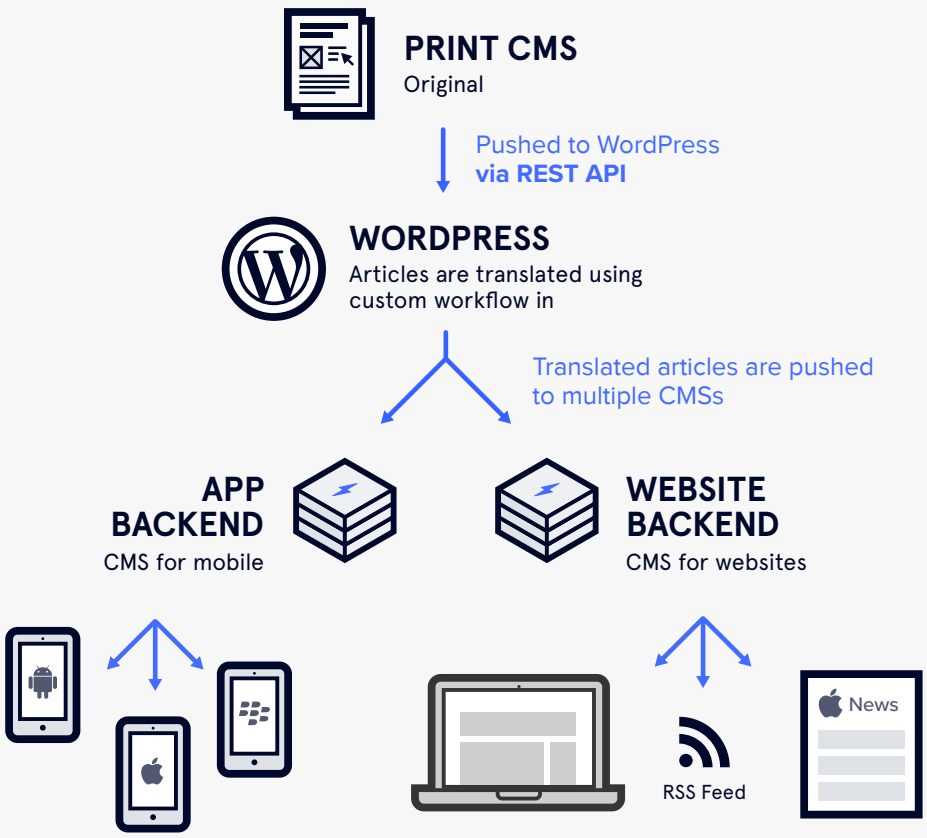
One content management system, multiple frontends

With a traditional, monolithic CMS, data is simply displayed by the CMS itself. Data stored in a headless CMS is available for display in any context. You may want to use it for a website now, but later you may decide to use the same data for a desktop or touch screen application. The stored data is always available via the API.



Multi-service content pipelines

A headless CMS can be used to store all of the data for one site or application, or it can just be **one element of a larger application** that retrieves and aggregates data. This means that data can be integrated into existing workflows as just one layer. For example, it could be used just as a layer for translating content which is then pushed to another CMS.



02 CASE STUDY: TECHCRUNCH

We joined TechCrunch as technology partners for their latest renovation project, resulting in a full site rebuild and a WordPress CMS with a headless React frontend, on traditional managed WordPress hosting.

https://tech-crunch.com/

Why use WordPress and the REST API?

TechCrunch adopted WordPress and the REST API to help them decentralise their publishing experience: ensuring they could keep the editorial simplicity inherent with a WordPress backend, whilst making use of the REST API to create a user friendly frontend. Maintaining WordPress' backend on their system enables their non-technical teams to independently run individual pages: making use of React on the frontend enables editors and authors to create tailor made solutions for each new page, empowering more people in their team to write and publish content effectively.

The build

With TechCrunch, we implemented a headless CMS on a PHP-only hosting infrastructure, hosted with WordPress.com VIP Go. We introduced our own VIP Go Builder which was later adopted by WordPress.com VIP, adding a build process that only has to deploy the master-build branch to send all merged changes live to the production website. For our local development environment, we extended the React app tool with WordPress specific customisations. This meant we could load the application and the stylesheets from WordPress.

We used React on the frontend to display data, which required us to repeatedly write the same boilerplate code to retrieve data from WordPress. The repetitive nature of this task inspired us to create Repress, a Redux Library for the WordPress REST API. This enables developers to retrieve data from the REST API, and add it to the store with just a few lines. Unlike many other React libraries for WordPress, Repress can be added to an existing store, allowing progressive adoption; and can even be combined with other methods of retrieving data from the WordPress REST API. Repress not only enabled us to facilitate the process of retrieving data from WordPress; it also introduces a higher order component to interface with React, making it easy to add data management to presentational React components.

On the frontend we were able to create specific page functions, like the river homepage experience

that creates a seamless and responsive user experience; enabling users to scroll and click on an article, that then snaps closed to resume their exact browsing position on the homepage. We also introduced the developing stories feature, helping TechCrunch's editorial team break news stories faster and more effectively. The developing story feature allows for short pieces of content to be displayed, and later grouped under one overarching headline to be viewed as a whole.

We also integrated their partner database for startup news, Crunchbase. Our team of engineers integrated the Crunchbase API with the React frontend to display detailed information on companies, investors, founders, and more, as hover cards within the articles.

Read more about Our React Tools for WordPress by Ryan McCue.

NICOLE WILKE
Head of Product, TechCrunch

"We wanted our redesign to be ambitious, and it was critical for us to choose a development partner who could deliver it without compromising on our aspirations.

Our design focused on fluid and interesting interactions over bold visual statements, heavily influenced by our decision to use the WP REST API with a JavaScript frontend. As such, it made sense to choose a partner that brought expertise in that realm, which Human Made certainly did."

03 What is a REST API?

POST, GET, PUT, DELETE: Dive into what a REST API is, why it’s RESTful, and what makes it open.

What is REST?

Representational State Transfer (REST) is a software architectural style for Application Programming Interfaces (APIs) that consists of guidelines and best practices for creating scalable web services. REST uses simple HTTP to make calls between machines. This happens via a request/response mechanism between the server and the client. For example, a client, let's say an Android application, makes a request for the most recent posts from the website. The server knows how to interpret this request, through REST, and satisfies the response by providing the most recent posts in a format understood by the client.

REST requests interact with the resources in your application (e.g. a Post or Page). These interactions are typically Reading, Creating, Updating, or Deleting. Combined with HTTP, REST requests are formed using four verbs:

- POST: Create a resource
- GET: Retrieve a resource
- PUT: Update a resource
- DELETE: Delete a resource

The data retrieved is supplied in a machine-readable format, often JSON in modern web applications.

What makes an API RESTful?

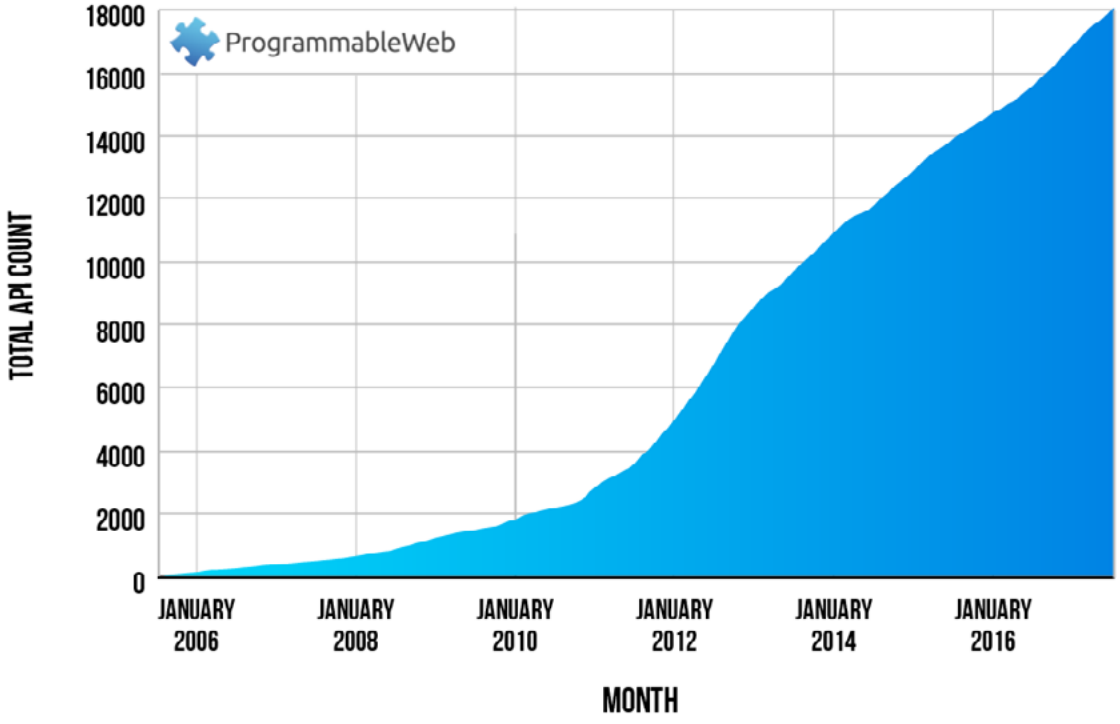
An API must have the following architectural features to be considered RESTful:

- **Client-server:** the client is separated from the server. This means that clients are not concerned with data storage and servers are not concerned with display. This ensures that data is portable and can be reused in multiple clients, and servers are simpler and more scalable.
- **Cacheable:** clients can, and should, cache responses to improve performance, and avoid the server with every request.
- **Stateless:** the necessary state to handle the request is contained in the request itself, whether as part of the query parameters, URL, body, or headers.
- **Uniform interface:** information transferred via REST comes in a standardised form, creating a simplified, decoupled architecture.

- **Layered System:** the architecture is composed of hierarchical layers. Each component cannot "see" beyond its layer: a client cannot tell if it's connected to the server or to an intermediary.
- **Code-on-Demand:** REST allows client functionality to be extended by transferring applets or scripts.

A separate, but closely-related concept is hypermedia. Similar to how hyperlinks on the human-readable web enable discovering new sites and content, hypermedia allows a client to more fully discover a REST API without needing to know anything about the structure of the API. Instead, the server provides whatever information the client needs to interact with it. This means that the

Growth in Web APIs since 2005



Source: Program-mableWeb

client can interact with the server in complex ways without knowing anything beforehand about it.

What is an Open API?

Open APIs are publicly available APIs that give developers access to proprietary software information that they can make use of in their own software and applications. REST is the ideal architecture for creating an Open API for the web because, by using HTTP, it is built on the principles of the open web. To leverage an open REST API a developer just needs to make a HTTP request.

By making data available for developers to use in their own applications, open APIs are transforming the internet. Developers can access data across services, creating applications that aggregate information from different providers, and leveraging that data to their own needs. The impact of APIs cannot be underestimated; they are transforming the way businesses and services are run. For example:

- Around 25% of annual revenue of the fundraising platform Justgiving is API-driven
- In 2011, Twitter reported that they had more than one million applications registered, with a number of entire companies built off the API
- The Skyscanner API is leveraged by startups who make use of its data feeds
- Hilton is making use of Uber's API to allow guests to book rides from the Hilton Honors App

This aggregation of public data across different platforms enables the creation of feature-rich,

04 WHAT IS THE WORDPRESS REST API?

Transforming WordPress into a headless CMS with the REST API.

powerful applications that do more than any individual product or service could do on its own.

The WordPress REST API allows access to a website's data, including users, posts, and taxonomies. In the past, developers needed to use WordPress' built-in Theme system and administration panel to display and edit content on a site. The REST API decouples the WordPress backend from the frontend, allowing developers to use it as an application platform: WordPress is used for data entry and storage, and the frontend can be built in any programming language. **The REST API transforms WordPress into a headless CMS.**

Endpoints

Endpoints are functions that are available through the API: they're the places where developers can do something with the CMS, whether that's creating, retrieving, updating or deleting (CRUD) data. This includes the four core data types in WordPress (posts, comments, terms, and users) initially, although these will grow in future versions of WordPress to support all data on the site. If you are building a website, application, theme or plugin, you can leverage the API by adding your own custom endpoints.

Authentication

A major challenge around building a REST API is authentication: how does an API know that a user should be allowed to update content on a site, for example? Who should be allowed to retrieve data? Under what conditions? The WordPress REST API uses two forms of authentication:

- **Cookie** — this is the basic authentication method used in WordPress. When you log into your dashboard a cookie is set in your browser. This method is only viable when the current user is logged into WordPress and that user has the capability to perform the action requested.
- **OAuth** — this is the main authentication method used for external clients, i.e. any third-party site or application that wants to interact with the API. With OAuth, logged in users can authorise clients to act on their behalf. Clients are issued with OAuth tokens so they can interact with the API.

The REST API is now focusing development on OAuth 2.0, after recently upgrading from OAuth 1.0. OAuth 2.0 requires HTTPS, and due to the increased uptake of SSL across the web, the decision was made to transition to the more modern plugin. Additionally, OAuth 2.0 provides a more streamlined solution to the distributed API problems.

As well as these methods, there is a Basic Authentication method for external clients. However, this is only recommended for development environments as it involves passing your username and password on every request, and giving your credentials to clients.

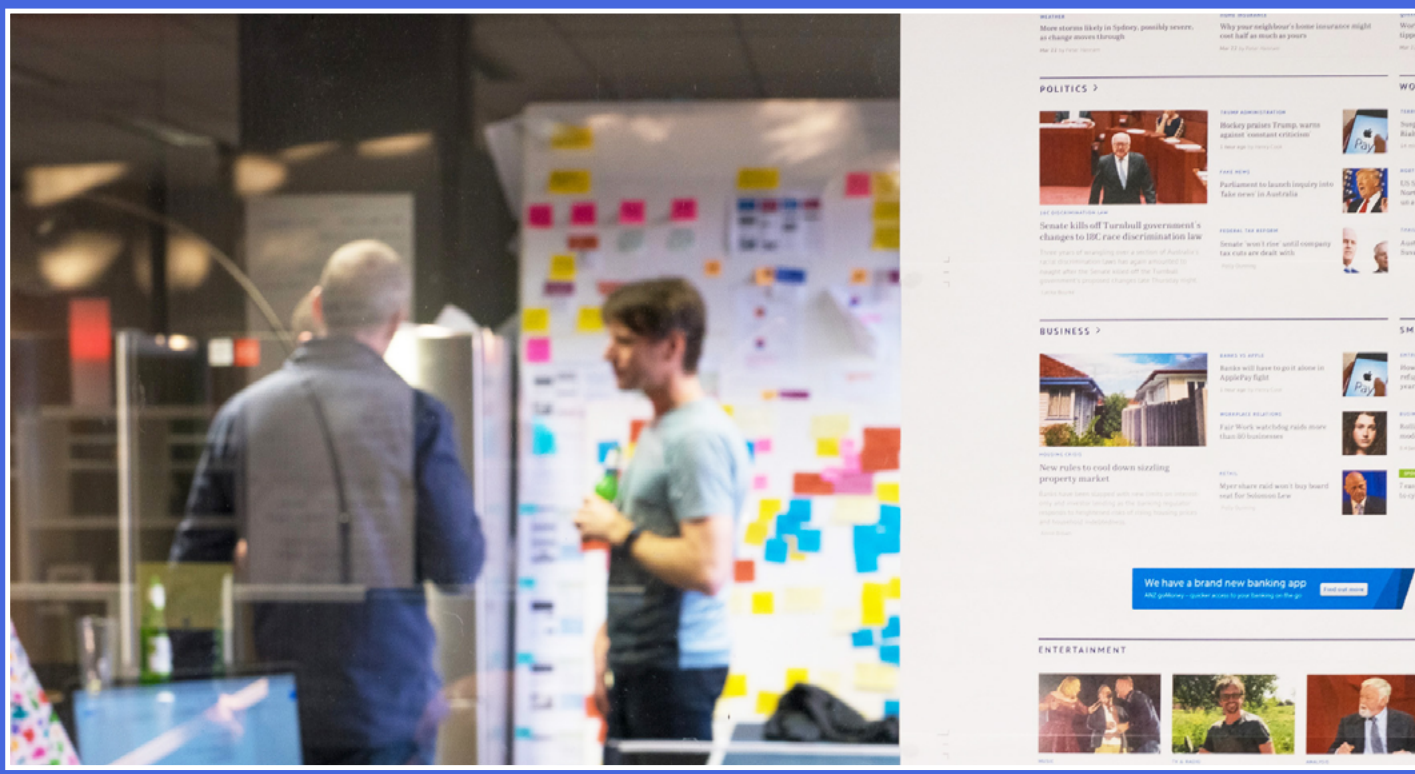
The Team

The WordPress REST API has contributions from 96 developers. However, the team has five main contributors:

- **Ryan McCue** (Human Made), co-lead of the REST API project
- **K. Adam White** (Human Made), co-lead of the REST API project
- **Joe Hoyle** (Human Made)
- **Daniel Bachhuber** (Hand Built)
- **Rachel Baker** (Wirecutter), co-lead emeritus of the REST API project.

05 Case Study: Fairfax Media

We joined Fairfax Media for an end-to-end newsroom transformation, and utilised the REST API to improve editorial workflows and publishing processes for Australia's leading media brand.



Fairfax Media

https://www.fairfaxmedia.com.au/

Why use WordPress and the REST API?

Fairfax Media wanted a tech partner to support them through their latest digital evolution, and implement solutions to streamline and improve their editorial and publishing experience.

They approached Human Made with a large-scale initiative: building a custom CMS based on headless WordPress, with a modern publishing workflow, and an audience facing React.js based frontend (both of which were developed in-house). The REST API was instrumental in this process; enabling us to update, streamline, and improve their editor screen, and helping us build a modern newsroom experience.

The build

Fairfax wanted to move quickly towards new and more effective publishing workflows for their team of editors and journalists. One of the first tasks we undertook was to create a custom feature to allow journalists to search the Fairfax Content API for wire feeds from other news sources. This allows journalists to select an article that is then injected into WordPress for editing, customisation, and publishing. Once we'd built the wire feed importer we began to refine other areas of the editorial process through modifications on the edit screen.

The WordPress edit screen can be slow, and overall does not offer a modern publishing experience. This is further challenged by the publishing demands of an enormous media organisation, requiring multiple people to publish content around the clock. One of the most impactful changes we made was to utilise the REST API to help the edit screen load faster, and prevent the lag which is caused by the 'Update' (post) button, resulting in the page refreshing in order to effect an editorial change. There were several reasons we used the REST API technology to achieve this; namely, because we were extensively customising the edit screen and because we wanted the edit screen to feel modern, responsive, and dynamic. The way to achieve this is by enforcing changes directly in the browser to prevent delays, and the most practical way of making

this happen in WordPress is to communicate with the REST API. At Fairfax Media, WordPress is used as an editorial interface only and the system by which reporters' create and file stories. But it is WordPress' capacity to communicate with the wire feed, content, and media APIs, that enabled us to create a flexible and streamlined tool for the modern newsroom. This approach was key to Fairfax. It not only enabled them to centralise their data, providing more consistency and convenience to the newsroom workflow: it also created a clean workspace they could use as a place for data entry, making their admin area much more professional and efficient.

Download our full Fairfax Media white paper to explore how we improved publishing workflows for Australia's leading media brand.

DAMIEN CRONAN
CTO, Fairfax Media

"Human Made were instrumental in developing our Editorial experience."

"Human Made's expertise in WordPress was key to a successful outcome"

06 Why use the WordPress REST API?

From centralising your data and services, to using WordPress as a module in a larger stack, discover the myriad of uses for the WordPress REST API.

Create context-specific solutions

Over 33% of the most popular websites use WordPress. These websites are PHP-based, with frontends built with the WordPress Theme system. **The API frees developers** and allows them to use any technology that will solve a problem in their specific context. WordPress no longer has to be concerned with the frontend: it can just deliver data to any frontend technology. A developer can take data from a WordPress website and display it using the technology of their choice, whether that's for a website, Android application, iOS application or whatever context the data is needed.

Reusable, portable content

The content entered into WordPress is no longer limited to being displayed on a WordPress website. A REST API powered website has content which is infinitely portable. Your content authors only need to enter data in one place. Once it has been authored and published in WordPress, it's now available via the API and can be delivered to websites, web applications, and mobile and desktop applications.

Separation of concerns

In traditional WordPress development, where the frontend and backend are tightly coupled, frontend developers need to be familiar with at least some aspects of WordPress. This makes it difficult to hire and work with purely frontend developers. In a decoupled environment this is no longer a problem. Different teams can work on different project elements while having access to the same data: a backend team can work on WordPress and the database, a team can build the frontend in JavaScript or another web technology, you can have an iOS team and an Android team. Your JavaScript developer no longer needs to learn PHP to work with WordPress, and your WordPress developer no longer needs to tinker with JavaScript. **This widens the pool of development talent** available to work on your website or application, and streamlines project management.

Familiar backend for authors and publishers

One of the reasons WordPress has been so successful is that it provides an easy-to-understand interface for non-technical users. With the REST API, you don't have to decide between using your frontend technology of choice and **giving your authors the admin interface they want**. Authors and editors can work in the WordPress admin and the data is delivered to the frontend by the API. You have the advantage of providing an admin interface which many authors will already be familiar with, reducing the the need for training and re-training, and letting authors quickly start adding content.

Integrate WordPress as one part of a content-authoring workflow

WordPress may only be suitable for one aspect of your website or application. The REST API allows you to use WordPress for just those elements that it is suitable for. The New York Times, for example, uses WordPress for its live coverage platform: data is received via WebSocket from a custom-built service, rendered in React, and served to the frontend via the WordPress REST API. In August 2015, the paper even added Slack to its publishing workflow. **This makes WordPress just one module in a larger stack**, making it more available to the wider web development community for smaller, specific tasks.

WordPress as a central repository

The web is increasingly API-driven, with websites and services aggregating data. The REST API makes it possible for WordPress to be the central place that brings all of this data together. This means that **all of your services and data can be centralised** while providing your authors with a straightforward interface that they are familiar with. This also provides a standard platform for further functionality with WordPress' plugin system.

Develop with live data

When data is exposed through the REST API it can be used by developers in their development environment. Content can be added to the CMS and is available to developers whether they are working on the frontend, the admin, or any applications.

07 Case Study: ustwo

ustwo wanted a decoupled website with a WordPress backend and a frontend built with React.

https://www.ustwo.com/

Why use WordPress and the REST API?

WordPress provides a straightforward interface that the company's global marketing team can use to author content. It's free and open source, so the ustwo design team could focus entirely on the frontend, without wasting time rolling their own CMS.

DANIEL DEMMEL
Full stack web developer, ustwo

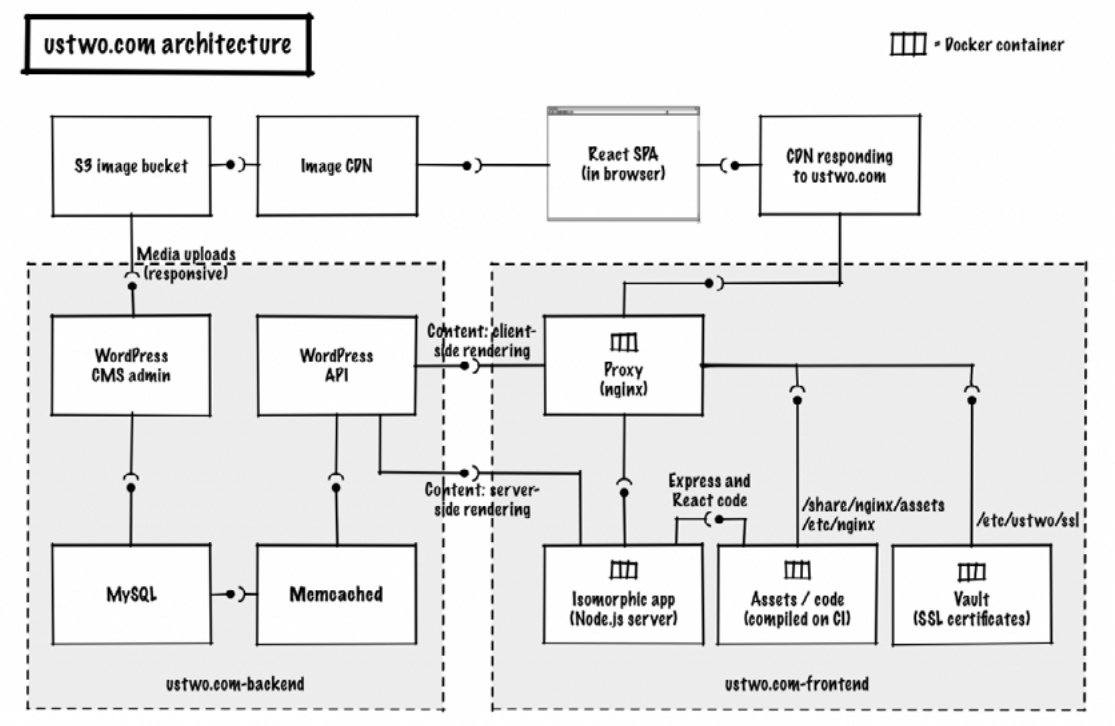
"We chose WordPress as we wanted to have an established open source CMS so that we can be confident that we'll never be left without support or ability to change. To fulfil our design ambitions we decided to build our frontend as a single-page application, which was made possible with the emerging WP-API."

The build

The ustwo website is a single-page application: the frontend is built using React and WordPress manages the content. React was used because it allows for isomorphic rendering (pages can be rendered on the server or by the client). There is a Node.js server that enables server-side rendering.

On the WordPress side, a custom page builder plugin gets authors to enter content in a modular fashion. This ensures that the content is modular and portable to different contexts.

The infrastructure for the REST API is used along with a bespoke API comprising custom endpoints that deliver content in JSON format to the frontend.



Source: uswtwo

08 How the REST API has changed WordPress development

With the REST API, WordPress has evolved from being a web development tool used in isolation to one module available in a web developer’s toolkit.

WordPress as part of a larger stack

WordPress has a familiar and easy to use user interface which authors want to use for managing and publishing content. With the REST API, you can provide this interface to your authors without compromising on the rest of your stack.

Permeation of WordPress outside of PHP communities

As a single module in a larger stack, WordPress can be used outside of its traditional community. The REST API allows developers to create websites and applications in any language without having to roll their own CMS.

New approaches to project management

The separation of concerns that come with a REST API project mean approaching project management in a new way. Developers are able to independently focus on different aspects of the website or application, working with live data retrieved using the API.

The emergence of funnelled, role-based admins

The REST API allows developers to create funnelled administration experiences that **focus on a particular user doing particular actions**. These focused admins will **remove the clutter and empower the user** to do exactly what they need to do.

The enhancement of built-in WordPress functionality

The REST API makes it easier for developers to enhance functionality in the WordPress admin. Developers can create client-side features in the admin that are more advanced and more performant than can be achieved with PHP.

Explorations in non-GPL products

The absolute separation of concerns means that frontend products that retrieve data from the API will not need to be GPL.

However, we have not seen a vast increase in API-powered themes due to the challenges of rebuilding native WordPress functionality on the frontend.

Gutenberg & the REST API: Using modern technologies to power advanced applications

Gutenberg is the first project in WordPress core using the REST API, and it has changed the way developers build on WordPress. For the first time, the REST API is being used to communicate data between the server and a JavaScript powered frontend.

Above: A screenshot from the Gutenberg editor.

RYAN McCUE
REST API co-lead

"Gutenberg's rapid development speed has shown the power of the REST API, and how it has changed the way developers build with WordPress. The rich, block-based data that Gutenberg provides will continue to push the developer experience of WordPress forward, while also providing a fantastic new experience for users."

WordPress and React

The Gutenberg interface is built in the React JavaScript framework (owned and authored by Facebook), the same library used to power Facebook's interface. This is one of many frameworks in use in WordPress, and recently it has been the toolkit of choice for developers building interactivity that works client-side.

In 2017, long running concerns about the patent clauses inside React's license came to a head. Facebook had included a clause stating that should the person or organisation using the source code ever sue Facebook for any kind of patent infringement, the license would be revoked — opening the person or organisation to a counter-lawsuit.

Fortunately, this issue was raised publicly by the Gutenberg team and very quickly the license terms were updated to the MIT license. Today the licensing of React is fully compatible with WordPress' GPL terms, and there are no licensing concerns about its ongoing usage.

09 Challenges presented by the REST API

The introduction of the REST API marks a new era in WordPress development. It's not yet entirely clear how the REST API has changed working with WordPress, but there are already emerging challenges.

Loss of core functionality

A REST API driven website loses frontend features that are linked to the WordPress Theme system, like menu management and post previews. Frontend developers need to take responsibility for re-implementing features that come for free with WordPress. If they are not rebuilt, users must do without them. When writing project specifications for an API-driven project, it will become necessary to be very specific about the features that the client needs and not just assume that because they are in WordPress they are available.

To solve this problem, we anticipate the emergence of REST API base themes that rebuild WordPress features on the frontend. These boilerplate themes will be written in different languages and will provide a starting point for frontend developers to build on.

Disempowers WordPress site builders

In addition to its ease of use, WordPress' strength is that it is easy to set up a website. Through WordPress, many people gain experience of PHP, CSS, and HTML, gaining confidence to make changes to the frontend of their website. The REST API completely decouples the frontend from the backend, disempowering those users, and making the frontend only editable by developers.

For this reason, it is unlikely that we will see a major disruption to the WordPress theme market. Instead, the REST API will be of most significance to large custom builds and WordPress-based SaaS products.

The necessity for structured, portable data

A headless WordPress requires data that can be used across multiple contexts. This means creating and storing it in a way that is completely frontend agnostic. In the first instance, you may just be using data on a website, but you may want to make it available later to a native application. The focus here is on *content management* as opposed to *web publishing*. This **data needs to be**

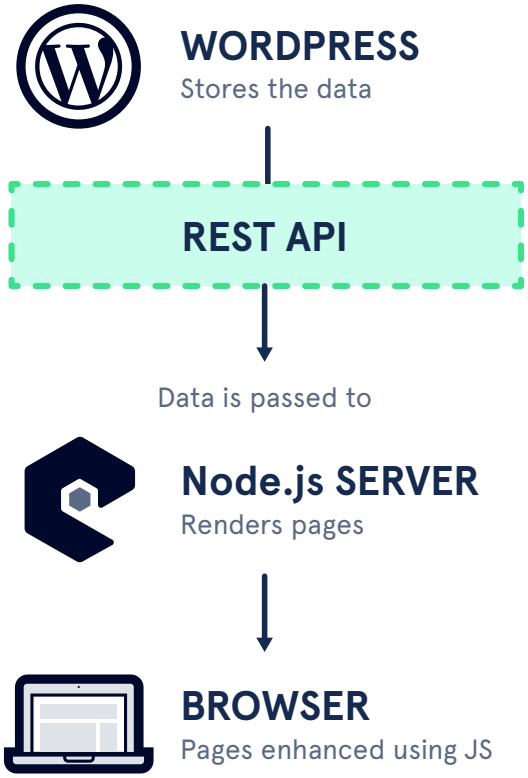
structured in a modular manner, separate to the CSS and HTML. For this reason, REST API-driven sites will not use the WYSIWYG capability in TinyMCE for page layouts, instead using content structured by modular page builders.

WordPress' commitment to backwards (and forwards) compatibility ensures that data produced by the API will be continue to be readable and usable well into the future. This means that you can safely store it knowing that it will continue to be available by a well-supported API. In addition, the WordPress REST API is open, ensuring that your data can be moved out of your site using standard tools.

Dealing with progressive enhancement

In an increasingly JavaScript-driven world, progressive enhancement is a challenge that has to be addressed. Some people have JavaScript disabled in their browser, either because they use assistance technologies, because of personal preference, or because the organisation they work for requires it to be turned off. If content from a REST API driven WordPress website is delivered to a JavaScript-power frontend, these people will simply see a blank page.

Developers need to address these issues to **ensure that the web stays accessible**. One method is to render frontend templates on the server using a technology like Node.js, and then enhance the website on the frontend using client-side JavaScript. This setup, however, requires an additional server, and developers with the experience to implement it.



10 Case Study: NPM

NPM wanted to use the REST API to deliver custom brochure pages and upsell boxes on their website.

https://www.npmjs.com/

Why use WordPress and the REST API?

NPM wanted to use WordPress as a central repository for their documentation and their product pages. Its straightforward interface meant that content authors could easily add data, which is delivered to the client in JSON format.

NICK CAWTHON
Head of Design & UX, npm

"One of things we could not compromise on was the integration with the WordPress REST API. From a security perspective, we felt as if there were only a few agencies that would even have the knowledge to integrate such a thing at the level of traffic we receive. Having been a part of the steering committee, as well as given talks about WPs REST API, we felt that Human Made was well positioned to help lead us."

The build

The NPM website has a WordPress backend and admin. A bespoke API built of custom endpoints serves content in JSON format to the Node.js server. This renders the final HTML and sends it to the browser where Handlebars renders the templates. The API doesn't just send the data: it sends rendered HTML along with scripts and stylesheets. This is cached by Node.js server so that the website stays up even if WordPress is unavailable. It also means that the website stays fast without having to expend effort scaling the database.

Some customisations recreate the post preview feature of WordPress. Parts of the CSS templates and handlebars frontend are used to create a basic WordPress theme. Authors use this to preview posts before they are published and pushed to the frontend.

GLOSSARY

Calypso:
The new WordPress.com interface. Built from the ground up as a single JavaScript application that uses the WordPress.com REST API to communicate to WordPress core.
Page 24

Client-side:
An alternative term for referring to the user-facing interface.
Page 7, 31, 33, 36

GPL (General Public Licence):
A common licence used for open source software.
Page 32, 33

Hypermedia:
Diverse types of interlinked, nonlinearly accessed media forms, typically used in the context of developing web based APIs.
Page 15

JavaScript:
A web programming language typically used to create interactive web pages.
Page 7, 24, 25, 32, 36

JSON (JavaScript Object Notation):
A format for storing data in a human-readable format.
Page 4, 14, 29, 38

MySQL:
An open source database management system.
Page 6

Node.js:
An open source server environment.
Page 7, 29, 36, 38

OAuth (Open Authorisation):
An open standard for authorisation on the internet, and a process whereby users can log in to third party websites without exposing their passwords.
Page 18, 24

PHP (Hypertext Processor):
An open source server-side programming language.
Page 6, 7, 11, 25, 26, 32, 35

React:
A JavaScript library for building user interfaces.
Page 11, 26, 29, 33

continues -->

React Native:
A framework for building mobile applications using JavaScript and React.
Page 7

Redux:
An open source extensible options framework for WordPress themes and plugins.
Page 11

Repository:
A central location where all associated code is stored.
Page 27, 38

Repress:
A library which takes control of part of your Redux state and handles talking to the WordPress REST API.
Page 11

SSL (Secure Sockets Layer):
The standard security technology for establishing an encrypted link between a web server and a browser.
Page 18, 24

Vue.js:
An open source JavaScript framework for building user interfaces.
Page 7

Links

The WordPress REST API:

- Official website and documentation
- WordPress core discussion about the REST API

Client Libraries:

- Node.js
- Backbone.js
- AngularJS
- PHP Client

Authentication:

- OAuth 2.0
- Basic Authentication

Tools:

- WP-CLI client
- API Console
- WP JSON API Connect
- API client UI
- Restplain
- Repress

Other resources:

- Picard React theme
- ustwo.com frontend

A Day of REST — The WordPress REST API Conference

*Above: **K. Adam White** presenting on the REST API at ADoR 2017 in Boston. K. Adam is a component maintainer for the WordPress REST API "wpapi" package on npm.*

In 2016 and 2017, we held two conferences on the REST API in London and Boston, respectively. With speakers from The New York Times, Human Made, Wired, Automattic, and Bocoup, including co-leads of the REST API project, we explored how people are using the REST API in their projects.

Watch the full event videos here:

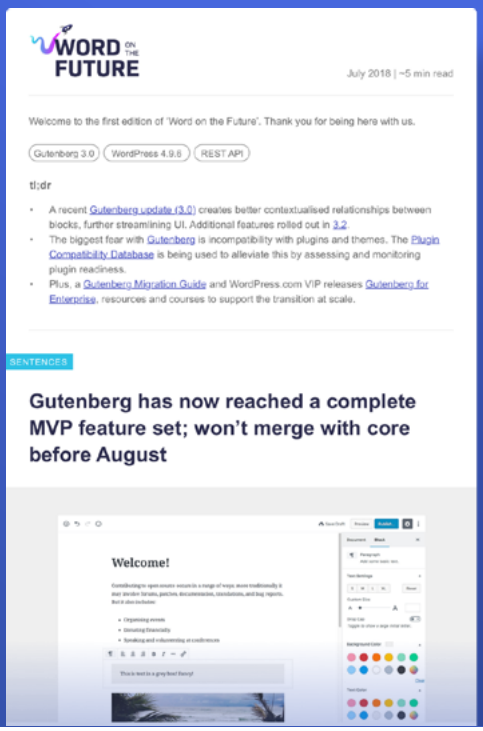
A Day of REST London, 2016

A Day of REST Boston, 2017

THE INDUSTRY NEWSLETTER FOR WORDPRESS

by Human Made

Subscribe to get the latest



Word on the Future — The Inside Track on Enterprise WordPress

Stay informed about the future of digital experiences with our Enterprise Newsletter, 'Word on the Future'— covering the inside track on Enterprise WordPress news with curated opinions and insights from those at the heart of the industry.

Subscribe at altis-dxp.com/newsletter to get:

- ✓ Curated opinions and insights on the latest WordPress news from some of the most respected professionals in the industry.
- ✓ Cut-to-the-chase stories you can consume in under 10 mins, direct to your inbox every month. Early access to white papers and documentation published by Human Made.
- ✓ Information to help you lead the future of your business, sourced from across the ecosystem with summaries highlighting news relevant to you.

Above: ***The Human Made team*** *during a company retreat in Sri Lanka, in 2019.*

Human Made is an award-winning, global WordPress agency and leading providers of digital experience platforms for enterprise.

Altis is our next-generation digital experience platform: backed by 10+ years of globally recognised engineering excellence building solutions for enterprises and leading brands. With deep roots in open source, Altis is a proven end-to-end enterprise-augmented WordPress platform.

Talk to us today. Discover Altis.

sales@altis-dxp.com

Ant Miller, Commercial Director
ant@altis-dxp.com

DIRECTOR(S) OF CLIENT SERVICES:

Americas

Sam Sidler
samuel.sidler@altis-dxp.com

Lonnie Tapia
lonnie@altis-dxp.com

Europe, Middle East, and Africa

John Bevan
john.bevan@altis-dxp.com

Ruxandra Gradina
ruxandra@altis-dxp.com

Asia Pacific

Jon Ang
jon.ang@altis-dxp.com

Shinichi Nishikawa
shin@altis-dxp.com

altis

Thanks for reading!