elasticpath

GraphQL & Cortex

What's the difference and why it matters in today's **API driven world**

Product Overview

Introduction

APIs are redefining everything we do in the digital world. Plugging into APIs helps companies focus on their core business while not having to build out every function on their own. APIs offer a freedom for innovation where by technology fits your business versus having to reengineer your entire business to fit the technology. All of this is key to the kind of flexibility that is essential for success.

In the commerce world, APIs have changed how commerce and customer engagement ecosystems are architected. Businesses are rapidly abandoning the monolithic all-in-one commerce platform in favor of a best-of-breed or headless commerce approach where many systems and applications work together via APIs to deliver seamless customer experiences. This gives businesses the agility to create experiences that keep pace with rapidly changing customer preferences, markets and emerging touchpoints by adding new elements to the ecosystem without having to rip and replace their entire commerce infrastructure.

So, it goes without saying that a commerce platform's API approach, coverage, quality of service, and performance are critical factors to consider when choosing a commerce platform.

The Commerce API Economy

REST has become the standard for companies deploying APIs and launching developer platforms. The beauty of REST is that a developer working with someone else's API doesn't need any special initialization or libraries as all requests follow the same HTTP standard. REST also inherits the scalability and performance HTTP provides, making it a great option for businesses of all sizes.

However, there are many variants of REST and some APIs that label themselves as REST do not follow all the REST constraints, which creates challenges for developers. For instance, poorly designed REST APIs require the developer to know exactly what information they want to retrieve and where that information resides. This requires them to spend more time investigating and navigating multiple API calls and sometimes over-fetching data. Over-fetching is when the client downloads more information than is actually required in the application. Poorly designed REST APIs may also introduce business logic to front-end applications, resulting in degraded system performance, more complex front-end code, and customer facing applications that are susceptible to breakage as API resources are updated.

Fortunately, there are ways to mitigate these problems, but they all point back to the API developer, their understanding of how the system they are building will be used and the intricacies of the API architectural style they choose. Let's take a closer look at those API architectural styles.

GraphQL

GraphQL is an API specification developed by Facebook in 2012 to address iOS and Android application performance and reliability problems created by the complexity of Facebook mobile apps. When GraphQL was developed, mobile bandwidth was a big issue which meant fetching discrete bits versus large blocks of data had measurable performance benefits.

It is similar to REST in that it is used to fetch data from any backend service, but how you fetch the data is different. The primary objectives of GraphQL are to minimize the number of API calls to data stores and to fetch just the right amount of data. Using GraphQL, developers choose what fields need to be included in the response and make a single API call to fetch the subsequent data, effectively solving the problem of over-fetching that poorly designed REST APIs can exhibit. GraphQL is also version-less and discoverable, making it easier for developers to use and code easier to maintain than REST Level 1 and 2 APIs.

Although GraphQL does address some of the challenges associated with poorly designed REST APIs, it has limitations that need to be understood. With GraphQL, developers hard code API calls into the client, resulting in applications that are tightly coupled to the server and require constant care and maintenance. Like REST, GraphQL is a specification and a particular API might be poorly designed and implemented. For instance, if GraphQL support is added as an afterthought to existing APIs, the resulting implementation is likely to be slow and inherit any problems associated with the original API design.

Designing and implementing an easy to use and performant API is difficult, and GraphQL is no exception. GraphQL has great potential to improve the developer experience and access to commerce data and services, but it is still relatively new - everyone is still figuring it out and the quality of implementations across providers varies.

Hypermedia API and Cortex

Hypermedia, or REST Level 3 APIs solve the problems most developers have experienced with poorly constructed REST APIs. You may not be familiar with hypermedia principles, and with good reason, there aren't many available because they take a lot of work to get right.
The reward for those that take the time to understand hypermedia and adhere to its constraints is a system with unmatched flexibility and complete decoupling of client and server so system components can evolve independently.

Elastic Path Commerce Cortex is a hypermedia compliant headless commerce API that is fully discoverable and serves as an API orchestration layer. Cortex manages dependencies, aggregates resources, eliminates over-fetching, and reduces the number of API calls to access commerce business services and data. For example, rather than an application having to make four separate API calls to fetch a product's name, description, price and associated promotions, as might happen with a poorly designed REST API, Cortex can simply retrieve all these data points at once with a single API call. Cortex understands event sequence and user state, eliminating the need for business logic to be embedded in the front-end application, so experiences are faster to create, and front-end code is easier to maintain.

Additionally, Cortex is version-less, meaning Elastic Path Commerce services can be updated without impacting front-end applications. Developers can extend Cortex to orchestrate resources that exist outside Elastic Path Commerce, so all resources are accessed in a consistent way. Consider the situation where a business needs to connect with a third-party credit bureau to verify customer credit worthiness. Cortex can be extended so the business can make a credit check a required step in the checkout flow. Cortex will enforce the process natively as a next required action, so any device whether it's a store POS, a website or a chatbot will be required to go through the same purchase flow.

With Cortex, developers have all the benefits of hypermedia because Elastic Path has done the hard work upfront and has proven the API can perform at scale.

What's the Difference Between GraphQL and Cortex Hypermedia?

Both offer APIs that are version-less, stateless, and discoverable, but there are important distinctions between them. GraphQL describes to front-end applications what actions can be taken, but it does not describe when those actions can be performed. Depending on the scenario, this may require front-end applications to include business logic to protect against situations where a proper sequence of events is important, like in a service sign-up, scheduling, configuration or checkout process. Thus, business logic makes its way back into the front-end application where it doesn't belong, eroding the value of a headless commerce platform. This becomes especially important when multiple front-ends are in play. Without Cortex, the result is a duplication of business logic across all front-end applications.

As a query language, GraphQL is easy to understand and use. In the context of a commerce ecosystem, it's great for fetching data to build views for a variety of client experiences and optimizing the performance of that data retrieval. On the other hand, Cortex Hypermedia supports workflow orchestration and is a more complete approach to expose data and services, scale for businesses of all sizes, and enable a complete separation of front-end applications and back-end commerce resources.

Elastic Path Commerce

Elastic Path offers the leading purpose-built headless commerce platform to unify experiences across the entire enterprise. As the pioneer of headless commerce, Elastic Path empowers you to sell products and services in the connected world through the web and a touch of a finger or a spoken command. Its products, including Elastic Path Commerce Cloud, deliver commerce freedom and accelerate the creation of any customer experience on a single platform. Through collaborative innovation between Elastic Path's team, customers and partners, Elastic Path leads the way in revolutionizing commerce.

Elastic Path is based in Vancouver, Canada, with offices in the U.K. and U.S. Learn more at elasticpath.com.