Why Developers Need APIs

By G. Andrew Duthie, Chief Consultant of Devhammer Enterprises LLC

If you've been paying attention over the last couple of years, you've probably noticed a lot of talk about APIs, specifically about Web APIs, sometimes referred to as REST or RESTful APIs. But if you're like many developers, you may still be wondering whether they matter to you.

Why Web APIs

Web APIs represent a great opportunity to use familiar and well-understood technologies to more easily create connected software. It's really that simple.

APIs are hardly a new idea. Whether SOAP-based ASMX web services, WCF services, or similar Java-based solutions, there have been many attempts to solve the problem of connecting systems in a way that is easy, consumable across different technology stacks, and secure. ASMX web services solved the first problem well, but cross-platform support wasn't straightforward. WCF did a great job with security, but at the expense of complexity.

The key to the success of modern Web APIs is that they leverage existing HTTP standards, rather than trying to reinvent the wheel. This, along with the adoption of JSON as a more lightweight data transport format (though many services support XML for compatibility), makes modern Web APIs easy to create, debug, and secure.

Why Web APIs Matter

For developers whose clients or companies aren't building or consuming APIs, it may be easy to dismiss them as another shiny object. I think that's a mistake. While it's certainly possible that Web APIs are a passing fad, I think they offer enough advantages that it's worth it for most developers to spend some time learning about them, and trying their hand at building and/or consuming these services. Here's why:

1. In an increasingly cloud-connected and API-driven world, understanding how to create and consume APIs will differentiate you from the crowd, and open additional career and advancement opportunities.

2. APIs can be helpful in monetizing data or services you (or your company) may own.

3. APIs can provide a nice layer of abstraction for your data and business logic, making it easier to use the same data and business logic with web clients as well as iOS, Android, etc. apps, as well as providing enhanced testability.

Just because you're not working for some hot startup doesn't mean you can ignore this trend. After all, you don't want to be caught looking like a deer in the headlights when your boss says it's time to build an API for your app.

Getting Started with Web APIs

While the underlying infrastructure that supports Web APIs are all about HTTP and related standards, how you get started will also vary somewhat based on the development stack you've chosen. Personally, I've spent the most time with Microsoft's ASP.NET stack, so that's where I'll focus. If you've spent time with the ASP.NET stack, you're probably aware that there have been a number of ASP.NET and related technologies designed to support web services, from the SOAP-based ASMX services that shipped with the very first version of ASP.NET, to the WCF-based ADO.NET Data Services (also known as WCF Data Services), to today's ASP.NET Web API. While it's still possible to use any of these to build services, some of them, including WCF Data Services, have been deprecated. For a clean, modern Web API, I believe ASP.NET Web API is the place to start today. ASP.NET Web API follows the architectural style of ASP.NET MVC, in that it separates out the application logic into API controllers, which are responsible for processing requests and (if appropriate) returning data, and models, which define the structure of the data, and can be either so-called POCOs (plain old CLR objects) or from an ORM such as Entity Framework. A routing engine is responsible for looking at incoming URLs and determining which requests need to be routed to which controllers. Unlike an MVC-based web site, there are no views. This architectural style provides clean separation of responsibilities, testability, and makes it easy to customize the behavior of your APIs.

Get started with ASP.NET Web API by visiting http://www.asp.net/web-api, where you'll find tutorials, videos and more.

REST and Hypermedia

Once you've mastered the basics of building Web APIs, you may want to delve further into building fully RESTful APIs, which includes the concept of hypermedia. Hypermedia-driven Web APIs take things a step beyond just using HTTP verbs and JSON, and also provide hypermedia links that allow clients to dynamically discover how they can interact with your API. By including hypermedia links, hypermedia-aware clients can automatically adapt to changes in your API, because they aren't relying on static links, but rather are interpreting the hypermedia links embedded in the responses your API returns.

The hypermedia aspect of Web API development is still very much in flux, with a number of competing ways to implement hypermedia support, including JSON-LD, HAL, Collection+JSON, and Siren, to name a few. A good overview of these options can be found at http://j.mp/apihypermedia.

The good news for those using ASP.NET Web API is that, through support for OData query options, Web API projects can easily add basic hypermedia support. For CRUD (Create, Read, Update, Delete) style APIs, this is a great way to get a strongly RESTful service up and running quickly. The latest available tutorial for OData in Web API as of this writing can be found at http://j.mp/webapiodatav4.

Jump In

Whether you go full-on REST, or just build some HTTP-based Web APIs, the key takeaway is to get started. If your current employer isn't ready for you to build APIs, look for a side project where you can build something real-world. Perhaps a stats API for your child's sports league, or something to help your church share their information. Building Web APIs is a skill you will find valuable and marketable, and the best time to start is right now.

About G. Andrew Duthie

G. Andrew Duthie is the founder and chief consultant for Devhammer Enterprises LLC (http://devhammer.net). Andrew is the author of several books on ASP.NET, and has spoken frequently on Web development at conferences such as VSLive!, as well as at numerous user groups and code camps throughout the eastern US. Andrew spent 10 years as a technical evangelist at Microsoft, and is now focusing on bringing great software solutions to businesses through his consulting work. You can reach Andrew via his blog at http://devhammer.net/contact.