# Foundations of Popfly

## Rapid Mashup Development

Eric Griffin

Apress®

**Foundations of Popfly: Rapid Mashup Development**

**Copyright © 2008 by Eric Griffin**

ISBN-13 (pbk): 978-1-59059-951-8

ISBN-10 (pbk): 1-59059-951-9

ISBN-13 (electronic): 978-1-4302-0568-5

ISBN-10 (electronic): 1-4302-0568-7

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editor: Ben Renow-Clarke
Technical Reviewer: Sarje Page
Editorial Board: Steve Anglin, Ewan Buckingham, Tony Campbell, Gary Cornell, Jonathan Gennick, Kevin Goff, Jonathan Hassell, Matthew Moodie, Joseph Ottinger, Jeffrey Pepper, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Project Manager: Kylie Johnston
Copy Editor: Heather Lang
Associate Production Director: Kari Brooks-Copony
Production Editor: Ellie Fountain
Compositor: Susan Glinert Stevens
Proofreader: Nancy Sixsmith
Indexer: Carol Burbo
Artist: Kinetic Publishing Services, LLC
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `http://www.springeronline.com`.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail `info@apress.com`, or visit `http://www.apress.com`.

The source code for this book is available to readers at `http://www.apress.com`.

*The book is dedicated to my wife, Susan, who is my source of unending support, love, and understanding.*

# Contents at a Glance

# Contents

# About the Author

**ERIC GRIFFIN** works as a Microsoft consultant. He is based in Atlanta and specializes in Microsoft Application Development technologies, tools, and platforms. This includes Visual Studio, SQL Server, ASP.NET, C#, Reporting Services, Office, SharePoint Server, and more.

# About the Technical Reviewer

**SARJE PAGE** is a consultant with Qualesco Consulting Group, where he specializes in Microsoft technologies and relational database management systems. In the last ten years, he has designed, deployed, and optimized many data-oriented software applications while working as an information technology consultant for leading companies in the consumer products, construction, and insurance industries.

# Acknowledgments

I would like to thank my technical reviewer, Sarje Page, for his comments during the writing of this book.

# Introduction

When I heard that Microsoft was developing a mashup creation tool, I knew two things: one, that I wanted to get access to it as soon as possible, and two, that I wanted to write a book about it.

Mashups are all the rage. The explosion of public APIs by the who's who in the Web 2.0 world (Google, Yahoo, Microsoft, and so on) has caused a revolution in the way software is developed—mashups are the precursor to the way software will be developed in the future. Software as a service has long been on the horizon, and Web Services, in its many technological forms, is the enabler of cross-platform, cross-service integration that is at the heart of mashups.

But even as mashups emerged, the tools used to create them were familiar only to Web-savvy programmers and enthusiasts. For the larger, nontechnical audience, the tools were unapproachable and difficult to use.

Then Microsoft introduced Popfly—a tool for the nonprogrammer enthusiast who wants to quickly create new software.

## Who This Book Is For

This book is for the nonprogrammer or enthusiast who wants to create new software fast. The tool enables users to take advantage of the heavy lifting already done by other programmers and the functionality supplied by public APIs in the form of Web Services from service providers like Microsoft, Yahoo, Google, and more.

## How This Book Is Structured

This book consists of ten chapters. The first six chapters show you how to use the Popfly environment to create mashups, and the last four teach you to create components, called blocks, that can be used to create mashups within Popfly.

### Chapter 1

Because mashups use technologies that are familiar to Web-savvy programmers in new and exciting ways, they are popping up everywhere—inspiring seasoned programmers and amateurs like. In this chapter, I introduce you to mashups and the rich world of creativity and freedom they offer. Mashups allow you to mix and match competing vendor APIs to create new, fun, and sometimes strange products and services.

## Chapter 2

In this chapter, you see how Popfly hides the complex technologies needed to create mashups through the use of its simple tools, which require no coding. It also has an online community that fosters sharing, copying, and feedback.

## Chapter 3

In Chapter 3, you create your first Popfly mashup. The mashup retrieves an RSS feed from a blog and displays it in a News Reader using blocks that you configure and connect in Popfly's block designer. You also add HTML to your mashup page.

## Chapter 4

In this chapter, you create two more mashups: one using the Flickr photo service and PhotoSphere blocks and another using the Upcoming service (a Yahoo web site that provides dates of entertainment events) and Virtual Earth. You learn how to generate and manage API keys and how add custom code to modify a block. You also learn to retrieve information from the user and pass it on to blocks with the User Input block.

## Chapter 5

In this chapter, you create your own web page using Popfly's Page Editor. You learn how to customize page styles and layouts and how to insert shared mashups onto web pages for the world to see.

## Chapter 6

This chapter explains the four ways that Popfly supplies for you to share your mashups with external users: web pages, Windows Vista gadgets, Windows Live Spaces, and Facebook. You also learn how to e-mail your mashup.

## Chapter 7

This chapter teaches you how blocks are defined (with XML) and executed (with JavaScript). You examine the RSS and News Reader blocks from previous examples to see how the block definitions and code are used together. You will learn about the Popfly Runtime Environment (PRE) and some of the helper functions it provides to make retrieving data from external sources easier.

### Chapter 8

In this chapter, you examine the Popfly Block SDK. We'll look at the Popfly SDK test harness, which was created with .NET technology, and how to run it in a free tool from Microsoft called Visual Web Developer.

### Chapter 9

In this chapter, you learn to use the rich functionality within Microsoft's Visual Web Developer to create blocks using the Popfly Block SDK. I'll explain how to amend a block's JavaScript file and block definition files in its editor. You also learn about Visual Web Developer's debugging capabilities and how to use the Block SDK schema files to help write and validate block definition files.

### Chapter 10

This chapter teaches you to use Popfly's Block Creator to create or add your block to the Popfly environment for use in mashups. The Block Creator is not as rich as an integrated development environment like Visual Web Developer, but it has some basic code completion functionality to help you write your block code. You will also learn how to copy, or rip, code from other blocks to help you learn how it works or to give you a head start on your own blocks.

## Prerequisites

Some knowledge of JavaScript and the technologies surrounding it, like AJAX and JSON, would be very helpful, as would knowledge of XML. No experience with Popfly is needed, and experience with software development kits (SDKs) and integrated development environments (IDEs) is not required but will be helpful.

## Downloading the Code

You can find the Popfly Technorati Block created as a sample in this block from `www.apress.com` in the Downloads section of this book's home page. Please feel free to visit the Apress web site and download all the code there. You can also check for errata and find related titles from Apress.

## Contacting the Author

You can reach Eric Griffin at his personal e-mail address at `ebgriffin1968@hotmail.com`.

CHAPTER 1

# Introduction to Mashups

Mashups are inspiring a new generation of technology enthusiasts and programmers. Mashups enable experienced, web savvy programmers to integrate with the giants of the Web 2.0 space. As the name implies, mashups mix and "mash" the programming interfaces from different companies' products and services to create new products and services. Yahoo, Google, Amazon, eBay, and Microsoft have published *application programming interfaces* (APIs) based on web standards that allow you to utilize their complicated functionality without being a programming expert. Dozens more companies, big and small, have followed in the same way, creating a mashup explosion of API mixing and matching. New, sometimes strange, mashup creations pop up all the time.

In this chapter, I will explain what mashups are and give you a brief history of how mashups evolved and what technologies are used to create mashups. I will also give you a small sample of the dozens of products, services, and resources that are available to make mashups.

## What Is a Mashup?

A mashup is the evolution of the way web applications are made: it allows a programmer to integrate products and services from competing companies like Microsoft, Google, Amazon, and Yahoo to create new, unique products and services, as illustrated in Figure 1-1.

These new products and services integrate APIs published by each company using web technologies that have evolved over the history of the web applications. We will look at these technologies in more detail later on in the chapter.

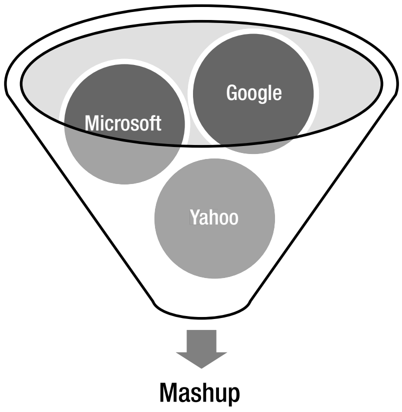

**Figure 1-1.** *Mashups can be created using APIs from competing companies.*

## A Brief History of Mashups

It will be difficult to detail the history of mashups without understanding the broader context of the history of the Web and how it has spurred the emergence of mashups.

Have you heard of Web 2.0? If you haven't, don't worry; this will be the first of many encounters with this phrase. Web 2.0 was coined in 2001 by Tim O'Reilly after the dot com crash. From a technical perspective, the word "Web" refers to the products, services, and business models that are created using the Internet as a platform. This is contrary to the PC or desktop computers being used as a platform. The "2.0" implies an upgrade from 1.0 products, services, and business models and the previous generation of technologies used to create those products and services. The new generation of technologies in Web 2.0 make web sites function and respond dynamically, like desktop applications.

Web 1.0 companies built products and services that would lock their customers in. They accomplished this by controlling the customers' data.

**Note** When I talk about customer data, I am referring to anything that the customer perceives as valuable. It doesn't have to be important to another customer. It is personal data but not necessarily sensitive like a Social Security number or address. And customers don't have to own the data. Examples of customers' data could be the weather forecasts in their city, a query on a search engine to find a consumer review web site because they want to buy a camera, a gallery of photos they want to share with family, or a subscription to a financial feed that tracks their favorite stocks. All of the examples show how data can be important to an individual, and that is the key to its value.

Let's look at the example of portals offered by vendors Netscape and Yahoo back in 1997. Customers logged in (using a vender-specific ID, of course) and customized his portals to access news, weather, and sports to their liking. It saved this information for them and updated it regularly. However, the news, sports, and weather were from sources offered only by that portal alone or by the portal vendor (e.g., you only had access to Yahoo News, Yahoo Sports, and so on). And it was likely the source of data had to pay to be available to the customer, and it might be available exclusively. In this model, by controlling the data, the portal vendors would create barriers to entry by centralizing and controlling data that was valuable to the customer. To get to data, customers had to go to that vendor's products and services.

In addition to being controlled, data was isolated without the ability to be integrated with other data from other sources. There was no easy way for the customer to tie two different bits of logically groupable data in a meaningful way. For example, there was no way to merge information about an event at a local park with the weather forecast or a directory of Italian restaurants with a map of your area.

Web 1.0 API technologies were proprietary and often built with standards that were PC based and not web friendly—plug-ins, C interfaces and dynamic link libraries (DLLs). These APIs, frequently packaged in software development kits (SDKs), were about getting the programmer immersed and invested in mastering the vendor's APIs. There were developer programs and conferences touting one vendor API over another.

Web 2.0 represents how business models built on the Internet evolved from Web 1.0. After the dot com bomb in 2001, a small wave of companies emerged with a different perspective on how to leverage the Internet. That leverage came from opening up the customer data their products and services controlled. Using web standards that were common and widely adopted across traditional competitors, a new value proposition was created. This value proposition supercharged start-ups like Google, Amazon, and eBay and revitalized established Internet players like Microsoft and Yahoo.

Portals in Web 2.0 put customers in charge of their data. Let's return to the example of portals and now move forward to 2008: myYahoo, Microsoft Live, and iGoogle not only enable the customer to completely control what data they see and what source it comes from, but where on the page and how often it is displayed. Each bit of data, contained in what is called a portlet, is customizable to further give customers control over their data. If you like Google's portal but love Yahoo Sports and MSNBC News, you can get access to them inside Google's portal.

Creating new content for modern portals is based on open standards. Gone are the proprietary SDKs that are bound to vendor platform products and services. Now, a programmer can leverage the same fundamental technical knowledge to develop for Google, Yahoo, or Microsoft, as all the APIs use the same technologies, like HTML, CSS, and JavaScript.

Web 2.0 technologies play a big part in the advancement of portals. The fact is that modern portals are mashups. Mashups enable the retrieval and control of data by using the open APIs provided by service providers. In the next section, we will detail the technologies important to mashup development.

## Understanding Mashup Technology

As with all technologies, mashup technologies evolved over time. Figure 1-2 shows the progression of technologies that emerged and serve as a foundation for mashup development.

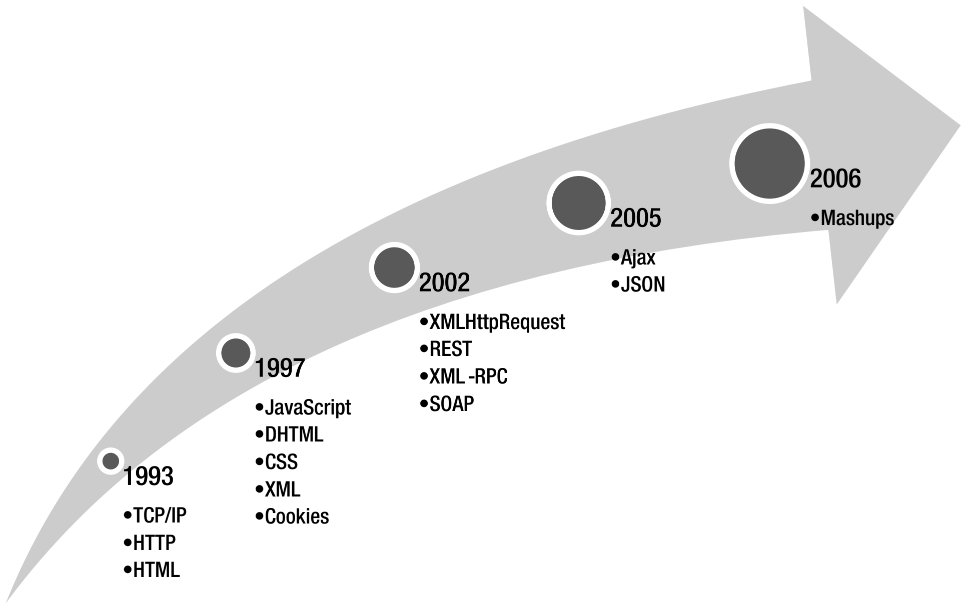

**Figure 1-2.** *Mashup technology timeline*

The most important thing to remember is not when a particular technology was created but when it became relevant to the overall community. One key factor is wide vendor adoption.

JavaScript would not be as important today if it had not been adopted by Microsoft and Netscape in their browsers. Netscape created JavaScript (originally called LiveScript) in 1995 in its browser Netscape Navigator. It wasn't until late 1996 that Microsoft shipped JScript (its own implementation of JavaScript) in its browser Internet Explorer. JavaScript became a de facto standard because it was used by virtually 100 percent of the market's browsers. Any newcomers in the browser arena had to implement JavaScript to be considered a viable alternative browser.

JSON was introduced as a JavaScript-based data format in 1999. It took over seven years to become relevant to the web development community. But as the web development community realized the need for a simple lightweight way to transfer data besides XML, JSON was rediscovered.

Before each of these technologies is explained, it is important to further categorize them into the roles they play in mashup development, which is done in Figure 1-3.

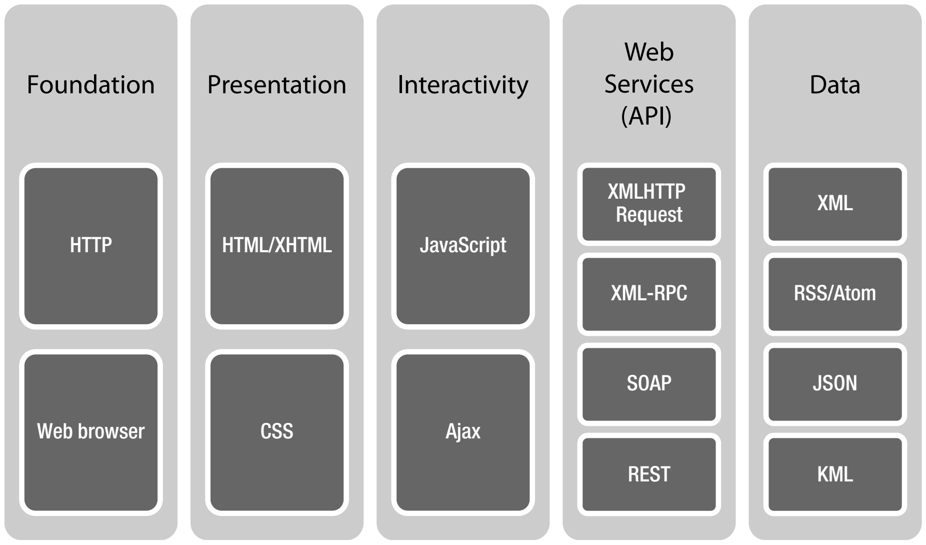

**Figure 1-3.** *Categorized technologies used by mashups*

## Foundation Technologies

You can't build a house without a foundation. The same statement holds true for creating mashups. There are two technologies that provide the foundation for all the others: HTTP and the web browser. There are dozens of books written on each of these technologies, and they are fairly well known in general, so I won't go into detail now. But it is always important to know how things work and interrelate, so I will give a broad overview here.

### HTTP

Hypertext Transfer Protocol (HTTP) is the protocol that enables us to navigate the Web. HTTP is a request/response protocol among clients and servers. An HTTP client initiates a request by establishing a TCP connection to a particular port on a remote host. An HTTP server listening on that port waits for the client to send it a request for a web page and then sends the client the web page it requests, provided the client has the relevant permissions to access that page.

### Web Browser

The delivery platform for mashups and all other web-based applications and sites is the web browser. A web browser is a software application that enables a user to display and interact with text, images, and other information. It hosts the technologies within the presentation, interactivity, web services, and data layers. The web browser uses HTTP to request web pages and data from remote servers. There are many browsers available today: Internet Explorer, Firefox, Opera, Safari, Camino (specifically on Mac OS), and Konqueror (specifically on Linux); there are even text-only browsers such as Lynx.

## Presentation Technologies

These technologies render the user interfaces that you see when you view web pages.

### HTML/XHTML

Hypertext Markup Language (HTML) is the language for the creation of web page structures. It describes the structure of text-based information in a document by denoting certain text as headings, paragraphs, and lists. It also denotes interactive forms, embedded images, and other objects. HTML is written in the form of elements called *tags*—labels surrounded by less-than (<) and greater-than signs (>).

XHTML is a reformulation of HTML in XML, therefore XHTML documents have to follow the strict rules of XML (they have to be well-formed, meaning elements need to be properly closed, element attributes need to have quotation marks around their values, etc.). It also provides new tags that have made structuring web pages easier. You can find a great overview of XML at XML.com (`http://www.xml.com/pub/a/98/10/guide0.html`).

### CSS

Cascading Style Sheets (CSS) is a language used to describe the presentation of a document written in a markup language. It is used to style web pages written in HTML and XHTML by defining rules that specify how markup should be styled and positioned.

## Interactivity Technologies

The interactive technologies are used to create custom dynamic behavior like showing or hiding content, dragging and dropping of content, and animations. Without interactivity in web pages, you are left with bland, static uninspired functionality that doesn't provide the same robust interaction that a desktop application provides.

### JavaScript

JavaScript is a scripting language used in web browsers to provide interactivity to web pages. It is compliant with a script standard called ECMAScript. JavaScript is the Mozilla Foundation's (originally created by Netscape Communications Corporation) implementation of the ECMAScript standard.

JavaScript was influenced by many languages and was designed to have a similar look to Java but be easier for nonprogrammers to work with. Contrary to its name, JavaScript is unrelated to the Java programming language. The language was renamed from LiveScript in a comarketing deal between Netscape and Sun in exchange for Netscape bundling Sun's Java runtime with its browser, which was dominant at the time.

I could spend the entire book focusing on JavaScript. Knowledge of JavaScript is not required to create basic mashups with Popfly, but it is required when you want to extend

Popfly's functionality with Popfly Blocks. There are dozens of books that can give you a deeper insight into it. I have listed some here:

- *Beginning JavaScript with DOM Scripting and Ajax: From Novice to Professional* by Christian Heilmann (Apress, 2006)
- *Practical JavaScript, DOM Scripting, and Ajax Projects* by Frank Zammetti (Apress, 2007)
- *Pro JavaScript Techniques* by John Resig (Apress, 2006)

### Ajax

Ajax is a JavaScript development technique that is so important to Web 2.0 development that it should be considered a separate technology. The name is an acronym standing for Asynchronous JavaScript and XML. It is a culmination of the trend in developing web applications that respond like desktop applications.

Ajax is asynchronous in that loading does not interfere with normal page loading and does not require that the entire web page be reloaded each time the user requests a change. For example, if you update the quantities in a shopping cart, the site could use Ajax to instantly display the new price and shipping total without having to waste time reloading the entire page again. Ajax increases a web page's interactivity, speed, functionality, and usability. JavaScript is the programming language Ajax function calls are made in. Data retrieved using the technique is commonly formatted using XML and JSON.

## Web Service Technologies: Application Programming Interfaces

When vendors want to create APIs for their products and services on the Web, web services are the means to do it. There are many options for vendors to use. Mashups use web service technologies to access a vendor's product functionality.

### XMLHttpRequest

Before we go into the various types of web services, it is important to call out the most important component of the emergence of web services as a way to create web APIs—XMLHttpRequest.

XMLHttpRequest is not a web service technology but an API that is available in JavaScript, to send data to and from a web server using HTTP, by establishing an independent communication channel between a web page's client side and server side.

The data returned from XMLHttpRequest calls are often provided by back-end databases. Besides XML, XMLHttpRequest can be used to fetch data in other formats such as HTML, JSON, or plain text.