shortcut

Addison-Wesley Professional Ruby Series

Mongrel

Serving, Deploying, and Extending Your Ruby Applications

Matt Pelletier
Zed Shaw

Addison-Wesley
Pearson Education

www.awprofessional.com/ruby

Visit us on the Web: www.awprofessional.com

ISBN 0-321-48350-2

First release, October 2006

To my father Mike,
who got me started many years ago.
—M.P.

To Mike and Roxanne,
for helping a poor boy when he needed it most.
—Z.S.

SECTION 1

What This Short Cut Covers

This short cut is an introduction and guide to Mongrel, a fast, versatile Ruby Web server. If you build or manage Web applications, this will be a useful reference as you set up and use Mongrel in your development and production environments, as well as a handbook for how you can extend Mongrel to suit your own needs.

In addition to covering how to use and extend Mongrel, we also review a number of topics that we consider "Best Practices" for modern software development, deployment, and performance testing. We discuss these in the context of using Mongrel, but they should be considered applicable to any software project. Our own work has always benefited from seeing the techniques of others, so we hope that sharing the experience, approach, and philosophy that went into designing and developing Mongrel will be interesting and helpful for your own pursuits.

Section 1:
What This Short Cut Covers

1.1 The Format of This Short Cut

This short cut uses the common formatting conventions to represent code, filenames, and so on. If you have never read a book on software before, the following description will prove essential.

Indented blocks of code or shell commands will appear in a fixed-width font:

```
uri '/', :handler => DirHandler.new('/var/www/opinions/')
```

A reference to a file, function, or class within a sentence—like *HttpHandler*, *mongrel.log,* or *some_function(param)*—will be in italics.

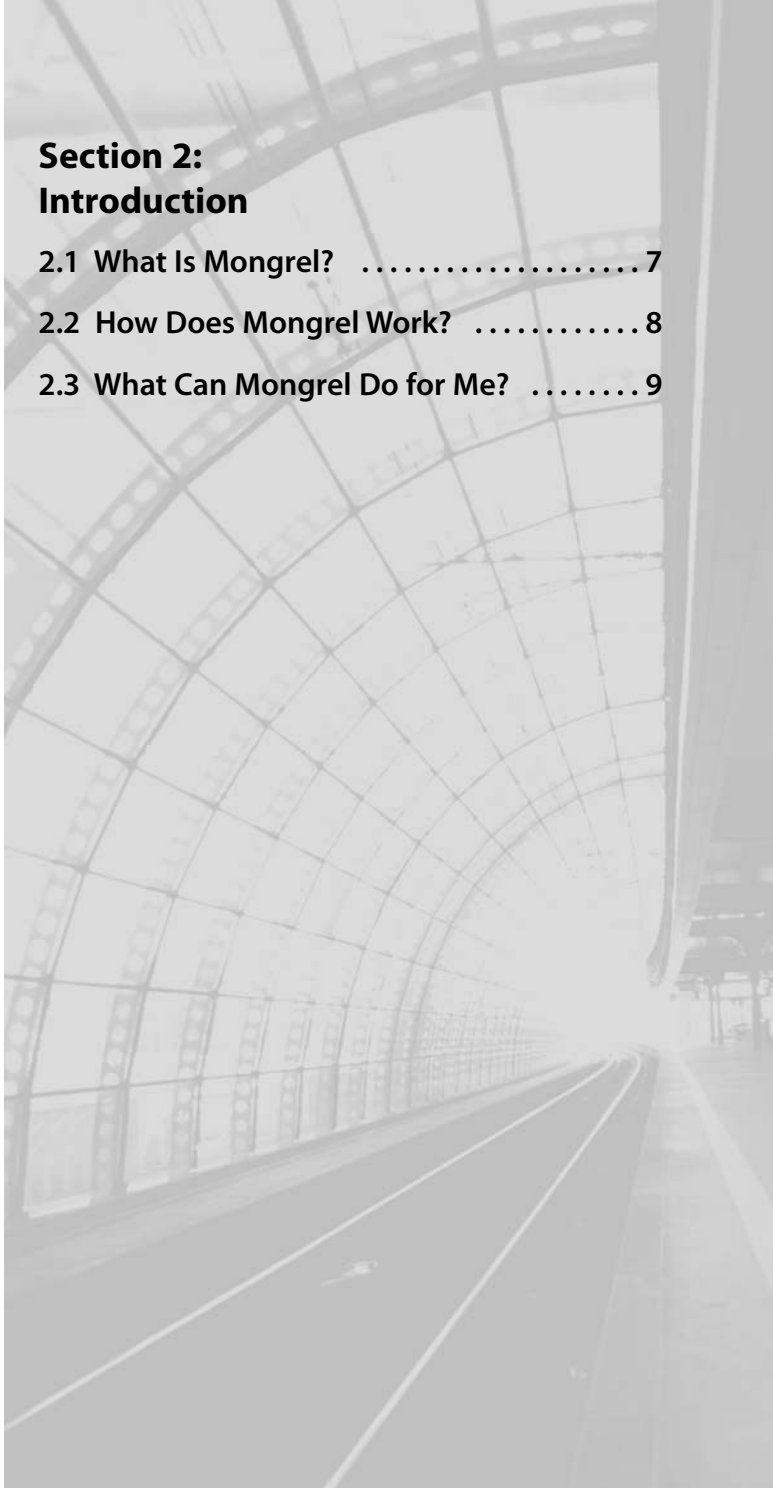
1.2 Zed Sez

We have created a special feature in this book called *Zed Sez*. Mongrel is certifiably *Opinionated Software*, and Zed Shaw, the author of Mongrel and coauthor of this short cut, is an Opinionated Developer. Mongrel represents somewhat of a "Best Hits" of the practices and techniques he has refined over time in his work. He wields his no-nonsense, pragmatic approach without fear or hesitation (as anyone on the Mongrel mailing list can attest), so we feel it is only appropriate to make space for some of Zed's rants to give you a real taste of what Mongrel is all about. You will see little sidebars throughout the book that come right from the heart (Zed even drew his own head). You may not agree with everything he says, but you probably should.

SECTION 2

Introduction

In this section we introduce Mongrel, explain its origins and history, and how it is used. We have found there are some common misconceptions about how it works or what exactly it does, so we will help clear these up so we can start off on the right foot. Even if you think you know what Mongrel is all about, make sure you read this first.



Section 2:
Introduction

2.1 What Is Mongrel?

Mongrel is a small, fast, mostly Ruby Web server.[1] It was created to do only a few things, but to do them really well: Make the development, deployment, and extension of Ruby applications Dead Simple. That's it.

2.1.1 A Little History

Zed Shaw started Mongrel in late December 2005. He wrote it because he was frustrated by the existing solutions for developing and deploying Ruby Web applications. FastCGI was abysmally broken and WEBrick was abysmally slow. He had written SCGI Rails Runner earlier in the year as a replacement for FastCGI, but he quickly hit a ceiling, as it was still just a middleman. Also, he was bored. So he wrote Mongrel to exorcise some demons, and to scratch an itch. It turns out that he was not the only one feeling frustrated, so his solution turned out to be pretty useful for a lot of Ruby developers and sysadmins. Mongrel's success is largely a result of Zed's approach to building software. In the Ruby development world, philosophy and practices are at least as important as the code itself, so Zed has forged Mongrel in his own way. It is a Web server; it is not a universal adapter. It is tight and small and secure, and he will be happy to explain why.

Mongrel is already packaged for distribution for the following flavors of Linux and will be included in Apple's next release of OSX (called Leopard, see http://www.apple.com/server/macosx/leopard/more.html for details).

[1] *By "mostly" we mean it uses some Ruby C extensions to handle the URI parsing, but otherwise it is all Ruby.*

ZED SEZ

When I started on Mongrel, the deployment landscape was nasty. It seemed that only 37signals knew how to deploy large-scale applications in Rails, and I think that's only because they sacrificed chickens to Ninhursag between bouts of Touretts-inspired screaming. I was working on SCGI Rails Runner, and that wasn't much of an improvement. What we needed was just a Web server that ran Rails as fast as FastCGI, but as simply as WEBrick. At every turn I tried to make sure Mongrel was easy to set up, but still flexible so it would fit into as many locations as possible.

2.2 How Does Mongrel Work?

The basic function of a Web server is to listen for HTTP requests and send back responses. The requests are usually just a URL, some headers, and sometimes a body (like form parameters). If an image was requested, the Web server will usually pick that file off the file system and send it along. If the request is intended to reach a Web application, the Web server will usually pass the request to the application, the application will generate a response in the form of headers and HTML, and the Web server will relay that back to the requesting party. Most Web servers have grown into hulking beasts because they are used as catch-all applications that come with every imaginable option, module, and feature. Mongrel does not wish to replace these kinds of applications, it just wants to serve HTTP requests really well, especially the kind that need a Ruby application. Mongrel has been designed from the start to be Dead Simple: easy to use, easy to extend, and pretty secure.

2.3 What Can Mongrel Do for Me?

This section describes what you can expect from Mongrel, whether you are a developer, sysadmin or manager.

2.3.1 For the Developer

Mongrel is great for local development.

If you are doing Ruby development, then using Mongrel is a no-brainer. It's much faster than WEBrick and can handle heavy development work (with all the logging and reloading turned on) easily. It installs with one command and configures with another. You can be up and running in less than a minute.

It integrates nicely with Rails (and other Ruby frameworks).

Mongrel was initially created to simplify developing and running Rails applications, so most Mongrel users are Rails developers. They have contributed some excellent plugins to handle common tasks like clustering, file uploads, and DRb process handling, and they are a great place to learn how to write your own. In June 2006 the Rails Core team integrated Mongrel with *script/server*, so if that is your comfort zone, you only need to have Mongrel installed.

Extend your application easily.

Mongrel has been designed to make extending it (or your Web application) easy. You can think of Mongrel as a tight Ruby library that deals with the sticky mechanics of HTTP, gives you simple access to the request, and otherwise stays out of your way. The interfaces for extending it are clean and easy to understand, so even if you shy away from Ruby outside of Rails-world, you should take

a look. Frequently you can speed up your application by offloading certain tasks and having Mongrel handle them instead. If you have your own Ruby library that is not one of the supported frameworks (see Section 3.4), you can write a handler so Mongrel can load it with all the special options you need. See Section 6 for details.

Mongrel includes excellent debugging tools.
Mongrel includes thorough tools to help you debug your application. You can watch the entire request and response process too, and write your own handlers and plugins to dig as deeply as you need to test your application and environment.

Mongrel is the same in production.
As you build, test, and deploy your application, you can rest assured that Mongrel will work exactly the same when in the production environment. This comes in handy when you need to run your tests, integrate with other systems and services, work with caching, etc. See Section 4 to see how Mongrel is used on both local development boxes and production environments.

2.3.2 For the System/Network Administrator

If you are wondering how Mongrel would fit into your system architecture, you shouldn't feel alarmed. Zed was a sysadmin for years, and spent so much time dealing with poorly designed packages that he knew what it would take to make Mongrel play nice with common server configurations.

Mongrel is a real Web server.
Mongrel is a fully HTTP-compliant server (more compliant that most), and can speak HTTP directly with all the network devices that common hardware and software configurations use. This saves you from the need to translate requests into CGI or use buggy or slow modules (like mod_ruby or FastCGI) to connect to your Ruby application.

Mongrel is a Ruby application.
Since Mongrel is written in Ruby, it can load and run any Ruby code natively, just like Tomcat does with Java Applications (except it doesn't need a 120MB footprint!). This is why you won't need something like FCGI, SCGI Rails Runner, or modules anymore to pass the HTTP Request to the Application Framework.

Mongrel can be clustered and deployed easily.
Mongrel has been designed to be Dead Simple to set up in production environments. This means that major concerns such as clustering and deployment have been addressed and best practices are already in place. See Section 4.2 to learn more about mongrel_cluster and Section 5 to see how folks are setting it up with Capistrano.

2.3.3 For the Manager

Discriminating minds choose Mongrel.
Mongrel has been widely seen as the clear choice among the available options for developing and running a Rails application. The same could be said for developing and running any Ruby application, but so far the biggest group of Ruby developers are using Rails to build their sites and Mongrel to run them. Mongrel is not a whole lot of code, and it hasn't even been around for a