

1

Getting Started with Rails

*First they ignore you,
then they laugh at you,
then they fight you,
then you win.*

— Mahatma Gandhi

Ruby on Rails is a highly productive Web application framework for the Ruby language. It will change the way you think about Web development and simplify the process of creating, deploying, and maintaining your Web applications. It could be argued that Ruby on Rails will ultimately make you a better developer.

These statements may seem like bold claims now, but over the course of the next few chapters, sufficient evidence is presented that will help to convince you otherwise.

The aim of this chapter is to provide you with an introduction to the Rails framework, and help you set up your favorite operating system so that it's ready to work with Rails.

What's in a Name?

The name Ruby on Rails is often shortened to Rails. Another common abbreviation is the acronym RoR (pronounced like a lion's "roar"). I feel it is important to clarify something that is often a source of confusion for newcomers: Ruby is a modern, object-oriented programming language, which predates Rails by about a decade. Ruby on Rails is simply the "full name" of a great Web framework written in Ruby, for Ruby developers. This name is also used for the URL of the official project website, available at <http://rubyonrails.org>.



Chapter 1: Getting Started with Rails

The Rise and Challenges of Web Development

Over the past few years the development world has experienced a radical paradigm shift from desktop to Web applications. It may be premature to call traditional desktop programs obsolete, but the rapid rise of Web-based software is a clear sign of the ever-increasing popularity and significance of Web development.

Web applications provide developers and users alike with a wealth of advantages. In particular, programmers can leverage a more immediate development, deployment, and maintenance cycle, while end users are able to utilize applications with desktop-like features and interfaces (often referred to as Rich Internet Applications or RIAs) directly from their browser. This enables users to access data in a platform-independent manner, on the operating system and device of their choice, and from anywhere an Internet connection is available. A significant part of the success of Web applications resides in this ability to respond well to the needs of a world that is continually more and more connected. Web development is therefore a stimulating and worthwhile endeavor.

The new Web — commonly dubbed Web 2.0 (a term that's attracted its fair share of critics) — poses a few challenges for developers, especially for beginners. The Web as a development platform is exciting, but far from perfect.

The main problem in this regard arises from the Web's origins. The HTTP protocol was created as a means to store and retrieve documents. HTML, on the other hand, is a markup language that was created to represent interlinked documents (hypertext). Traditionally, the Web's entire architecture was document-based.

Yet, its worldwide success and the growing reliance upon it by more than a billion people has forced the Web to rapidly evolve over the past 15 years to the point where multimedia and very complex applications are possible. In moving from an era of documents to one of rich applications and interfaces, a series of new technologies were introduced. It isn't far-fetched to say that we're pushing the Web far beyond the limits for which it was initially created.

It is very challenging to approach the new Web with general-purpose languages and tools. For example, though it is possible to write CGI scripts today, far better-suited tools exist, which make a developer's job much easier and more enjoyable, while delivering solid applications that are simple to maintain. These tools — and Rails is an excellent poster child for this — are specifically tailored for the new challenges of the modern Web and for this reason are often considered to be an example of Domain Specific Languages (DSLs). Rails and similar frameworks incarnate and adhere to the best practices assimilated by the industry, and attempt to hide the tedious, repetitive, low-level work required by Web applications from the developer.

Web applications are often developed by a small team and tend to change at a rather fast pace. This is in stark contrast to development scenarios that are present within the Enterprise world, where extensive planning and long release cycles are not uncommon. It is therefore necessary to use development methodologies and tools that allow the developer to be productive and embrace change.

It is also important to understand that a user typically interacts with a Web application through a browser, hence its limits will inevitably affect the user's experience on the Web.





Chapter 1: Getting Started with Rails

One of the toughest quirks to work around is the issue of cross-browser compatibility. The contents of non-trivial Web pages are displayed and behave differently based on the browser that's being used. In some cases, a Web application may not work at all in certain browsers.

Dozens of Web browsers exist, even limiting the playing field to the most popular and widely adopted choices (multi-browser compatibility); at the very least, attentive developers must test their applications with common browsers such as Internet Explorer, Mozilla Firefox, Safari, and Opera. Plus, one has to keep in mind that different versions of the same browser will typically render the content in a different manner (for example, IE 6 and IE 7). Sadly, this aspect is often overlooked, and it isn't uncommon for many companies and developers to verify their Web apps with Internet Explorer alone.

Browser vendors are making a conscious effort to improve their browsers in order to achieve better compliance with major Web standards and specifications, but it would be naïve to expect complete compatibility anytime soon. Coping with this flock of browsers adds to the complexity and can be detrimental to the fun of programming.

If you are already a Web developer or a Web designer, you're probably aware of the importance of adopting the W3C recommendations (for example, XHTML, CSS, and DOM), and international standards (for example, ECMAScript, aka JavaScript), as well as testing your application in multiple browsers.

Mentioning the names of a few commonly adopted languages brings up another issue: the modern Web is rather complex and, in order to work well, requires a set of server- and client-side technologies that you should be familiar with. The skills required encompass (depending on your role): a solid understanding of the HTTP protocol, XHTML, CSS, JavaScript, Ajax (or Flex, or Microsoft Silverlight), server-side programming, database design, SQL, Web Services, security, Web servers, and so on, with a list that could continue on for a good long while. Contrary to popular belief — that is perhaps a remnant of the early days of simple HTML homepages — Web development is a complex balance between art and engineering.

As the number of Internet users grows, so too does the demand for a higher degree of interaction, useful features, and — thanks to fast connections and technologies like Ajax — responsive and advanced Web UIs. A few examples of such Web applications are Gmail, Google Docs (an online Office suite), Google Maps, YouTube, Flickr, and more recently, Adobe Photoshop Express (Photoshop on the Web). In other words, the bar for what constitutes a good Web application has been raised considerably over the past few years.

Luckily for you, Ruby on Rails is the ideal tool when it comes time to approach these challenges.

What Is Rails?

Rails is an open source, cross-platform, full-stack, Model-View-Controller (MVC) framework for the Agile development of database-driven Web applications that use the Ruby language. This dense definition incorporates quite a few concepts that you may not be familiar with. Let's break things down so as to get a better overview of the framework.

Open Source

Rails is an open source project. It's released under a very liberal license (MIT) that enables you to freely modify, contribute, and distribute the framework. In the vernacular of the Free Software world, Rails is





Chapter 1: Getting Started with Rails

free as in beer and as in speech. As a Microsoft developer, you're probably used to proprietary software, where several of the components that you employ in your applications are closed source.

The fact that Rails is open source implies that, whenever you encounter a bug in the framework itself, you'll have full access to the source code of the libraries that constitute the framework. You will be able to identify where the problem lies, report it with accuracy, and even correct it yourself and submit a patch to the project. In other words, stepping through and reviewing the source code helps you to better understand how the framework operates and in turn enables you to build better applications.

Cross-Platform

Unlike ASP.NET (with an exception made for its alternative implementation through the Mono project), Rails runs on a number of platforms. The most popular choices within the community are operating system members of the *nix family (like GNU/Linux, Mac OS X, and *BSD), but Rails can be used on Windows as well.

Full-Stack

The term *full-stack* means that the framework provides you with a supportive, integrated environment in which it's possible to develop complete Web applications from start to finish. To borrow an expression from the Python world, Rails "comes with batteries included." Aside from handling the request-response cycle and providing you with the necessary libraries for easier database, server-side, and Web Service programming, on the client side, Rails also includes a JavaScript framework (Prototype) and a library (script.aculo.us) for creating Ajax-powered applications.

Support for testing, a very important topic for Rails developers, is baked-in as well. To top it all off, Rails provides you with a handy HTTP server known as WEBrick which, albeit meant for development purposes only, enables you to get started right away without having to worry about configuring more complicated Web servers.

If you're accustomed to developing with large frameworks such as ASP.NET or J2EE, Rails' full-stack nature may seem "not full enough." This is intentionally so, in order to avoid complexity and bloat in the framework and leave a core that's reasonably slim and extendable through free, third-party plugins. Rails' plugin architecture encourages code reuse and enables the developer to extend Rails' core behavior and out-of-the-box features in reusable plugins that are often shared with the whole world. As a Rails beginner, you can learn a lot by simply studying the code of high-quality Rails plugins, and you can take advantage of a wealth of functionalities by simply installing them within your own Rails projects on an as-needed basis. Plugins are, in my opinion, one of Rails' best features.

The MVC Pattern

MVC is an architectural pattern designed to satisfy the principle of separation of concerns. When successfully applied to the design of a software project, it promotes the isolation of the user interface from the business logic. This separation is extremely important in order to create maintainable applications, and it's especially true on the Web, where the distinction between business logic (server side) and presentation layer (client side) is almost intrinsic to the medium.

Models represent the data, views the user interface, and controllers act as coordinators, controlling the flow of the application. Controllers are the glue of the application; they allow modification and retrieval



Chapter 1: Getting Started with Rails

of model data, and preparation of the content so it can be rendered in the view. Changes performed to the view by, your designer, for example, shouldn't affect the team of developers working on the backend.

If you have done ASP (classic) or PHP development, where access to the database, business logic, and presentation layer are often intermingled, you'll find RoR's approach rather refreshing, tidy, and even elegant.

Rails generates the skeleton of an application and, in effect, forces you to adopt an MVC-style of programming. Every Rails application applies separation between the model, the view, and the controller, storing the files for each of these components in different folders. You'd have to go out of your way to be able to intentionally break the enforcement of this pattern in Rails. But don't let this apparent strictness toward MVC get you down. In most cases ASP.NET developers who switch to Rails find themselves liberated by the rigidity of the language (be it C# or VB.NET) and the framework they were used to.

In Chapter 2, the MVC pattern is described in further detail, with particular emphasis on the way Rails implements each entity and their interaction with each other.

Agile Development

Agile development is a perfect match for the Rails developer. It can be argued that Rails is an attempt to bring Agile methodologies to the Web, where development has often been led by opposite principles. If you are not familiar with the Agile movement, I invite you to read the following Manifesto for Agile Software Development (<http://agilemanifesto.org>). The Agile Manifesto was the brainchild of 17 pioneers who decided to start a movement to improve the practice of software development. Among these folks, three individuals (Martin Fowler, Dave Thomas, and Andy Hunt) are deeply involved with the Ruby and Rails communities.

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

— Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas.

Rails absolutely embraces each of these four principles, and the so-called "Rails Way" of programming faithfully adheres to the Agile methodologies. Rails' main philosophies are often condensed into mnemonic mantras that originated and are well-known in the Extreme Programming (XP) and Agile communities. The three canonical ones are: Don't Repeat Yourself (DRY), Convention over Configuration, and You Ain't Gonna Need It (YAGNI).



Chapter 1: Getting Started with Rails

“Convention over Configuration” expresses Rails’ philosophy of adopting a series of sensible assumptions, which frees programmers from defining and configuring every single detail of their application. Most Web applications share common elements and Rails requires configuration only when the conventions adopted by the framework are not endorsed and, therefore, need to be overwritten. For example, Rails assumes that the class `Order` will correspond to the pluralized `orders` table in the database. Unless you’re required to overwrite this convention, the mapping between the Ruby class and the database table is automatic and no explicit configuration is required.

Even if you intend to overwrite a given convention, unlike with other frameworks, Rails’ configuration is performed in simple readable text files, rather than using verbose XML files. A medium-sized application in a traditional framework can end up having hundreds, if not thousands, of lines of XML just to define a correspondence between the objects within the code and the relational structure in the database. This is intentionally not the case with Rails, even if it provides programmers with the ability to configure and overwrite conventions in order to meet their needs.

The “Don’t Repeat Yourself” mantra implies that writing less redundant code and reducing duplication ends up producing maintainable and less bug-prone applications, which can easily evolve and change. A change in a given point of an application should not affect unrelated elements and should be properly reflected in related ones, without requiring multiple changes. Localizing change is a principle that is fundamental to Rails development, and the framework structure promotes and enforces this.

Finally, the “You Ain’t Gonna Need It” principle is a reminder about the importance of implementing features that are actually needed now, and fighting the urge to write code for features that may only be necessary in the future. This approach to software engineering has been embraced by Rails’ creators as well as being common among developers using Rails. Following the YAGNI principle offers a greater focus on the required core functionalities, keeps software lean, and helps in retaining the application’s flexibility to change as needed. 37signals endorses this principle and extends it into the principle of “less software,” which is aimed at outsmarting the competition with focused software that has fewer features than their competitors. So far, it has worked wonders for them.

Understanding the Rails philosophy of Web development is essential to successfully employing the framework and becoming an effective programmer.

These principles are closer to the Unix philosophy, rather than the one that’s common within the Microsoft development world. This by no means implies that, as a Microsoft developer, you may not have already adopted and sought out these development practices, but it was essential to explain them further and state their importance throughout the book.

Database Driven

Rails assumes that each Web application is going to store data within a database. It is a reasonable assumption for all but the most trivial of applications. More importantly, Rails uses an Object Relational Mapper (ORM) Ruby library called *ActiveRecord*, that follows the Active Record design pattern as defined by Martin Fowler in his popular book, *Patterns of Enterprise Application Architecture* (Addison-Wesley 2002).

ActiveRecord greatly simplifies CRUD, the four basic functions of persistent storage — create, read, update, and delete — enabling you to favor Ruby code over SQL queries (most of the time). It’s the abstraction that allows domain models to wrap database objects and handle their underlying relationships. Thanks to a series of adapters, *ActiveRecord*, and therefore Rails, can be used with all of the most



Chapter 1: Getting Started with Rails

popular databases, from the file-based SQLite (Rails' current default) to more "enterprisey" choices such as Microsoft SQL Server, IBM DB2, or Oracle. The Rails community has a clear preference toward open source databases such as MySQL and PostgreSQL, but you are more than welcome to adopt whatever RDBMS you have at hand or are more comfortable with.

This book uses the default database (SQLite) and recommends that you do the same to follow along, but don't be too concerned if you're using a different one. That's the beauty of ActiveRecord's abstraction: the Ruby code will be (virtually) the same no matter what database is being used. That said, there are special considerations for SQL Server, DB2, and Oracle, and pointers for these are provided in Chapter 11.

Ruby: Rails' Secret Sauce

Ruby is an open source, modern, object-oriented programming language — and a fantastic one at that. It synthesizes the best lessons learned from other programming languages like Smalltalk, Perl, and Lisp, combining the elegance of the object-oriented paradigm (in Ruby everything is an object; there are no primitive types) with the immediacy of a scripting language, further combined with functional programming. If you are not familiar with this last programming style, but have had a chance to try out the language extensions to C# 3.0 and Visual Basic 9 — provided by LINQ in the .NET 3.5 Framework — you've already had your first contact with the functional world.

If you're used to programming languages like C#, VB.NET, C++, or Java, you'll be blown away by how Ruby is concise yet readable, expressive, powerful, and easy-to-learn. Its dynamic, interpreted nature makes it much less tedious to work with, when compared to the aforementioned compiled programming languages.

Ruby is a very high-level language and it's truly Rails' secret sauce. Its flexibility and reflective nature make it an ideal language with which to implement a framework/DSL like Rails.

A good part of the fun of writing Rails applications lies in the fact that you get to code in Ruby, a language that was invented by Yukihiro Matsumoto (commonly known as Matz), with the specific intention of being programmer-friendly, productive, and maintainable.

Rails takes Ruby to the next level, by expanding its out-of-the-box capabilities through utility classes and extension to the Standard Library; on the other hand, Ruby's openness and flexibility enable you, the developer, to extend Rails to suit your needs.

Rails is written in Ruby for Ruby programmers, so it is essential to be well versed in Ruby before attempting to create interesting Rails applications. This is a common mistake by people who are trying to learn Rails. They skip Ruby and dive right into the framework, tempted by Rails' approachability. The end result is not pretty, with a lot of confused newcomers and very basic Ruby questions popping up in the Rails mailing lists and forums.

In this book Ruby is not an afterthought. I deemed it crucial to include two whole chapters dedicated to the language, as opposed to a simple appendix as often happens in other Rails books. I've also tried to emphasize and clarify aspects of the language throughout this book, whenever required.

Greater Than the Sum of Its Parts

Reading about Rails features and design choices may lead you to realize that Rails didn't invent anything particularly new. The powerful MVC pattern was first described back in 1979 and was already



Chapter 1: Getting Started with Rails

adopted by other MVC frameworks. Conversely, the Active Record pattern was well known, too. Over the past few years, all sorts of Ajax libraries and frameworks have been released into the wild; and Test Driven Development (TDD) and the Agile methodologies were also not invented by Rails.

What Rails did was to put each component in the right place, in a coherent manner, while attempting to keep everything as simple as possible for the programmer. The end result is a powerful and fun toolkit that lets you concentrate on the actual application, rather than on small technical details that are repeated over and over in each project. Does the programmer really need to specify the whole connection string in order to access the data within the database? Rails doesn't think so. Rails favors "Convention over Configuration" because when these conventions are sensible, they truly free the developer from having to take care of minutiae in configuration files.

This homogeneous set of features, conveniences for the developers, "best practices," and guiding philosophies make Rails invaluable when it comes to producing solid Web applications in very short time frames, when compared to other existing solutions. The Rails community has done a good job of conveying the framework's strengths, and understandably many developers are excited about the chance to use Ruby on Rails in their projects.

A Brief History of Rails

To fully understand the reasons behind Rails' design choices, it is beneficial to very briefly learn about its history.

Understanding Rails' Origins

Rails was released to the world back in July 2004. The framework was extracted from a Web-based management and collaboration application project called Basecamp. Rails was created by David Heinemeier Hansson (often referred to simply as "DHH"), a Danish programmer and partner with 37signals, the firm that produces Basecamp and other similar Web applications. This brief piece of information offers us some important preemptive insight.

Rails was not designed by a committee. It was extracted from a real-world application. Even after Rails' incredible success, David has insisted that there won't be a Rails, Inc. because he firmly believes in the importance of working on real applications and only then applying the most useful lessons learned (and possible missing features) back into the framework.

This approach guarantees that Rails doesn't end up becoming a bloated framework that includes all sorts of features, to satisfy the requirements of any possible company or scenario out there. Rails is intentionally general enough to be used for a wide range of applications, but its focus has always been the needs of 37signals and other companies/developers who take up similar principles. That's where Rails really shines. To paraphrase what David said during a keynote at Startup School (<http://startupschool.org>): 37signals targets Fortune 5,000,000 companies.

37signals' team has strong opinions about how software development should be done. They embrace Agile development, simplicity, and software that focuses on a relatively small number of features (the previously mentioned YAGNI principle). Rails is opinionated software because it was tailored for the needs of 37signals, their products, and their way of developing. The good news is that, not only are they very successful, but the practices that they promote are well proven within the industry, and make a great deal of sense from a business and engineering standpoint.



Chapter 1: Getting Started with Rails

Born into this kind of context, Rails makes assumptions about your applications. It assumes that you are going to use a database and that you'll be dividing your work into three environments: development, test, and production. It assumes that you'll be starting from scratch, rather than working with legacy databases. Ruby is not the fastest language out there, but that's acceptable because from 37signals' viewpoint, a need for extra hardware implies a greater number of paying customers. Developer time is much more expensive than hardware. Having to make a choice while creating Rails, they opted in favor of programmer productivity, code maintainability, scalability, and speed of development, as opposed to the raw speed of the framework and the chosen language.

37signals

If you'd like to learn more about 37signals, I invite you to read their popular design and usability blog, "Signal vs. Noise" (<http://www.37signals.com/svn/>) and their book on how to build successful Web-based applications, called "Getting Real" (<https://gettingreal.37signals.com>).

Rails' origins help you better comprehend what its sweet spot is. Rails is particularly well suited to applications that have the following characteristics:

- ❑ Applications and sites that aren't trivial. Employing a whole framework for a page or two is still probably overkill and there are more straightforward solutions.
- ❑ Applications built from scratch, following Rails conventions. Working against Rails conventions is possible, but if your project heavily requires going against the stream, working with Rails won't be as easy. An example of this situation is when you are trying to deal with legacy databases and corporate environments.
- ❑ Applications hosted on VPS (Virtual Private Servers), dedicated servers, or elastic/cloud computing services. Shared hosting is an acceptable solution for non-critical applications and low volume websites, but it is neither ideal nor within Rails' sweet spot. You can read more on deployment options and considerations in Chapter 11.

Rails doesn't usually prevent you from building any type of applications, but it is opinionated and you'll be able to get the best out of it when you take advantage of "the Rails Way" of development or, in other words, when your opinions match those of Rails.

If the core Rails functionalities don't quite cut it for your project, you can still decide to use other open source plugins (or write them yourself) in order to allow Rails to behave in manner that's closer to one of your specific needs. For example, you may require support for composite primary keys, which by default are not supported by Rails. There is a homonym plugin that extends ActiveRecord to add this functionality.

It isn't uncommon for the Rails core team (a group of a few open source developers captained by David) to reply to requests of the "wouldn't it be cool if Rails was able to..." sort with the acronym PDI, which stands for Please Do Investigate. Theirs is not a flippant answer, but rather a pragmatic one. It's an open source project after all and anyone can contribute or pay someone else to do it for them in order to get the kind of features that they may require for their own purposes, which don't quite fit into the Rails core.

If your development style, environment, and practices are entirely opposite to the Agile ones promoted by Rails, chances are that you have a bigger problem than deciding whether or not Rails is a good tool for you. In this case, the answer is clear: Rails may not be the best tool in this kind of context, and a .NET or J2EE solution might end up being less problematic.



Chapter 1: Getting Started with Rails

Powering the Web 2.0

Since its release in the summer of 2004, Rails has managed to become one of the most used and appreciated frameworks on the Web. By 2006, Rails had arguably already achieved its tipping point, and nowadays most developers have heard about Ruby on Rails. It quickly became the tool of choice for most of the (so-called) Web 2.0 startups, and today is widely adopted by some of the largest sites on the Web.

Scribd.com, YellowPages.com, Hulu.com, Twitter.com, RevolutionHealth.com, 43things.com, Helium.com, and Funnyordie.com are but a few examples of popular sites that are currently written in Rails, which you may have visited or heard of.

And the list of Rails users doesn't end with startups and popular websites. Companies of all sizes are employing Rails talent and starting new projects, embracing Ruby and a more Agile style of programming in pursuit of productivity. While typically very popular with smaller and medium companies, Rails has also been used within the borders of giants like IBM (which I work for), Amazon, Yahoo!, NASA, Oracle, EA, BBC, Cisco, and a long list of other successful Fortune 500 members.

Endorsements

"Ruby on Rails is a breakthrough in lowering the barriers of entry to programming. Powerful Web applications that formerly might have taken weeks or months to develop can be produced in a matter of days."— *Tim O'Reilly*

This enthusiastic quote from the founder of O'Reilly Media is just one of many great comments that Rails has received, from all sorts of experienced developers and IT veterans. You can read more at <http://rubyonrails.org/quotes>.

Ruby on Rails took the Web by storm and, along with Ajax, it became one of the greatest "revolutions" in modern Web development history. In fact, Rails' influence isn't limited to the Ruby community. It helped popularize the concept of MVC for many beginners, and inspired other developers to start similar projects (or clones) using Ruby and other programming languages, including but not limited to C#, PHP, Python, Java, and even JavaScript.

You may be familiar with the fact that the .NET community created its own open source version, called MonoRail, through the Castle Project (<http://castleproject.org>) before Microsoft made the wise move to respond to Rails' success with its ASP.NET MVC framework.

The Rise of Ruby

Despite its tagline of being a "programmer's best friend," and its ever growing popularity in its homeland (Japan), Ruby's worldwide adoption was initially limited by its lack of English documentation. In 2000, with the appearance of the first English literature on Ruby, and the involvement of "The Pragmatic Programmers" within the community, Ruby started to become more widely used. But the advocacy and promotion from early adopters was not enough to bring it directly into the spotlight because, at this stage, relatively few programmers had even heard of this thing called Ruby.

When Rails became such a smash hit, developers started using Ruby and began to appreciate it for its own merits. Today, most people still use Ruby to develop with RoR, but it has become quite a common choice outside of the Web or in conjunction with alternative Web frameworks, too. Ruby made Rails great, and Rails made Ruby much more common and accepted within the development world.



Chapter 1: Getting Started with Rails

This works well for Rails developers, who can benefit from a larger Ruby community that's ready to improve the existing implementation of the language and share libraries for all sorts of development purposes.

Installing Rails

This section takes a break from discussing theory in order to get your environment set up. It provides you with step-by-step instructions for installing Ruby, Rails, and all the other necessary components of a development stack on Microsoft Windows, GNU/Linux, and Mac OS X.

For the Windows installation, two different methods are illustrated: the first leverages an installer and the second uses a learning environment known as Instant Rails.

Can You Use Rails on Windows?

Macs are very popular within the Rails community. If you ever get the chance to attend a Rails conference, you'll see a very high percentage of Apple laptops. The entire Rails core team uses Macs. The community seems to be keen on GNU/Linux as well, which is another common option for Rails development, and the most popular deployment one. In fact, most Rails hackers that I know develop on a Mac and deploy on GNU/Linux or *BSD, a combination that I adopt and enjoy myself.

You probably won't see many Windows systems. Generally speaking, the community prefers Unix-like environments and few people would admit in public to consciously choosing Windows at any Ruby or Rails venue. There are many reasons for this, most of which are cultural ones. Does this mean that you can't use Windows or that you'll be the only one doing it?

Although this book can be followed by utilizing any operating system of your choice, the assumption is that as a Microsoft developer, you're primarily familiar with Windows .NET, and other Microsoft developers often face a culture clash when trying to learn Rails. This book tries to minimize that by letting you take advantage of the tools and skills that you're already familiar with. For this reason, I've employed Windows to write this book, and the screenshots are from (the much debated) Vista.

Developing with Rails on Windows is usually not a problem, given that in the end we are just editing files, but it's a fair assessment to say that both Ruby and Rails work much better on Unix-like operating systems. For example, Ruby is significantly faster on Ubuntu, Fedora, or Mac OS X than it is on Windows XP (where it still performs better than it does on Vista).

Some people even go so far as to run Rails from within a virtual machine (using GNU/Linux or BSD) in Windows. There are also specific cases of libraries or particular deployment options that are well supported on, say, Linux, but not on Windows due to crashes and other problems. In general, it is recommended that you deploy your Rails applications on *nix systems, but it is understood that doing so is not always an option. You'll feel reassured to know that Rails can be deployed on Windows and that this is commonly done. Chapter 11 tells you how.

I wanted to give you a heads up about the special relationship between Ruby, Rails, their respective communities, and Windows, but please don't let this discourage you. Windows is a viable platform for Rails development and there are several initiatives to further improve the current situation. Rails' success and



Chapter 1: Getting Started with Rails

mainstream acceptance will also depend, in my opinion, on its ability to succeed on Windows, which is still the most popular operating system out there.

If you enjoy using Windows, I'll let you in on a surprising piece of information: Ruby is very popular on Windows. In fact, I'm going to prove its popularity in a somewhat scientific manner. Many Ruby and Rails projects are hosted at a site called RubyForge (<http://rubyforge.org>), the equivalent of SourceForge or CodePlex for Ruby. The most popular download is, surprise-surprise, a one-click installer for installing Ruby on Windows. As I write this, it has been downloaded about 3 million times. And the third most popular download is Instant Rails, a package that helps you to quickly get up and running with Rails on Windows (it's had more than 700,000 downloads so far). Rails on Windows might very well be a silent majority. In other words, and to quote Michael Jackson: you are not alone.

Installing on Windows

For this book you'll need to install the following components:

- ❑ **Ruby:** The Ruby interpreter plus its core and Standard Library. Ruby 1.8.6 or newer is required for modern versions of Rails (for example, 2.2.2 and newer).
- ❑ **RubyGems:** A packaging system used to install, update, and remove Ruby libraries and programs (packaged and distributed as “gems”).
- ❑ **Ruby on Rails:** All the gems required to run Rails.
- ❑ **Mongrel:** A much faster server that we'll use in place of WEBrick.
- ❑ **SQLite3:** A lightweight, file based, ACID (Atomicity, Consistency, Isolation, Durability) compliant database that's available in the public domain.
- ❑ **sqlite3-ruby:** A gem used by Rails to access SQLite3 databases.
- ❑ **Subversion:** An open source version control system required to install many of Rails' plugins.

Installing the One-Click Ruby Installer

Windows is one of the easiest platforms on which to install such a complete Rails stack, thanks to the abovementioned One-Click Ruby Installer, which takes care of the first two elements in the preceding list.

To install it, follow these simple steps:

1. Visit the homepage of the project at <http://rubyforge.org/projects/rubyinstaller> and click Download on the right-hand side, half way through the page.
2. On the download page, there will be several versions available. Ensure that you download the latest one. At the time of this writing, the current stable version is `ruby186-26.exe`; click this or a more recent version if available.
3. Double-click the downloaded executable to start the setup wizard. Go through the installation process, accepting the default options (see Figure 1-1). The operation may take a few minutes to complete.



Chapter 1: Getting Started with Rails

At this point Ruby, RubyGems, and a handy text editor called SciTE are all installed in `c:\ruby` (unless you specified a different location during the installation process).

To verify that the installation was successful, you can run a quick sanity check by opening the command prompt (`cmd.exe`) and running the following command:

```
ruby -v
```

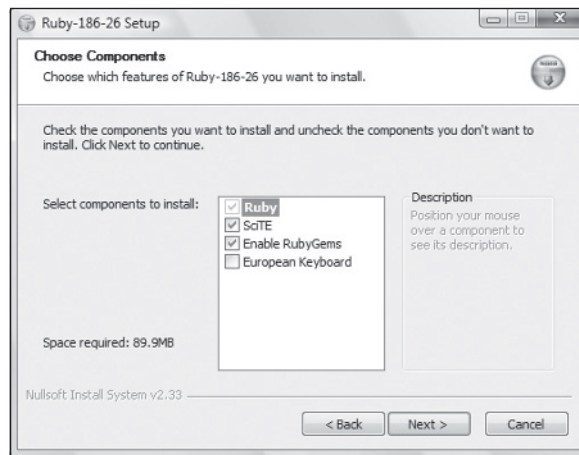


Figure 1-1

Ruby should reply, stating its version number to the prompt; for example, on my machine I obtain the following: `ruby 1.8.6 (2007-09-24 patchlevel 111) [i386-mswin32]`. Ensure that you have Ruby 1.8.6 or newer, because this is a requirement for Rails.

Updating RubyGems

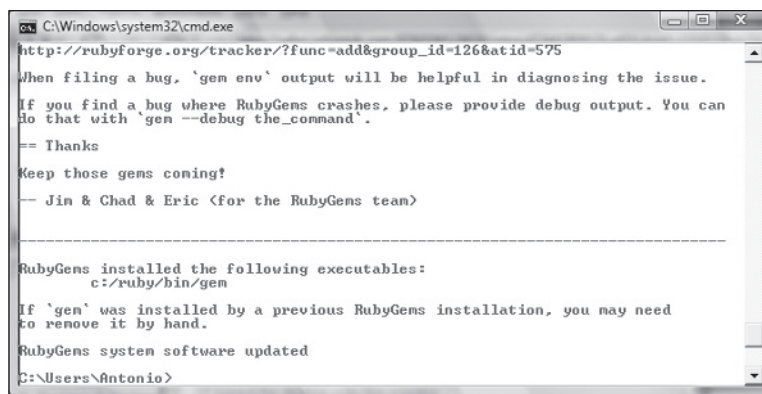
The version of RubyGems that ships with the One-Click installer may not be the most recent one. You can verify the installed version number by running `gem -v` from the command line and upgrade to the latest version by issuing the following:

```
gem update --system
```

This will fetch and install the latest version of RubyGems from the RubyForge repository, so you'll need to be connected to the Internet. When the update is finished, you should get the message "RubyGems system software updated" or similar, as shown in Figure 1-2.

Running `gem -v` again will give you the peace of mind that the update was indeed successful and that the command is still working.

Chapter 1: Getting Started with Rails



```
C:\Windows\system32\cmd.exe
http://rubyforge.org/tracker/?func=add&group_id=126&atid=575
When filing a bug, 'gem env' output will be helpful in diagnosing the issue.
If you find a bug where RubyGems crashes, please provide debug output. You can
do that with 'gem --debug the_command'.
-- Thanks
Keep those gems coming!
-- Jin & Chad & Eric (for the RubyGems team)

-----

RubyGems installed the following executables:
      c:/ruby/bin/gem
If 'gem' was installed by a previous RubyGems installation, you may need
to remove it by hand.
RubyGems system software updated
C:\Users\Antonio>
```

Figure 1-2

Now that you've updated the system, you can proceed to update the actual gems that came with the One-Click installer by running `gem update` from the command line. You can obtain a list of installed gems by running the following command:

```
C:\> gem list
*** LOCAL GEMS ***

fxri (0.3.7, 0.3.6)
fxruby (1.6.18, 1.6.12)
hpricot (0.6.164, 0.6)
log4r (1.0.5)
ptools (1.1.6)
rake (0.8.3, 0.7.3)
rubygems-update (1.3.1)
sources (0.0.1)
test-unit (2.0.2)
win32-api (1.3.0, 1.0.4)
win32-clipboard (0.4.4, 0.4.3)
win32-dir (0.3.2)
win32-eventlog (0.5.0, 0.4.6)
win32-file (0.6.0, 0.5.4)
win32-file-stat (1.3.2, 1.2.7)
win32-process (0.6.0, 0.5.3)
win32-sapi (0.1.4)
win32-sound (0.4.1)
windows-api (0.2.4, 0.2.0)
windows-pr (0.9.8, 0.7.2)
```

As you can see in the preceding output, when multiple versions of a gem exist, they are listed as well between parentheses.

Installing Rails

Now that the must-have RubyGems packaging system is installed, you can use it to install a fresh copy of Rails by issuing the following command:

```
gem install rails -v 2.2.2
```

RubyGems Improvements

RubyGems has come a long way. In its previous versions, the developer had to indicate the gem version and platform required during the installation of a gem. If you want to install a specific version of a gem, you can now use the `-version` or `-v` option (for example, `-v 2.2.2`).

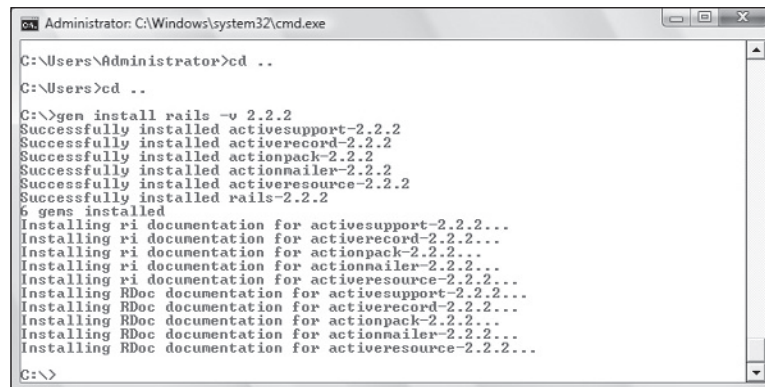
The dependencies weren't installed by default either, and the `-y` or `-include-dependencies` option was required to include them. Now that gem automatically installs the dependencies, you can specify the `-ignore-dependencies` option in the rare occurrence when you don't want them to be installed.

This command will fetch the `rails` gem version 2.2.2 and all its dependencies from the default source repository at `http://gems.rubyforge.org` (you can run `gem source -l` to see a list of sources used by the command). If you omit `-v 2.2.2`, the command will install the latest available version. This book uses version 2.2.2 so it's recommended that you follow along with the same version even if version 2.3 will be out by the time you read this.

No huge differences exist between Rails 2.2.2 and 2.3. At the time of writing 2.3's release date has not been announced but I will point out throughout the book when the known differences exist.

The installation process may take a while, as gem proceeds with installing Rails and five other gems plus their documentation, as shown in Figure 1-3.

To see a list of remote gems you can use the `-remote` option. When used in conjunction with `list` it can help you find gems by their name. For example, `gem list sql -remote` will show you a list of gems that start with `sql` (it's not case sensitive).



```
Administrator: C:\Windows\system32\cmd.exe
C:\Users\Administrator>cd ..
C:\Users>cd ..
C:\>gem install rails -v 2.2.2
Successfully installed activesupport-2.2.2
Successfully installed activerecord-2.2.2
Successfully installed actionpack-2.2.2
Successfully installed actionmailer-2.2.2
Successfully installed activereource-2.2.2
Successfully installed rails-2.2.2
6 gems installed
Installing ri documentation for activesupport-2.2.2...
Installing ri documentation for activerecord-2.2.2...
Installing ri documentation for actionpack-2.2.2...
Installing ri documentation for actionmailer-2.2.2...
Installing ri documentation for activereource-2.2.2...
Installing RDoc documentation for activesupport-2.2.2...
Installing RDoc documentation for activerecord-2.2.2...
Installing RDoc documentation for actionpack-2.2.2...
Installing RDoc documentation for actionmailer-2.2.2...
Installing RDoc documentation for activereource-2.2.2...
C:\>
```

Figure 1-3

Gem places the `rails` command within `c:\ruby\bin`, which is in the Windows's Path environment variable and therefore executable from any command prompt.

Running `rails -version` (or its shorter version `-v`) tells you which version is currently active.



Chapter 1: Getting Started with Rails

Installing Mongrel

It's nice that Rails ships with an HTTP server. Unfortunately, even strictly for development purposes, it's not very fast. Through RubyGems you can easily install a much faster replacement called Mongrel that, in more elaborate configurations, is often used to run some of the largest Rails websites out there.

To install Mongrel, simply run:

```
C:\> gem install mongrel
Successfully installed gem_plugin-0.2.3
Successfully installed cgi_multipart_eof_fix-2.5.0
Successfully installed mongrel-1.1.5-x86-mswin32-60
3 gems installed
Installing ri documentation for gem_plugin-0.2.3...
Installing ri documentation for cgi_multipart_eof_fix-2.5.0...
Installing ri documentation for mongrel-1.1.5-x86-mswin32-60...
Installing RDoc documentation for gem_plugin-0.2.3...
Installing RDoc documentation for cgi_multipart_eof_fix-2.5.0...
Installing RDoc documentation for mongrel-1.1.5-x86-mswin32-60...
```

This command installs the Windows version of Mongrel and the required dependencies as well.

A second gem, `mongrel_service`, exists for installing Mongrel as a Windows service. This is not needed for development but it's very useful when deploying an application, to ensure that Mongrel automatically starts if the machine reboots. You shouldn't worry about this gem until you are ready to deploy your application. This subject is discussed in Chapter 11.

Installing SQLite3 and `sqlite3-ruby`

SQLite version 3 is a very nice, lightweight, file-based (like Microsoft Access) relational database. It's ideal for quick prototyping (in development mode) even though you'll probably want to use a data server when dealing with all but the smallest amount of traffic in a production setting.

MySQL used to be the default database system for Rails, but now SQLite has taken its place, lowering the entry barrier for developing in Rails even further and quickly allowing you to get started. There are no users, authentication, or ports to configure: just simple files. In this book, I decided to stick with the default database. Switching to a different database system is most often trivial.

Installing SQLite is a piece of cake. Just download the DLL contained in a zip file that's available from the official website at <http://www.sqlite.org/download.html>. The current version at the time of writing is 3.6.10, so I downloaded the file `sqlitedll-3_6_10.zip` containing `sqlite3.dll`. You should grab the most recent version available.

Extract the zip file and place the DLL in a location on your path. `c:\ruby\bin` is a good place, given that you're installing SQLite specifically for Rails development purposes.

You should also download the command-line program for accessing and modifying SQLite databases. It's available on the same download page, and its file name (as of this writing) is `sqlite-3_6_10.zip`. Again, extract it and place the `sqlite3.exe` file in `c:\ruby\bin`. From now on, running `sqlite3` from the command line opens the SQLite3 shell. Beginning with Rails 2.1, this shell can also be invoked using a Rails script (`dbconsole`), as explained in Chapter 5.



Chapter 1: Getting Started with Rails

At this point you'll need to install the Ruby bindings through RubyGems as follows:

```
C:\> gem install sqlite3-ruby -v 1.2.3
Successfully installed sqlite3-ruby-1.2.3-x86-mswin32
1 gem installed
Installing ri documentation for sqlite3-ruby-1.2.3-x86-mswin32...
Installing RDoc documentation for sqlite3-ruby-1.2.3-x86-mswin32...
```

The preceding command specifies version 1.2.3 because it ships with the Ruby bindings in binary form. The latest release, 1.2.4, attempts to build the Ruby extension from source, which is more complicated on Windows because it requires Ruby's development headers and nmake, Microsoft's version of make.

Installing Subversion

Subversion (SVN) is a very popular open source version control system. Though it might not be very accurate from a technical viewpoint, it may help you to think about SVN as an efficient and whittled down version of Visual SourceSafe or Microsoft Team Foundation Server.

SVN and similar alternatives are crucial for working on real projects, but in this context I invite you to install the SVN client tools, because they're required for installing Rails plugins. When you attempt to install a plugin, Rails essentially performs a checkout from the given remote Subversion repository you specified.

On Windows, you can get SVN from the URL <http://subversion.tigris.org/servlets/ProjectDocumentList?folderID=91>. The most recent setup file in the download list will do (in my case, that's `svn-1.4.6-setup.exe`). All that's left to do is for you to double-click the downloaded installer and go through the default setup process.

Setup takes care of concatenating the SVN bin directory (for example, `c:\Program Files\Subversion\bin`) to your Path environment variable. Opening a new command prompt should now allow you to run `svn` and obtain in return a Type 'svn help' for usage message. It doesn't do anything too useful, but it's a good sanity check that guarantees you that `svn.exe` is available from the command prompt.

Many plugins are released on GitHub.com, a site that hosts thousands of projects that use the Git distributed revision control system. For this reason, you may want to install a Git client as well. On Windows, you could install `msysGit` (<http://code.google.com/p/msysgit/>). If you are new to Git, you should also check out "An Illustrated Guide to Git on Windows" available at <http://nathanj.github.com/gitguide/tour.html>.

Configuring Instant Rails

The original author of the One-Click installer (Curt Hibbs) created an alternative project called Instant Rails. Unlike the One-Click installer above, Instant Rails doesn't require any installation, nor does it modify your environment. You download a zip file, extract it to the folder of your choice, and you will automatically have a full Rails stack at your fingertips. The current version includes Ruby, RubyGems, Rake, Rails, Mongrel, support for SQLite3, MySQL, Apache, and PHP (for `phpMyAdmin`, which is used to manage MySQL), and a couple of sample Rails projects.

It essentially contains anything that'll you need to get started with Rails. You will need to manually install SVN, but other than that, you'll be all set (which helps explain the popularity of this package). Because it's such a great all-in-one package, it is very commonly used by Rails beginners who are working with Windows.



Chapter 1: Getting Started with Rails

You could use Instant Rails while reading this book, but I don't recommend that you do so. Instant Rails is meant to be a development-only package, and it's not typically used in production environments. You're also somewhat tied to its management application (InstantRails Manager), even though you could manually configure the binaries to be accessible throughout the whole system. For as great as it is, Instant Rails remains a prepackaged solution that is not as flexible and "solid" as the manual installation previously discussed. On top of that, it could be using different version numbers and make this book harder to follow.

With that in mind, there are still times when you can't install software on your Windows system or, for whatever reason, prefer not to touch the existing environment. In such situations, Instant Rails is the right solution and you should have no qualms about giving it a spin. To install Instant Rails, follow these steps:

1. Download Instant Rails from <http://rubyforge.org/projects/instantrails/>. As usual, click Download and grab the latest stable version of the zip file that's available, which has the following format: `InstantRails-X.Y-win.zip`, where X and Y (obviously) make up the version number.
2. Extract the contents (see Figure 1-4) of the zip file to a convenient location, like `c:\InstantRails`.

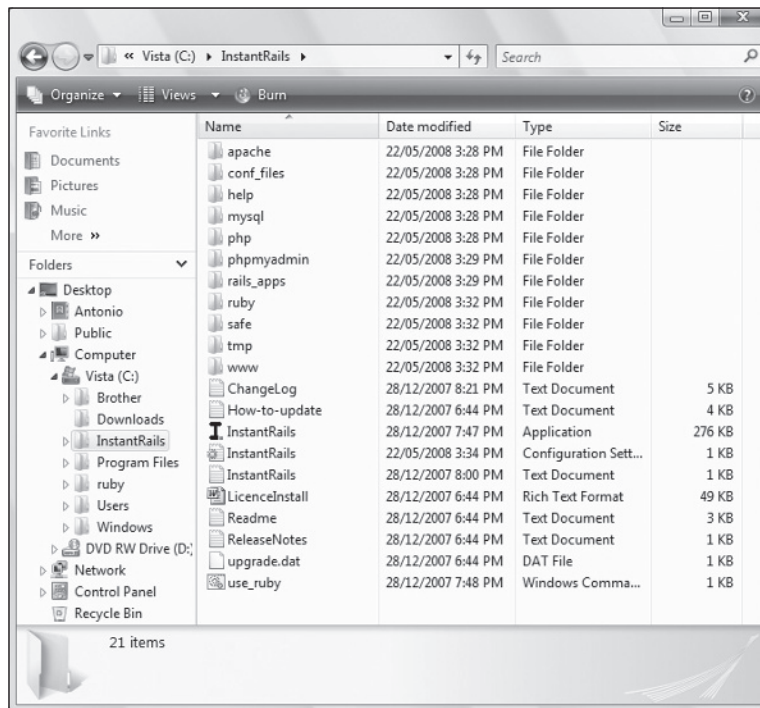


Figure 1-4

3. Double-click the InstantRails application and accept the configuration regeneration, as shown in Figure 1-5.



Chapter 1: Getting Started with Rails

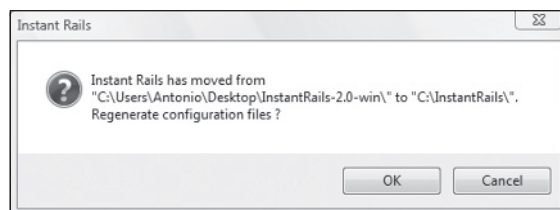




Figure 1-5

- 4 The Instant Rails management application now appears and attempts to start Apache and MySQL for you. The Windows Firewall may ask you to allow this. Feel free to keep blocking Apache, given that the combination of Apache and `phpMyAdmin` is not used throughout this book. You can even stop both services by selecting Stop (or Kill, if stop is not available) in the menu that appears after clicking respectively on the MySQL and the Apache buttons. Configuration and startup settings are available by clicking the  button within the interface, and selecting the (previously concealed) menu Configure  Instant Rails as shown in Figure 1-6.

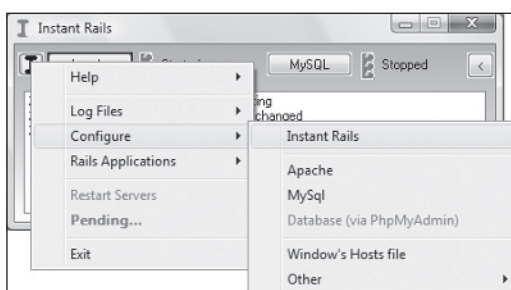




Figure 1-6

As per the “manual” installation through the One-Click installer, it is a good idea to update the installed gems, by issuing a `gem update`. In this case though, `gem` is not accessible from any command prompt, so you’ll have to open the Ruby console from Instant Rails, by clicking Rails Applications  Open Ruby Console Window.

You can also create and manage your applications by selecting Rails Applications  Manage Rails Applications. The Rails Applications window will appear. To create a new application, you’d click the Create New Rails App button. What this does is simply open a command prompt, which is aptly located in (depending on the folder you picked in the beginning) `c:\InstantRails\rails_apps`.

These brief instructions should get you started on the right foot if you were not able to proceed with the recommended method in the previous section. Please note, however, that throughout the rest of the book, you won’t find instructions that are specific to Instant Rails.

Installation on Other Platforms

This section is just a quick pointer in case you decide to get started with Rails on *nix systems rather than Windows.



Chapter 1: Getting Started with Rails

Mac OS X

Apple has been shipping Ruby on its OS for some time now. The default Ruby installation on systems older than Mac OS X 10.4.6 doesn't work well with Rails. If this scenario applies to your system, you have a few options besides upgrading your OS.

You can download (<http://www.ruby-lang.org/en/downloads/>) and then compile and build from source code, if you are familiar with the process. If you're brave, this guide leads you through the installation step-by-step:

```
http://hivelogic.com/articles/2005/12/ruby_rails_lighttpd_mysql_tiger
```

You can also simply use one of the popular distribution systems like MacPorts (<http://www.macports.org>) or Fink (<http://www.finkproject.org>). You should install the whole stack that I pointed out in the Windows section in order to successfully follow this book and start working with Rails projects.

The two preceding methods are only needed if you are running a Mac OS X version that's older than 10.4.6. But it's not uncommon for developers to install their own customized stack rather than to rely on Apple's copy.

A much easier alternative for beginners is Locomotive (<http://locomotive.raaum.org>), a popular project with similar aims (and arguably caveats) to those of Instant Rails, but targeted for Mac OS X.

If you are running Tiger, with a version number equal or greater than 10.4.6, your Ruby installation is good and you'll just need to update RubyGems, the pre-installed gems, and then proceed with the installation of Rails. Conversely, on Leopard (Mac OS X 10.5.x), Rails, Mongrel, and a series of other goodies are already included for you and you'll just need to update them to the latest version. In both cases, you can run the following commands from the Terminal:

```
sudo gem update -system
sudo gem install rails
sudo gem install mongrel
sudo gem update
```

GNU/Linux

If you are using GNU/Linux, you're aware of the many variants (aka distributions, or distros for short) available. It would be impossible for me to provide instructions on how to proceed for each of them. Chances are that no matter what package management system your GNU/Linux version adopts, Ruby 1.8.6 or later, RubyGems, SQLite3, and SVN will be around for you to work with. From RubyGems you can do the rest by updating RubyGems itself, installing (or updating) Rails and Mongrel (and optionally sqlite3-ruby if not available through the package management of your distro).

One of the most common distributions nowadays is Ubuntu (and its variants like Kubuntu and Xubuntu). As an example, I've provided the following instructions on how to set up a Rails stack on version 8.04 of K/X/Ubuntu:

```
sudo apt-get update
sudo apt-get install build-essential
sudo apt-get install subversion
sudo apt-get install ruby-full rubygems libsqlite3-dev libsqlite3-ruby1.8
```



Chapter 1: Getting Started with Rails

```
sudo gem update --system
sudo gem install rails
sudo gem install mongrel
sudo gem update
```

You'll also have to add Gem's bin folder to your `PATH`, by adding the following line (or a similar one, if you're not using the default BASH shell) in your shell profile (for example, in `~/.bashrc`):

```
export PATH=$PATH:/var/lib/gems/1.8/bin
```

By the time you read this book, things may have changed, but several people have reported a few issues with the RubyGems version installed through `apt-get`. Alternatively, you could remove `rubygems` from the preceding instructions and install it manually by downloading and building its source code (http://rubyforge.org/frs/?group_id=126) before proceeding with the last four instructions.

RubyStack

BitNami produces several open source multi-platform installers for popular development stacks. Among these, RubyStack installs Ruby, RubyGems, Rails, ImageMagick, Subversion, SQLite3, MySQL, Apache, PHP, and `phpMyAdmin`. Their installer works with Windows, GNU/Linux, and Mac OS X and, unlike Instant Rails, performs an actual installation and provides scripts to manage the various servers installed.

If you're struggling with alternative installation methods, you may want to give RubyStack, available at <http://bitnami.org/stack/rubystack>, a try.

Editors and IDEs

Talking about editors is like entering a minefield. As the Latin would say "De gustibus non est disputandum," which can be liberally translated as, "There's no arguing with taste." Editor and IDE preferences are highly personal, and I will therefore refrain from telling you which one you should use. I do, however, provide a list of a few popular choices, and I invite you to try them out at your leisure, until you find the one that's right for you.

IDEs Are Helpful, Not Necessary

Microsoft evangelists may occasionally use Notepad in their demos, but as a Microsoft developer you know all too well that without a serious IDE like Visual Studio, writing ASP.NET or .NET desktop applications would be a nightmare. Your proficiency in Visual Studio may make or break your productivity.

In the Rails world there isn't an official IDE and we're not big fans of drag-and-drop tools either. The truth of the matter is that Rails doesn't need either of these two things. Most Rails developers are very happy about using text editors. As a matter of fact, when programming in Ruby or in Rails, a good text editor is all you really need. Even something as simple as SciTE, which we installed through the One-Click Ruby Installer, is probably sufficient.

C# and Visual Basic are both relatively verbose languages that take advantage of a huge framework. Ruby is so expressive and concise that an IntelliSense system like the one you are used to in Visual Studio would be helpful, but it is not required to the same extent.



Chapter 1: Getting Started with Rails

Another great feature provided by IDEs is their ability to do the drudge work for you by generating a lot of code. This, again, is not necessary in Rails. The generation of controllers, models, or scaffold (aimed at rapid prototyping) to name but a few, are all carried out by simple scripts that you can run from the command line. And the generated code could be written (even in Notepad) in very little time. When dealing with Rails there simply isn't a need for a lot of code generation of the sort you'd expect with some other languages and frameworks.

Don't use Notepad as your editor. There are issues associated with the use of the Windows' end of line characters, which would cause trouble when deploying or committing your code to a non-Windows machine. If you must, at least consider Notepad++ (<http://notepad-plus.sourceforge.net>).

IDEs help with the compilation of software, but Ruby is interpreted and, as such, the process is already much more straightforward. You write the code and then run it directly, plain and simple.

That said, IDEs can be helpful even when programming in Rails. But they are usually much lighter in terms of features (and responsiveness) when compared to Visual Studio and similar programs. A good Rails IDE needs first and foremost to have a solid editor. Syntax highlighting and auto indentation of the code are, in my opinion, a must. Not so much with Ruby, but Rails has a tendency to use very long method names, which can become quite tedious to type. An editor that enables fast typing and proposes contextual automatic completion of code can make your life easier.

I find that the most successful Rails IDEs (and some editors) tend to provide quick access to files and folders within your project, given that multiple folders and files are created when Rails generates the skeleton of an application. IDEs should also allow you to run shell commands directly within the environment, so that you aren't forced to switch between the editor and the command line. Some IDEs also provide support for refactoring, testing, and debugging/profiling of both Ruby and JavaScript.

Popular Choices

In an informal survey — admittedly not one of an overly scientific nature — performed by Tim Bray, the Ruby and Rails communities were asked to provide information about what tools they were using. I'm reporting the results of Bray's survey here because they definitely confirm the anecdotal evidence that I've gathered over the past few years in the Rails world.

Figure 1-7 shows a chart with the most popular choices for Rails, according to the survey available online at <http://www.tbray.org/ongoing/When/200x/2007/11/26/Ruby-Tool-Survey>.

Of the first 1,000 programmers who responded to the survey, 38.30% replied that TextMate was their Rails editor of choice. As you can see, text editors are very popular; in fact TextMate, Vi, Emacs, gedit, SciTe, jEdit, and E Text Editor are all editors. NetBeans, Eclipse, and IntelliJ are IDEs. ActiveState Komodo represents a mix of both, given that there is an Edit version and a commercial IDE. From these numbers, let's briefly consider the most popular choices.

TextMate: The King of Rails Editors

TextMate (<http://macromates.com>) is not an IDE, but a fast and extendable editor that is both very powerful and easy to use. It is a commercial product, but it's fairly inexpensive at about \$63 US. TextMate's strength lies in its ability to employ user-contributed bundles, which are very easy to create and modify. It ships with a Ruby and a Rails bundle, among others, and new contributions, with improved features created by other open source developers, are often shared with the community.



Popular Rails Editors

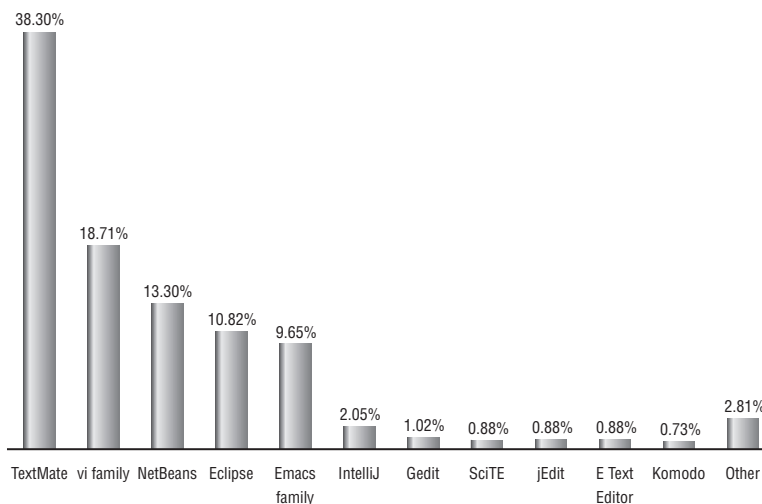


Figure 1-7

TextMate has a Drawer that can contain your project folder and file structure, multi-tabs, and the ability to quickly access a file by simply pressing a shortcut and typing the start of the file name. Among many useful features is the ability to trigger snippets of code (often, common idioms), which can save a lot of time and help you with Rails' long method names (for example, `validates_uniqueness_of`). The catch is that it's only available for Mac OS X.

E Text Editor (<http://www.e-texteditor.com>), used by less than 1% of the surveyed sample, is a clone of TextMate for Windows and its tagline is quite aptly "The Power of TextMate on Windows." It's cheaper than TextMate at about \$35, and can take advantage of TextMate's bundles (see Figure 1-8), but you'll need Cygwin, a Linux-like environment for Windows, in order to get the best from its many features.

Cygwin and the One-Click Ruby Installer

The first thing I do whenever I need to work on a Windows box is to ensure that it has a POSIX emulation layer installed. I do this because I feel greatly incapacitated without being able to take advantage of the many powerful tools available in Unix-like systems.

The E Text Editor encourages you to install Cygwin upon startup and thanks to this you will be able to take advantage of TextMate bundles on Windows. Unfortunately, the native Ruby installed by the One-Click Ruby Installer and the emulated Ruby provided by Cygwin are not compatible. If you installed the One-Click Ruby Installer, and then tried to use Ruby or Rails from within Cygwin, you'd face all sorts of problems.

If you decide to uninstall the One-Click Ruby Installer and use the Cygwin version of Ruby, you will have to manually install RubyGems and then Rails through the `gem` command within the shell provided by Cygwin. Alternatively, by taking the E editor out of the picture, you can use the command-line tools provided by Cygwin (or similarly MinGW), but resist using the Ruby version they provide.

Chapter 1: Getting Started with Rails

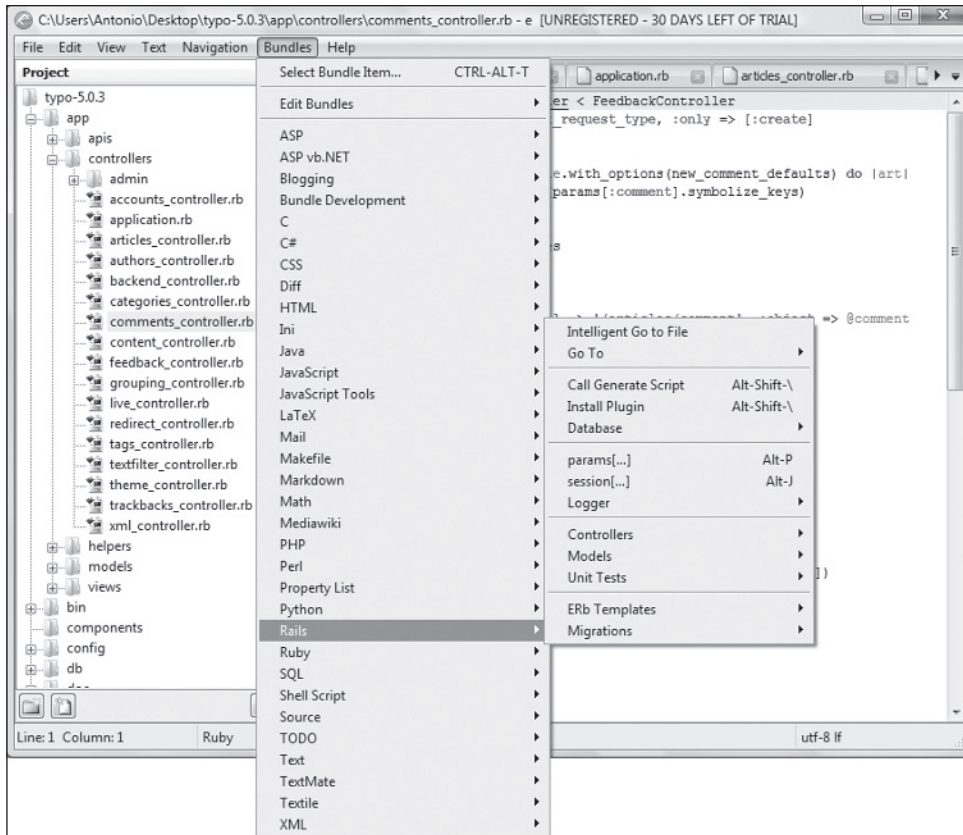


Figure 1-8

TextMate is what I personally use when I'm on a Mac, and I feel that the experience provided by TextMate on a Mac is hard to beat; but given that TextMate is not available for Windows, you may find E to be very useful. Expect it to be very different from what you're accustomed to with large IDEs. After all, it's still a text editor.

Vi and Emacs

Vi and Emacs, and all of their respective variants, are two powerful and very different text editors that have been popular for decades in the *nix world. They're free software and have Windows versions, but if you've never used them before, both will seem extremely complicated to you. In the Rails community Vi is the most used of the two, and in the Ruby community, according to that particular survey, members of the Vi family are even more popular than TextMate.

Some developers swear by either of them, whereas others consider them suboptimal for developing in Rails. It's true that Vi and Emacs are very extendable, powerful, and that there are all sorts of resources for using them for Ruby and Rails development, but unless you come from a Unix background, I don't particularly recommend you check them out. They are worth all the effort that's necessary to learn them, but I don't feel that you need extra obstacles while trying to learn Ruby, Rails, and a different culture.

NetBeans IDE

Sun Microsystems has vested a lot of interest in Ruby and RoR. One of the fruits of its involvement with the community is the NetBeans IDE. This multi-platform Java-based tool has been continuously improved to add new Ruby- and Rails-specific features and, in a short amount of time, it has gathered a growing following.

The tool works well and it even offers code auto completion as shown in Figure 1-9, but don't raise your hopes too high, because currently it's not nearly as refined and efficient as Microsoft's IntelliSense.

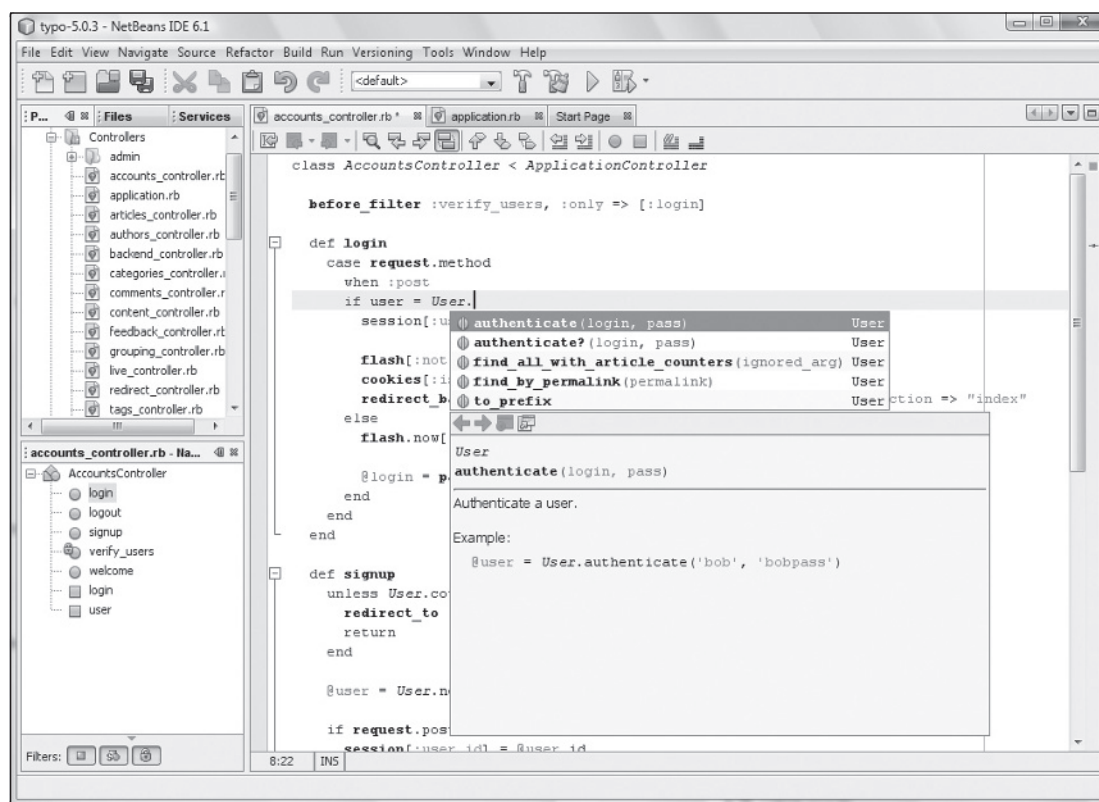


Figure 1-9

Aptana Studio and RadRails

Aptana RadRails is an award-winning open source plugin for Eclipse-based IDEs. It is available as a plugin for Eclipse and can also be installed from within the free-of-charge Aptana Studio, the community edition of a commercial IDE (Aptana Studio Pro). Aptana Studio is based on Eclipse, and once you've installed the RadRails plugin, it provides you with a very pleasant working environment, full of useful Rails features. This product is constantly being improved upon, has excellent support from the company, and it's one of the best liked and most used IDEs in the community.

Chapter 1: Getting Started with Rails

Just like NetBeans, it offers basic support for code completion (see Figure 1-10) and many more features. You can see a feature comparison with NetBeans on Aptana's website (<http://www.aptana.com/rails/>).

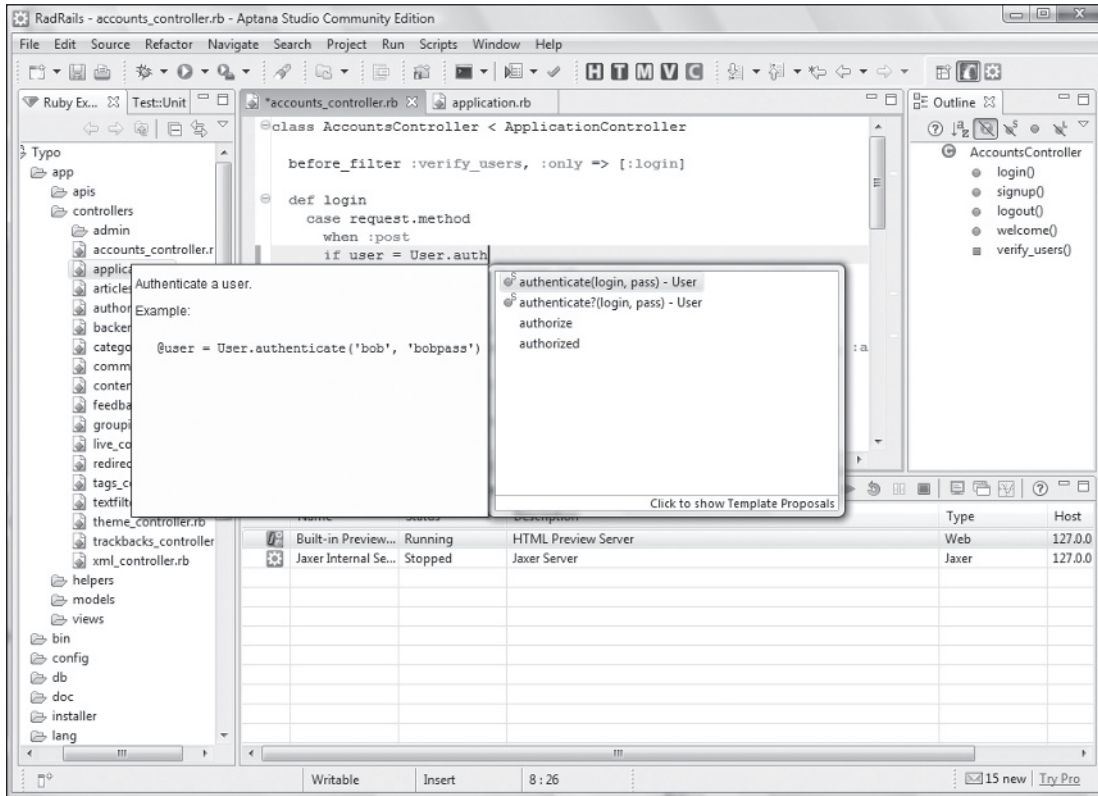


Figure 1-10

Out-of-the-box features aside, its real advantage over NetBeans is that it's part of the Eclipse ecosystem, where many useful plugins are available for Web developers. Of all the editors and IDEs presented here (which work on Windows), Aptana Studio plus the RadRails plugin are definitely one of the most valid choices (and what I use when I'm on a GNU/Linux or Windows workstation).

If you decide to try RadRails, install Aptana Studio (or Eclipse) and then click Install RadRails from the Aptana Start Page.

RubyMine

Currently available as a public preview release, RubyMine is a new Rails commercial IDE by JetBrains. Dubbed by its makers as “The Most Intelligent Ruby IDE,” it’s probably too early to establish the impact that it will have within the Rails community. The Java-based IDEs mentioned previously are quite smart in their own right and are available for free.

Nevertheless, JetBrains has a reputation for creating great IDEs (it makes IntelliJ IDEA) and RubyMine 1.0 is scheduled to be released in the first quarter of 2009. By the time you read this sidebar, it will most likely have been released, and it may be worth checking out.

Ruby In Steel: I Still Want Visual Studio

Ruby In Steel is not on the list of the most common Rails tools. Still, in a book aimed at Microsoft developers, it’s too important to be omitted.

You may spend a week or so trying out several of the recommended IDEs and notice that they all have little in common with the RAD tools you’ve spent most of your professional life working with. You may feel a bit lost, and would prefer to still work with something along the lines of Visual Studio 2008.

If that describes you, Ruby In Steel may be your best bet. It’s not just similar to Visual Studio, it actually is Visual Studio! In fact, Ruby In Steel adds support for Ruby on Rails to Visual Studio 2005 and 2008. As such, this commercial IDE may greatly simplify your transition to the Rails world.

Four downloads are currently available (<http://www.saphiresteel.com/spip?page=download>):

- ❑ **Ruby In Steel Developer 2005:** A plugin for Visual Studio 2005.
- ❑ **Ruby In Steel Developer 2008:** A plugin for Visual Studio 2008.
- ❑ **Ruby In Steel All-in-One Installer:** A complete package that installs Visual Studio 2008 Shell (integrated mode) with the Ruby In Steel Developer edition. This installer can also optionally setup Ruby, Rails, and MySQL for you.
- ❑ **Ruby In Steel Text Edition:** A lightweight entry level edition that can be installed in Visual Studio 2008 (standard and up) or in the Visual Studio 2008 Shell that is installed by the All-in-One Installer (but not with Visual Studio 2005).

Both versions (Developer and Text Edition) have a very fast debugger called “Cylon” and provide support for IntelliSense (as shown in Figure 1-11). The Developer edition also features a Visual Rails WorkBench for drag-and-drop client-side design.

Chapter 1: Getting Started with Rails

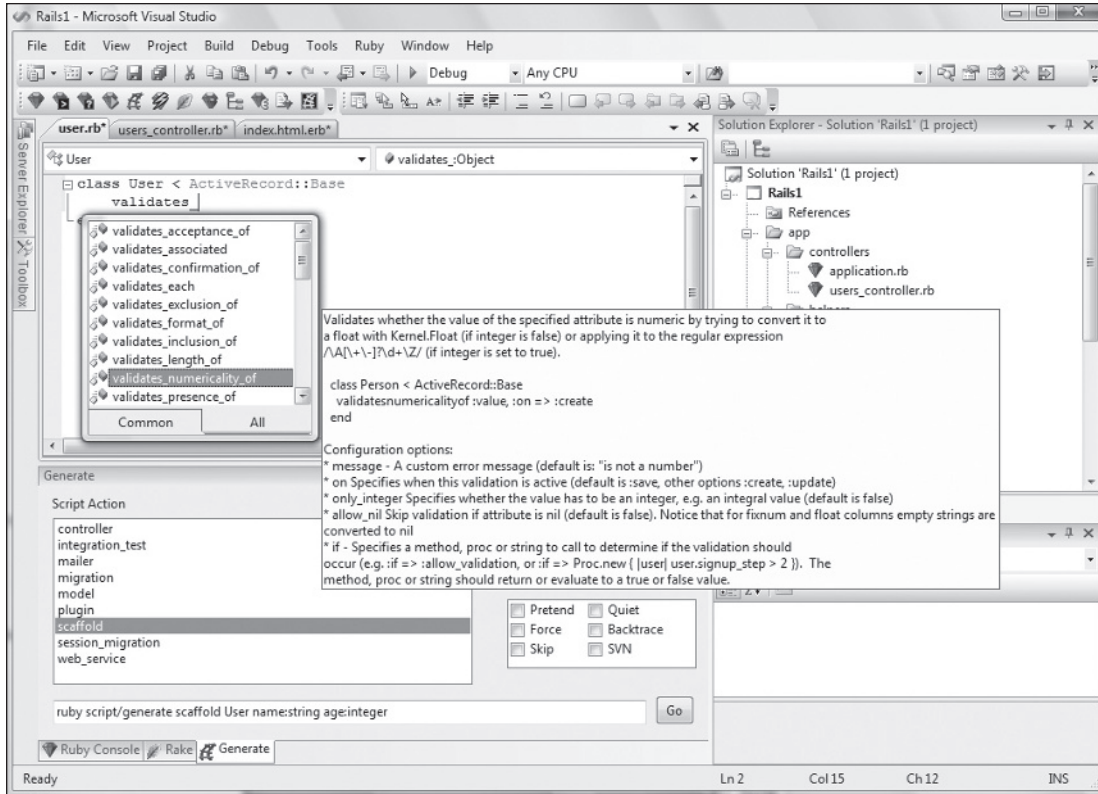


Figure 1-11

Ruby In Steel Developer (version 1.3) currently costs \$199, and Ruby In Steel Text Edition is sold by Sapphire In Steel for \$49. The trial versions will permit you to use them free of charge for up to 60 days. You may not be fond of Visual Studio in the context of Rails, but it's far more likely that you'll enjoy the familiarity of the environment.

Recently the company introduced a free edition known as Ruby In Steel PE 2008, which has most of the useful features offered by the commercial versions, but lacks the integrated debugger (available starting with the Text Edition), auto-expanding of code snippets, and technical support. Though not as useful as the commercial versions, it is still a good starting point nevertheless. You can find it online at <http://www.sapphiresteel.com/Ruby-In-Steel-New-Free-Edition>.

A fourth edition, Ruby In Steel IronRuby Edition — for Microsoft's implementation of Ruby on the Dynamic Language Runtime (DLR) — is free and currently in Alpha. I don't recommend that you install the Alpha version at this stage because neither IronRuby nor its VS plugin are currently suitable for developing Rails applications. This will probably change fast though, so feel free to check IronRuby's progress periodically at <http://www.ironruby.net>.

Spend some time exploring these tools; it's important to find the editor/IDE that you're most comfortable with. No matter what you choose, you'll be able to follow the rest of the book with ease.

Whetting Your Appetite

A “Getting Started” chapter would not be complete without an example to show you how quickly applications can be prototyped thanks to Rails. The aim of this section is truly to “whet your appetite” as opposed to provide extensive explanations of each step. In Chapter 5, after covering the Ruby language in Chapters 3 and 4, you’ll create a more complex Web application and everything will be explained in detail. Here I’m going to provide you with a sneak preview. After all, Rails became so popular thanks also to its ability to create entire applications with just a few commands.

Begin by creating a Rails project that can hold your friends’ addresses. From within a directory of your choice (for example, `c:\projects`), use the command prompt to run the following commands:

```
rails addressbook
cd addressbook
```

This generates an `addressbook` directory that contains the skeleton of your Rails application, and the second instruction makes it the current directory in your prompt.

This being Rails, you’ll use a table within a database to store the data. This table will need fields like the person’s name, address, phone number, and perhaps email and blog/site URL.

You can specify those fields and their data types thanks to the so-called `scaffold` generator, as follows:

```
ruby script/generate scaffold person name:string address:string phone:string
email:string blog:string
```

This will generate a model, controller, and a series of view templates required to provide you with a basic application for performing CRUD operations.

Before you can see what it looks like, you will need to create a development database for the application as well as apply the table definition stored by `scaffold` in a migration file. The two tasks are achieved by a single command:

```
C:\projects\addressbook> rake db:migrate
(in C:/projects/addressbook)
== CreatePeople: migrating =====
- create_table(:people)
-> 0.0660s
== CreatePeople: migrated (0.0680s) =====
```

By convention the development database will be `db/development.sqlite3`; you didn’t have to specify that nor provide a connection string. Also note how Rails is smart enough to figure out that a person’s data should be stored in a `people` table (which is correctly pluralized).

Now that the database is taken care of, start the Web server:

```
ruby script/server
```

The Windows firewall may ask you to unblock the server, which you should agree to.

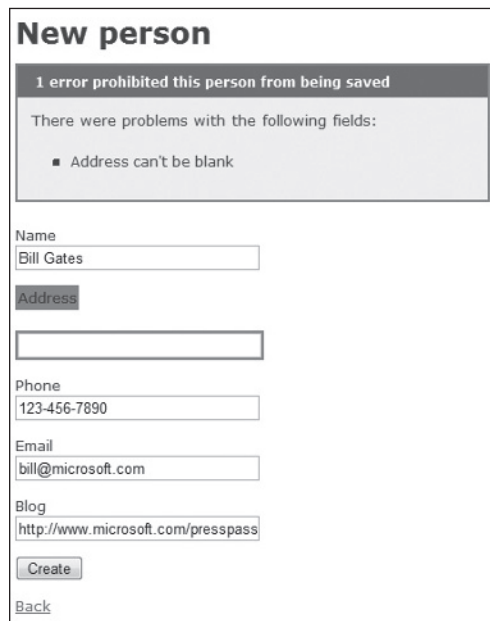
Chapter 1: Getting Started with Rails

Point your browser to `http://localhost:3000/people` and you should see an interface for listing, creating, editing, and deleting entries in your address book. This also incidentally demonstrates Rails' cleverness in handling multiple versus singular terms, because the URL for the application is `http://localhost:3000/people` (where we'd expect to see a list of people). It's remarkable that no code was written to achieve all this.

Would you like to add validations to make the name and address mandatory? And how about ensuring that there are no duplicated names? Just add the following two highlighted lines within your `person.rb` model in the `app/models` directory:

```
class Person < ActiveRecord::Base
  validates_presence_of :name, :address
  validates_uniqueness_of :name
end
```

You now have a fully functioning, database-driven Rails Web application that has complete CRUD functionality. Into the bargain the application validates your input, checking that you have put in a name and address and checking that there are no duplicates (with nice friendly error messages if you get it wrong, as shown in Figure 1-12).



The screenshot shows a web form titled "New person". At the top, a dark grey banner contains the message "1 error prohibited this person from being saved". Below this, a light grey box states "There were problems with the following fields:" followed by a bulleted list: "Address can't be blank". The form fields are: "Name" (filled with "Bill Gates"), "Address" (empty, with a red error message above it), "Phone" (filled with "123-456-7890"), "Email" (filled with "bill@microsoft.com"), and "Blog" (filled with "http://www.microsoft.com/presspass"). At the bottom of the form are "Create" and "Back" buttons.

Figure 1-12

Summary

This first chapter provided a general introduction and overview of Ruby on Rails, highlighting part of its story, philosophy, and relevance to the current Web development world. Step-by-step instructions to set up a complete development environment as well as a quick glance at common editors completed the chapter.

Chapter 2 debunks common misconceptions, explores the concepts behind MVC, takes a closer look at the philosophical principles embraced by Rails, and finally, provides a macroscopic analysis of the differences between this and the ASP.NET world.

