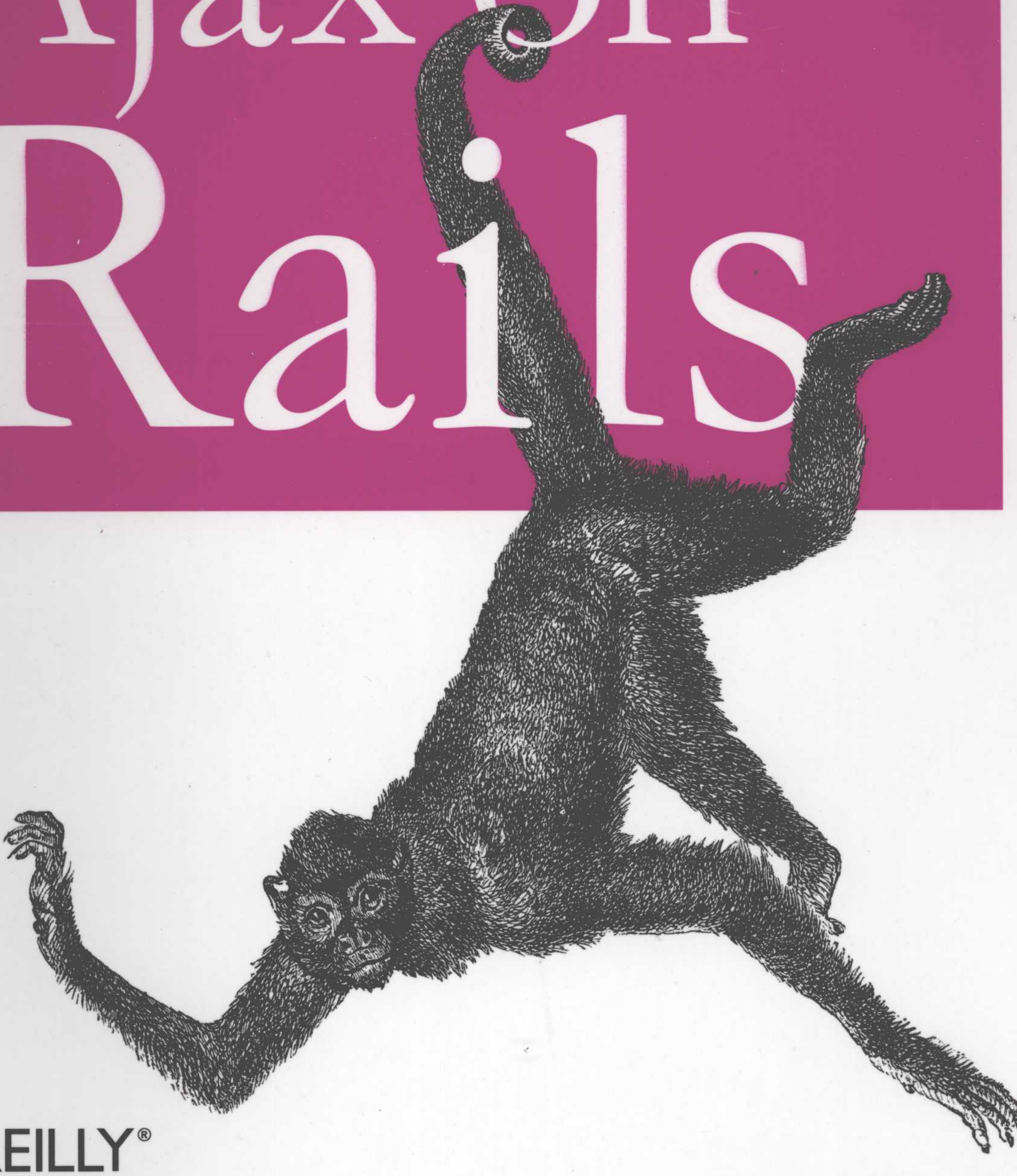
Ajax on Rails（影印版）

Ajax on Rails



O'REILLY®

東南大學出版社

Scott Raymond 著

Ajax on Rails(影印版)

Ajax on Rail

Scott Raymond

O'REILLY®

Beijing • *Cambridge* • *Farnham* • *Köln* • *Paris* • *Sebastopol* • *Taipei* • *Tokyo*

图书在版编目（CIP）数据

Ajax 与 Ruby on Rails=Ajax on Rails:英文 /（美）雷蒙德（Raymond, S.）著 .—影印本 .— 南京：东南大学出版社，2007.6

书名原文：Ajax on Rails

ISBN 978-7-5641-0745-1

Ⅰ . A… Ⅱ . 雷 … Ⅲ . 计算机网络－程序设计－英文 Ⅳ .TP393.09

中国版本图书馆 CIP 数据核字（2007）第 063450 号

江苏省版权局著作权合同登记

图字：10-2007-103 号

书　　名 /	Ajax on Rails（影印版）
责任编辑 /	张烨
封面设计 /	Edie Freedman，马冬燕
出版发行 /	东南大学出版社（press.seu.edu.cn）
地　　址 /	南京四牌楼 2 号（邮政编码 210096）
印　　刷 /	扬中市印刷有限公司
开　　本 /	787 毫米 × 980 毫米　16 开本　22 印张
版　　次 /	2007 年 6 月第 1 版　2007 年 6 月第 1 次印刷
印　　数 /	0001-2500 册
书　　号 /	ISBN 978-7-5641-0745-1/TP · 115
定　　价 /	45.00 元（册）

出版说明

随着计算机技术的成熟和广泛应用，人类正在步入一个技术迅猛发展的新时期。计算机技术的发展给人们的工业生产、商业活动和日常生活都带来了巨大的影响。然而，计算机领域的技术更新速度之快也是众所周知的，为了帮助国内技术人员在第一时间了解国外最新的技术，东南大学出版社和美国 O'Reilly Meida, Inc.达成协议，将陆续引进该公司的代表前沿技术或者在某专项领域享有盛名的著作，以影印版或者简体中文版的形式呈献给读者。其中，影印版书籍力求与国外图书“同步”出版，并且“原汁原味”展现给读者。

我们真诚地希望，所引进的书籍能对国内相关行业的技术人员、科研机构的研究人员和高校师生的学习和工作有所帮助，对国内计算机技术的发展有所促进。也衷心期望读者提出宝贵的意见和建议。

最新出版的影印版图书，包括：

- 《深入浅出面向对象分析与设计》（影印版）
- 《Ajax on Rails》（影印版）
- 《Java 与 XML 第三版》（影印版）
- 《学习 MySQL》（影印版）
- 《Linux Kernel 技术手册》（影印版）
- 《Dynamic HTML 权威参考 第三版》（影印版）
- 《ActionScript 3.0 Cookbook》（影印版）
- 《CSS:The Missing Manual》（影印版）
- 《Linux 技术手册 第五版》（影印版）
- 《Ajax on Java》（影印版）
- 《WCF Service 编程》（影印版）
- 《JavaScript 权威指南 第五版》（影印版）
- 《CSS 权威指南 第三版》（影印版）
- 《嵌入式系统编程 第二版》（影印版）
- 《学习 JavaScript》（影印版）
- 《Rails Cookbook》（影印版）

Preface

This book is for web developers wanting to master two of the most promising recent developments in the field: Ajax and Ruby on Rails. By the end of this book, you'll be equipped with the knowledge to build richly interactive web applications with Rails.

Assumptions This Book Makes

This book assumes that you're familiar with the basic technologies used in building dynamic web sites, on both the client and server sides.

On the client slide, that means HTML/XHTML (which, for the purposes of this book, will be considered equivalent) and CSS. Extensive JavaScript knowledge isn't required, but you'll be well served by a refresher on JavaScript syntax.

On the server side, no specific language experience is assumed, but some grasp of the basic concepts is. If you have experience building web applications in a language like PHP, Java, or ASP, you'll have no trouble understanding the concepts behind Ruby on Rails. But, because this book doesn't cover everything there is to know about Ruby and Rails, you'll want to augment it with other resources—such as those recommended in Chapter 1.

Contents of This Book

This book can be roughly divided into three major parts, plus three complete example applications. The first part introduces all the tools and techniques of Ajax on Rails development, in a fairly linear fashion, from soup to nuts. The second part takes on a handful of larger themes (e.g., usability, security, testing) and provides an in-depth guide to each, in the context of Rails and Ajax. The third part is a comprehensive reference to Rails' two core JavaScript libraries, Prototype and script.aculo.us.

The first part, encompassing Chapters 1 through 5, is a tutorial. Each chapter builds on the previous, and each chapter balances theory and practice. Chapter 1 starts from scratch—installing Ruby and Rails, introducing the fundamental concepts of Ajax development, and providing the context and rationale for the rest of the book. In Chapter 2, the idea is to take a walking tour, in baby steps, through some really simple Ajax examples. Rails provides a powerful suite of shortcuts for Ajax development. But to get the most out of them, it's essential to understand the "long" solution first; that's exactly the approach taken in Chapter 2. Chapters 3 and 4 introduce the shortcuts (Rails' helper methods), which are the workhorses of the Rails way. Lastly, Chapter 5 is the guide to the crown jewel of Ajax on Rails: RJS.

In the second part, we step back from the tutorial format and look at larger themes of professional web development. Chapter 6 deals with usability, cross-platform development, and how Ajax techniques relate to those problems. Chapter 7 covers logging, testing, and debugging. Chapter 8 is on security—always a consideration in web application development, especially when handling financial or other sensitive information. Performance and scalability are covered in Chapter 9. Snappy performance is often the most obvious benefit of Ajax—but that doesn't mean performance issues don't arise.

The third part, Chapters 10 and 11, shifts into reference format. First up is Prototype, one of the most popular and elegant JavaScript libraries. Chapter 10 comprehensively tackles each method that Prototype provides. Chapter 11 covers script.aculo.us, in the same fashion—primarily reference, with generous examples. Both Prototype and scriptaculous are central to Ajax in Rails, but they are also commonly used outside Rails. So these chapters are a valuable reference even if you're building Ajax applications in another server-side language.

Sometimes, the best way to master new technology is to go straight to the source. So the book ends with three complete, professionally designed example applications, each showcasing different Ajax techniques in the context of a real application.

Conventions Used in This Book

The following typographical conventions are used in this book:

Plain text
: Indicates menu titles, menu options, menu buttons, and keyboard accelerators (such as Alt and Ctrl).

Italic
: Indicates new terms, URLs, email addresses, filenames, file extensions, pathnames, directories, and Unix utilities.

`Constant width`
: Indicates commands, options, switches, variables, attributes, keys, functions, types, classes, namespaces, methods, modules, properties, parameters, values, objects, events, event handlers, XML tags, HTML tags, macros, the contents of files, or the output from commands.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values.

This icon signifies a tip, suggestion, or general note.

This icon indicates a warning or caution.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Ajax on Rails* by Scott Raymond. Copyright 2007 O'Reilly Media, Inc., 978-0-596-52744-0."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

We'd Like to Hear from You

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

http://www.oreilly.com/catalog/9780596527440

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, conferences, Resource Centers, and the O'Reilly Network, see our web site at:

http://www.oreilly.com

Safari® Enabled

When you see a Safari® Enabled icon on the cover of your favorite technology book, that means the book is available online through the O'Reilly Network Safari Bookshelf.

Safari offers a solution that's better than e-books. It's a virtual library that lets you easily search thousands of top tech books, cut and paste code samples, download chapters, and find quick answers when you need the most accurate, current information. Try it for free at http://safari.oreilly.com.

Acknowledgments

First, I'm honored to have Sergio Pereira's contribution of Chapter 10—it's a tremendous boon to the book.

If not for my wife's tireless encouragement and valuable suggestions, I'd still be writing this—thank you, Brooke! I'm very grateful to the rest of my family, especially my parents, Doug and Katy. I'm also indebted to my editor, Michael Loukides, an invaluable guide through the process of writing this book. Thanks to Derek Di Matteo for his adept copyediting.

Thank you to these technical reviewers, whose expertise and attention to detail shaped the book significantly: John Aughey, Trey Bean, Jeremy Copling, Kevin Eshleman, Cody Fauser, Brian Ford, Thomas Fuchs, Erik Kastner, Thomas Lockney, Marcel Molina Jr., Tim Samoff, Brian Spaid, Sam Stephenson, and Bruce Williams.

Thanks to the Rails core team and all those who've contributed to Rails, Prototype, and script.aculo.us.

Lastly, thanks to Kansas City's fine coffee houses that supported this project with espresso and Wi-Fi: Broadway Café, Latté Land, and The Roasterie.

Table of Contents

CHAPTER 1

Introduction

Where, where lieth the fatally named,
intractable Ajax?
—Sophocles

Purely in terms of buzz, two of the hottest web-development terms in recent memory are *Ajax* and *Rails*. *Ajax* was just coined in February 2005, and seemingly overnight it sparked summits, workshops, books, and articles aplenty. At the beginning of that year, Rails was still a newborn getting scattered discussion in developers' weblogs. Almost two years later, it claims hundreds of thousands of downloads, nine slashdottings, two conferences, and tens of thousands of books sold.

Why all the noise? Are these technologies fads or worthy of lasting attention?

There are solid reasons to believe that both Ajax and Rails will be significant features of the web development landscape for some time. Big players are leading by example: Yahoo, Google, Apple, Microsoft, and IBM have all started using and touting Ajax techniques, and Rails has become so associated with web startups that it's almost cliché. And for each high-profile implementation, there are dozens created for smaller audiences or for internal use. As awareness of both technologies grows and they prove their value, the snowball will only roll faster.

Ajax on Rails is the definitive guide to where these two technologies converge.

Who This Book Is For

This book will help you use Rails for building richly interactive web applications with Ajax. It provides comprehensive reference and detailed examples for every JavaScript method that Rails offers, as well as its JavaScript-*generating* methods. More than just recipes, you'll also get a thorough, low-level understanding of what's happening under the hood. And beyond the how-to, we'll spend time considering when Ajax is (and isn't) appropriate and the trade-offs associated with it.

This book is written for developers who have experience building for the Web—working knowledge of HTML, CSS, and JavaScript is assumed. Using Rails will require some use of the command line, so you should be familiar with those facilities of your operating system. If you are new to Rails, this book provides a quick introduction, the big picture, a walk through the installation process, and some tips on getting started. But to develop full applications, you'll benefit from a good guide to Ruby itself, as well as the other Rails components. Fortunately, there are many great tutorials and references available online and in print to fill those needs, and we'll point you to the best.

If you have started working with Rails and seek to deepen your skill set, this book will do just that. You'll find dozens of examples drawn from real-world projects, exhaustive reference for every relevant feature, and expert advice on how to "Ajax-ify" your applications.

What Ajax Is

Ajax represents a significant shift in how the Web is built—and even in how it's conceived. But it's a really simple idea: web pages, already loaded in a browser, can talk with the server and potentially change themselves as a result. So instead of a form submission causing a whole new page to load, an Ajax form submission happens in the background and just updates the current page in place—no refresh, no flash of white as the page changes, no change in the address bar. That's the essence of Ajax, in the concrete. It's really that simple! While keeping in mind that simple, concrete definition of Ajax, let's take a minute to look at Ajax in a more abstract way. First, consider how the Web traditionally works.

The Traditional Model

Think about the way the Web usually works, without Ajax. First, the browser creates an HTTP request for something on the server, say */page1.html*. Figure 1-1 shows the life cycle of the request.

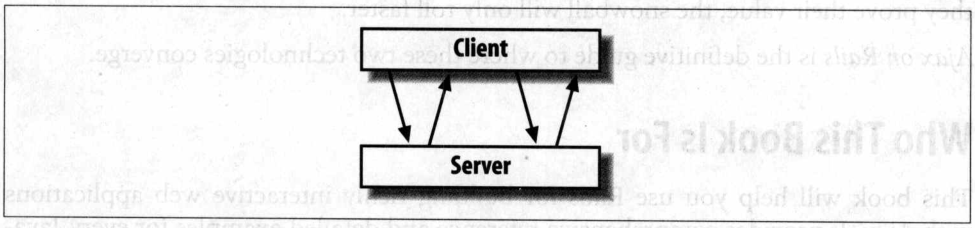


Figure 1-1. The traditional (non-Ajax) request model

In this model, the server sends back a response containing a page—perhaps including a header area with a logo, a sidebar containing navigation, and a footer. With the next click on a link or button, the whole cycle repeats for */page2.html*: a new connection to the server, a new request, and a new page. Even the parts of the page that haven't changed (say, the header and sidebar) are sent over the wire again.

The process of sending the request, waiting for the response, and rendering a new page might take a while, and once the user has clicked, he's effectively committed to that wait before he can proceed.

This model works fine, to a point. In fact, when the nature of your site is primarily document-centric, it's quite desirable. But when developing web applications, it's a bit *heavy*—small interactions that ought to feel responsive are sluggish instead. For example, imagine a web application for managing to-do lists. If simply checking an item off the list causes the entire page to be re-fetched and rendered, the cause and the effect are pretty disproportionate.

The Ajax Model

Remember how simple Ajax is in concrete form: it's just pages talking with the server without a full refresh. With that in mind, contrast the traditional request model with the Ajax model, as seen in Figure 1-2.

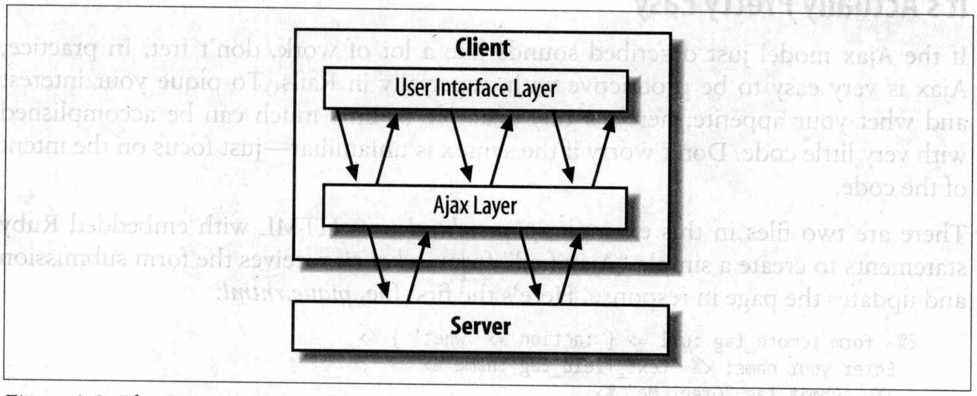


Figure 1-2. The Ajax request model

In the Ajax model, the action on the client side is split into two logical parts—a user interface layer and an Ajax layer. When a user clicks a link, or submits a form, that input is handed to the Ajax layer, which could then interact with the server, and update the UI layer as appropriate.

This is the conceptual cornerstone of Ajax: the UI interaction is logically separated from the network interaction.

There are a few important points to draw from the diagram of the Ajax model:

- The Ajax layer *might not* need to call the server (for example, it might only need to perform simple form validation, which could be handled completely client-side).
- Because the requests between the Ajax layer and the server are for small pieces of information rather than complete pages, there is often less database interaction,

rendering time, and data to transport—making the round-trip time for the request shorter.

- The UI layer is not directly dependent on the server's responses, so the user can continue to interact with a page while activity is happening in the background. This means that, for some interactions, the user's wait time is effectively zero.
- Communication between the page and the server doesn't necessarily imply that Ajax always results in a change to the UI. For example, some applications use Ajax to notify the server about the user's interactions with the page, but don't do anything with the server's response.

These fundamental differences from the traditional request cycle are what enable Ajax applications to be significantly more responsive. And that means that web applications can start to perform like desktop applications—and retain all the benefits of being hosted, rather than installed locally.

It's Actually Pretty Easy

If the Ajax model just described sounds like a lot of work, don't fret. In practice, Ajax is very easy to be productive with, especially in Rails. To pique your interest and whet your appetite, here's a tiny example of how much can be accomplished with very little code. Don't worry if the syntax is unfamiliar—just focus on the intent of the code.

There are two files in this example: *pique.rhtml* uses HTML with embedded Ruby statements to create a simple "Ajaxified" form; *whet.rjs* receives the form submission and updates the page in response. Here's the first file, *pique.rhtml*:

```
<%= form_remote_tag :url => { :action => 'whet' } %>
  Enter your name: <%= text_field_tag :name %>
  <%= submit_tag "Greet Me" %>
<%= end_form_tag %>
<h2 id="greeting" style="display: none"></h2>
```

This code creates a familiar-looking HTML form with one field and a submit button, as well as a hidden HTML heading (see Figure 1-3). When the form is submitted, it will use Ajax to invoke the second file, *whet.rjs*:

```
page[:greeting].hide
page[:greeting].update "Greetings, " + params[:name]
page[:greeting].visual_effect :grow
page.select("form").first.reset
```

These four lines of code pack a wallop—they are instructions telling the page how to update itself. Taking it one line at a time, the instructions are:

1. Hide the element called "greeting" (in case it's not already hidden).
2. Update the element—that is, replace the text inside the tags with some new text.
3. Show it again, animating it onto the screen with a zoom effect.
4. Find the first form on the page and reset it, so that the input field is blank again.

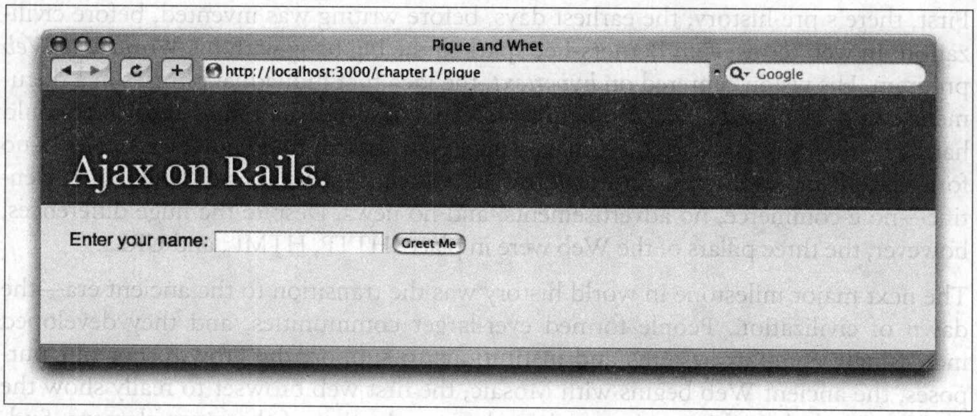


Figure 1-3. A simple Ajax form

The end result after submitting the form is shown in Figure 1-4. Note that the address bar hasn't changed—that's because the page wasn't *replaced* with a new one, it was just *updated* in place.

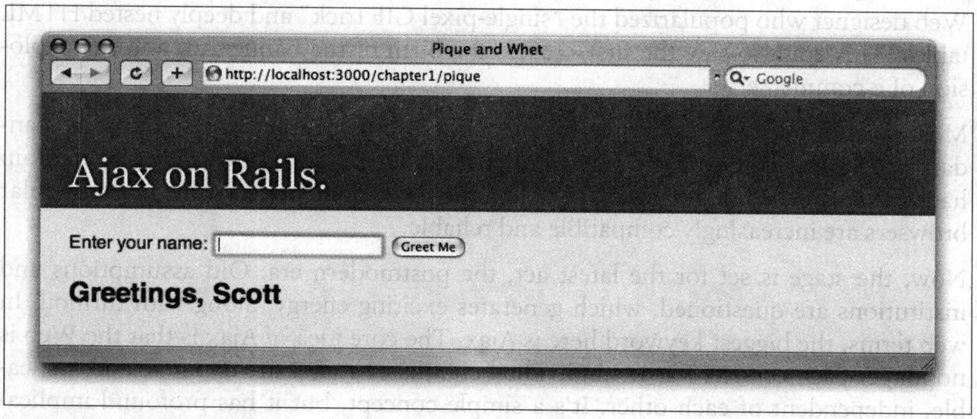


Figure 1-4. After submitting the Ajax form

If you're surprised at how little work is needed to get such impressive results, welcome to Ajax on Rails.

The Eras of Web Development

The web has only been a mass phenomenon since about 1995, so for many developers, it's not hard to remember how we got here. Still, in order to understand the significance of Ajax, it's valuable to look back at the big themes. At the risk of being overly grand, let's compare the history of the Web to the history of the world. Historians organize time into a handful of eras—long periods with distinctive, defining characteristics. With a bit of hyperbole and broad-brushing, the same divisions can be used to understand the eras of web development.