Embedding Perl in HTML with Mason

Mason is a powerful templating system for serving dynamic content on the Web. Based on the Perl programming language, Mason allows you to embed Perl into any text-based format. Mason is most commonly used to build dynamically generated web sites but can also be used to customize configuration files, manipulate XML documents, generate form letters, serve PDF files, or any other task that can be easily simplified with templates.

*Embedding Perl in HTML with Mason,* written by members of Mason’s core development team, shows you how to use Mason’s many features and avoid the obstacles that inexperienced users may encounter. Mason’s strength is in its use of *components,* which are sharable and reusable among Mason documents. When you edit a shared component, you instantly change all documents that refer to it. This can make Mason extremely powerful but requires that you’ve designed your site to take full advantage of it.

This book shows you how to create large, complex, dynamically driven web sites that are remarkably simple to maintain. It shows how to set up a Mason site and configure it properly. It explains the design of Mason, the Mason API, and how to use Mason with CGI or mod_perl. The book also includes a chapter of Mason-based “recipes” with real-world examples of how things are done.

Using Mason isn’t difficult, but using it effectively takes some planning. Mason is so-named because components are like building blocks, and as in masonry, there is an art to knowing how to put them together to create a stable web site. Instead of approaching a Mason-based site as a set of disparate files and scripts, you can build a more effective site using Mason’s support for large-scale structural organization.

With Mason’s unique features, you can streamline the design of any web site or application. Mason is an essential tool for any Perl programmer who wants to simplify their web-site design.

“... the reference sections and copious recipes in this book help to fulfill Mason’s original mission: to provide a fun, simple, and powerful environment for mod_perl development.”

—Jonathan Swartz, creator of Mason
(From the Foreword)
Embedding Perl in HTML with Mason
# Table of Contents

**Foreword** ................................................................. ix

**Preface** ...................................................................... xi

1. **Introduction** ......................................................... 1  
   A First Example ................................................... 2  
   The Main Features of Mason ............................. 2  
   Alternatives to Mason ...................................... 6  
   Philosophy ........................................................ 14  
   Getting Started with Mason ............................ 14

2. **Components** .......................................................... 18  
   Mason from 10,000 Feet ....................................... 18  
   Core Concepts ................................................... 19  
   Basic Component Syntax .................................. 20  
   Component Arguments ..................................... 32  
   Component Return Values ................................. 36  
   Special Globals ............................................... 37  
   Sample Component .......................................... 37

3. **Special Components: Dhandlers and Autohandlers** ............... 39  
   Dhandlers .......................................................... 39  
   Autohandlers .................................................. 42  
   Using Autohandlers and Dhandlers Together .... 49

4. **APIs** ...................................................................... 50  
   Request Class and Object API ......................... 50  
   Component Object API ................................... 64  
   Buffers ............................................................ 68
<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating a CGI-Based Site in Mason</td>
</tr>
<tr>
<td>Using Mason Templates Inside Regular CGI Scripts</td>
</tr>
<tr>
<td>Differences Between Mason Under CGI and mod_perl</td>
</tr>
<tr>
<td>10. <strong>Scalable Design</strong></td>
</tr>
<tr>
<td>Modules Versus Components</td>
</tr>
<tr>
<td>Components as Independent Units</td>
</tr>
<tr>
<td>Component Layout</td>
</tr>
<tr>
<td>File Naming and Directory Layout</td>
</tr>
<tr>
<td>Random Advice</td>
</tr>
<tr>
<td>11. <strong>Recipes</strong></td>
</tr>
<tr>
<td>Sessions</td>
</tr>
<tr>
<td>Making Use of Autoflush</td>
</tr>
<tr>
<td>User Authentication and Authorization</td>
</tr>
<tr>
<td>Co-Branding Color Schemes</td>
</tr>
<tr>
<td>Developer Environments</td>
</tr>
<tr>
<td>Using Mason Outside of Dynamic Web Sites</td>
</tr>
<tr>
<td>12. <strong>Custom Mason Subclasses</strong></td>
</tr>
<tr>
<td>Class::Container as a Superclass</td>
</tr>
<tr>
<td>Syntax: Your Very Own Lexer</td>
</tr>
<tr>
<td>Output: Compiling to a Different Output</td>
</tr>
<tr>
<td>Storage: Replacing the Resolver</td>
</tr>
<tr>
<td>Request: A Request Object with a Built-in Session</td>
</tr>
<tr>
<td>Argument Munging: ApacheHandler</td>
</tr>
<tr>
<td>More Reader Exercises</td>
</tr>
<tr>
<td>A. <strong>The Mason API</strong></td>
</tr>
<tr>
<td>B. <strong>Object Constructor Parameters</strong></td>
</tr>
<tr>
<td>C. <strong>Text Editors That Understand Mason</strong></td>
</tr>
<tr>
<td>D. <strong>Content Management with Bricolage</strong></td>
</tr>
<tr>
<td>Glossary</td>
</tr>
<tr>
<td>Index</td>
</tr>
</tbody>
</table>
Foreword

When Dave asked last year if I wanted to cowrite a book about Mason, I mused over the idea for a few hours but ultimately declined. I’ve already written a good deal about Mason in documentation and articles and wouldn’t want the book to become a rehash of my particular point of view. On a more personal level, the thought of creating any text longer than a few pages fills me with trepidation. Given the choice of eight hours of backbreaking Perl coding or an hour of writing, I’d take the code any day.

This book was authored, instead, by two Mason developers who have a fresh perspective and who, apparently, like writing as well as coding. Dave and Ken have been involved with Mason for years; they’ve contributed to the project immeasurably, first in discussions and later to the Mason core itself. I cannot imagine two people better suited for this book.

When authors have unfettered access to the CVS repository, as Dave and Ken do, the writing process ends up shaping the product itself (e.g., “I just wrote about this feature but it turns out it doesn’t exist—I’m going to implement it” or “I just wrote about this feature and it’s horrible, can we kill it?”). As a result, Mason has grown over the past year in ways I never would have predicted.

Read Chapter 12, for example, to see the new parsing/compiling infrastructure so insanely flexible that one can compile Mason into Embperl or replace Mason’s syntax with an XML variant. (Consider the original parser, a single 371-line subroutine, and you get an idea of how far we’ve come.)

Or Chapter 5 to see how “Calling Components with Content Blocks” turns content filtering from a limited hack into a first-class language feature and opens the door to the creation of custom language tags.

Or Chapter 9 to see how support for CGI (and its high-powered cousins FastCGI and SpeedyCGI) has gone from a few paragraphs in the FAQ to a full-fledged handler, inside a new modular architecture, ready for additional platforms.
Such changes expand Mason’s reach to an ever larger set of problems. At the same time, the reference sections and copious recipes in this book help to fulfill Mason’s original mission: to provide a fun, simple, and powerful environment for mod_perl development. Should we stray too far from this path, we will certainly hear of it from the users.

Reading this book has taught me a surprising number of techniques about a product I used to know everything about and has reignited my enthusiasm for building cool web applications. I hope it does the same for you.

—Jonathan Swartz
swartz@pobox.com
Preface

Mason is a tool for embedding the Perl programming language into text, in order to create text dynamically, most often in HTML. But Mason does not simply stop at HTML. It can just as easily create XML, WML, POD, configuration files, or the complete works of Shakespeare.

Mason was originally written by Jonathan Swartz, with the help of the rest of the CMP development team at CMP Media in 1996, and in its earliest incarnations it was known as Scribe.

Mason was first made publicly available as Version 0.1 in August of 1998. Since that time, Jonathan Swartz has invited your humble author to participate in the further development of Mason. Mason has been expanded, and rewritten and is much changed from those early days. This book covers Version 1.12 of Mason.

Intended Audience

This book assumes that the reader is familiar with Perl at an intermediate level and that common Perl idioms don’t stop you in your tracks. While you need not have written your own modules previously, familiarity with Perl’s object-oriented syntax will be helpful.

Since Mason is most often used in the generation of web sites, this book frequently presents its example in terms of generating HTML pages. You definitely do not need to be an HTML expert to read this book, but a passing familiarity with HTML will be helpful in understanding what the output is intended to look like.

As previously mentioned, Mason is well-suited for the generation of any sort of dynamic text, including markup languages such as XML, as well as configuration files, email bodies (think mail merge, perhaps), or even code.

Finally, experience with mod_perl and CGI will be helpful for Chapters 7 and 9, which discuss integrating Mason with mod_perl and CGI, respectively, though the
rest of the book should be easily comprehensible regardless of your familiarity with those particular topics.

**Requirements**

As of Version 1.12, Mason requires Perl Version 5.00503 or later, as well as the following modules available from CPAN: Class::Container 0.07+, File::Spec 0.8+, Params::Validate 0.24+, Exception::Class 1.01+, and Cache::Cache 1.00+.

If you want to run Mason under mod_perl, you will need to have at least Version 1.22 of mod_perl installed (although installing the very latest version in the 1.x series is highly recommended). You’ll also probably want Apache::Request 0.31+, though this is not strictly required.

Some of the examples are designed to work in a web server context and may require either CGI or mod_perl capabilities. For the CGI examples, any CGI-capable web server is sufficient, though our configuration examples always assume Apache.

Mason itself is designed to be platform-agnostic. Any behavior to the contrary should be considered a bug in Mason, and bug reports are always welcome.

In writing this book, we did our testing with Perl 5.6.1 and mod_perl 1.25, with Apache 1.3.22, so it is possible that something shown here may not work as expected with other versions of these pieces of software. If you find this to be the case, please let us know and we’ll either fix the example or make a note of the dependencies.

**How to Read This Book**

This book is not a reference manual. While certain chapters, notably Chapter 4, are reference-like in nature, the book as a whole builds chapter upon chapter and often refers to material covered earlier. Nonetheless, some readers may prefer to look at Chapter 8 early, so an alternate path through the book might be to read that chapter right after Chapter 1.

Once you become more familiar with Mason, we expect that much of the book will still be useful as a reference, particularly chapters 4, 5, 6, and 7.

Appendixes A and B present much of the same material as those chapters in a more streamlined reference style, and should also be useful long after you have become a Mason guru. Appendix C is a short discussion of editors you can use with Mason, and Appendix D covers Bricolage, a content management system based on Mason.

Finally, the book includes a short glossary for those of you who are new to Mason and its terminology.
Overview

Chapter 1, Introduction
This chapter goes into more detail on what Mason is, highlights a few of its most notable features, and also discusses some alternatives to Mason.

Chapter 2, Components
Components are the basic building block in Mason. This chapter covers component syntax and how components can call one another, much like Perl subroutines.

Chapter 3, Special Components: Dhandlers and Autohandlers
The topic of this chapter is dhandlers and autohandlers, two features unique to Mason that really can help you maximize code reuse.

Chapter 4, APIs
This chapter is a bit more of a reference than those before it, as it has a lot of ground to cover. We cover the APIs of various objects that you might want to access from your components, primarily the Mason request object.

Chapter 5, Advanced Features
Now that you know about component syntax, dhandlers, autohandlers, and a good chunk of the Mason API, it’s time to learn about some of Mason’s more complex features, like component methods and attributes, components that filter content, and Mason subrequests.

Chapter 6, The Lexer, Compiler, Resolver, and Interpreter Objects
These objects form the core of Mason itself, and their constructor parameters and object APIs let you customize the way Mason handles requests.

Chapter 7, Using Mason with mod_perl
Mason and mod_perl are designed to play nice together, and this chapter shows you how to make this happen.

Chapter 8, Building a Mason Site
The obligatory sample site. See Mason at work on a real, live, on-the-Internet, usable-by-you web site.

Chapter 9, Mason and CGI
If you can’t use mod_perl, you may still want to use Mason to generate web pages via CGI scripts.

Chapter 10, Scalable Design
Mason is cool, but it’s not magic. You still need to think about architecture and design when designing a Mason-based application. We try to give you a few things to think about here.

Chapter 11, Recipes
This chapter demonstrates a number of different uses of Mason in order to stimulate your thinking, as well as provide some easily customized example code.
Chapter 12, *Custom Mason Subclasses*
   A how-to on creating your own customized Mason subclasses.

Appendix A, *The Mason API*
   All the APIs we covered earlier, in a more concise form.

Appendix B, *Object Constructor Parameters*
   A spiffy table of constructor parameters.

Appendix C, *Text Editors That Understand Mason*
   Information on using Emacs and Vim to edit Mason components.

Appendix D, *Content Management with Bricolage*
   David Wheeler explains how to use Bricolage, a content management system built with Mason that can use Mason to generate web pages.

**Other Resources**

The most important online resource for Mason is the Mason HQ site located at [http://www.masonhq.com/](http://www.masonhq.com/). This site includes an online copy of the documentation, a Frequently Asked Questions database, news, user-contributed code and documentation, and much more. The bug tracker on Mason HQ is kept up-to-date, and it is a good source of information about limitations in certain versions of Mason. There is also a to-do list that can give you insights on future directions planned by the Mason developers.

Mason HQ also provides information on subscribing to the various Mason mailing lists, of which the most important is the *mason-users* list. See [http://www.masonhq.com/resources/mailing_lists.html](http://www.masonhq.com/resources/mailing_lists.html) for more details.

This book’s web site, at [http://www.masonbook.com/](http://www.masonbook.com/), will always contain the latest digital version of the book, freely available. In addition, we will make sure to report any errata found in the published edition(s), as well as providing downloadable versions of all of the source code found in the book.

**Typographic Conventions**

Constant width
   is used for literal text, module names, function names, method names, and keywords

*Italics*
   is used for filenames, components, commands, URLs, emphasis, and terms when they are first introduced
About the Authors

Dave Rolsky has worked as a paperboy, supermarket bagger, temporary secretary, ear-training and music theory-teaching assistant and, every so often, as a computer programmer specializing in Perl. He is an animal rights activist, obsessive reader of the works of Gene Wolfe, Hong Kong film aficionado, and the owner of many black t-shirts.

Dave has been a member of the Mason core team since the summer of 2000, a position he attained primarily by constantly nagging Jon Swartz, Mason's creator. He has written numerous CPAN modules, including Alzabo, Exception::Class, Log::Dispatch, and others. He was never convicted.

Ken Williams enjoys sitting by the park and taking long walks in the fireplace. He has a hard time staying interested in any one thing for very long; he has recently worked as a math teacher, choral conductor, Perl consultant, and liturgical bongoist. He is currently a researcher in Document Categorization at the University of Sydney in Australia.

He has written CPAN modules of varying utility while masquerading under the seedy, secretive pseudonym of “Ken Williams.”

Ken joined the Mason core team in the fall of 2000 and has managed to convince the other core members that his flashy looks and his appeal with Mason's ultra-hep Gen-Z users outweighs his tendency to forget what planet he's on.

Colophon

Our look is the result of reader comments, our own experimentation, and feedback from distribution channels. Distinctive covers complement our distinctive approach to technical topics, breathing personality and life into potentially dry subjects.

The animal on the cover of Embedding Perl in HTML with Mason is a Hamadryas (Arabian) baboon. This species inhabits the dry plains and rocky hills of northern Africa and the Arabian Peninsula. Though their primary diet consists of roots, seeds, and fruit, Hamadryas baboons also eat insects and small animals, including other monkeys. They travel and forage in bands of 50 to 100 during the day and gather in troops as large as 750 to sleep on steep-sided cliffs during the night. All adults have long, dense, silky fur that is gray in males and brownish in females. Mature males weigh an average of 45 pounds and have a silver cape or mane over the head, neck, and shoulders. Females are considerably smaller and have no mane.

Arabian baboons live in a highly developed social order based on harem groups—a single adult male is accompanied by up to four females and their offspring. Males control their family by brute force, often biting females on the nape of the neck. Their powerful canines are also bared to threaten predators, which include leopards, jackals, hyenas, cheetahs, and lions. During an attack, they may yawn, slap
their hands and feet, scream, and alert other baboons with a dog-like bark. They are fierce combatants, often winning fights against animals larger than themselves.

Ancient Egyptian artwork frequently pictures Hamadryas baboons as attendants of Thoth, scribe of the gods, and himself the god of wisdom, learning, and magic. The Egyptians recognized the intelligence of “sacred” baboons and reportedly trained them to wait on tables, pluck weeds from garden plots, and assume positions of prayer when in a temple. They also helped make wine; tomb paintings depict them harvesting grapes and using their own body weight to increase the tension of wine presses. Today these baboons are listed as a threatened species, and they no longer inhabit Egypt. Cultivation and development have destroyed much of their natural habitat and forced some bands to rely on crops and garbage dumps for food.

Philip Dangler was the production editor for *Embedding Perl in HTML with Mason*. Norma Emory was the copyeditor. TIPS Technical Publishing, Inc. provided production services and wrote the index. Emily Quill and Mary Anne Weeks Mayo provided quality control.

Ellie Volckhausen designed the cover of this book, based on a series design by Edie Freedman. The cover image is a 19th-century engraving from *Animal Creation*. Emma Colby produced the cover layout with QuarkXPress 4.1 using Adobe’s ITC Garamond font.

David Futato designed the interior layout. This book was converted to FrameMaker 5.5.6 with a format conversion tool created by Erik Ray, Jason McIntosh, Neil Walls, and Mike Sierra that uses Perl and XML technologies. The text font is Linotype Birka; the heading font is Adobe Myriad Condensed; and the code font is Lucas-Font’s TheSans Mono Condensed. The illustrations that appear in the book were produced by Robert Romano and Jessamyn Read using Macromedia FreeHand 9 and Adobe Photoshop 6. This colophon was written by Philip Dangler.