BOOK REVIEWS

Catalyst

reviewed by Yanick Champoux,
yanick@babyl.dyndns.org

For those who haven't heard of it yet, Catalyst is a swanky (relatively) new web framework, often touted as Perl's answer to Ruby on Rails. One of its greatly appealing aspects is its core philosophy, which is very close to Perl's own. It puts a strong emphasis on code re-use, is engineered for maximal DWIMness and leaves developers an insane amount of flexibility and freedom.

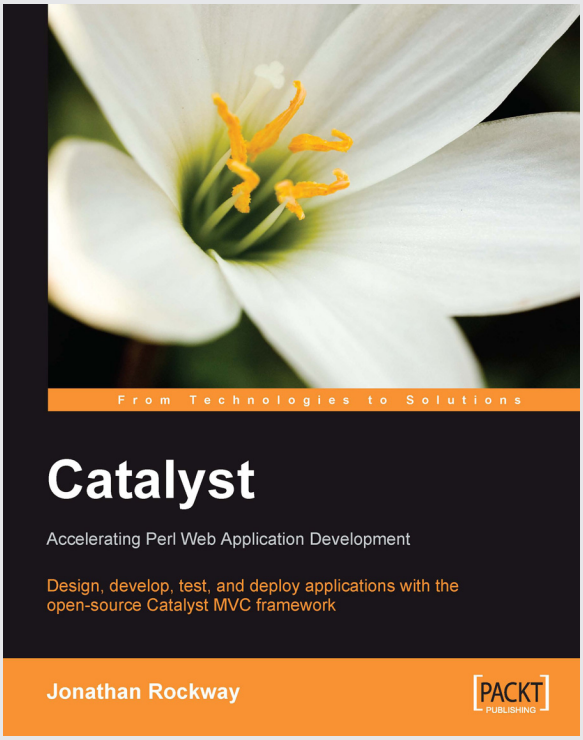
Unarguably, this makes Catalyst a very powerful tool. But it also turns it into a fairly daunting, not to say scary, beast. Not because it lacks documentation, mind you. Its core modules have excellent manpages and many helpful tutorials can be found on Catalyst's official web site (*http://catalyst.perl.org/*) and elsewhere. It's rather Catalyst's sheer scope that makes it hard to tackle. And because of its infamous flexibility, the different approaches of various tutorials and pieces of documentation sometimes make it hard to grasp the big picture. Indeed, for some time now the community has been longing for a simple, yet extensive introduction to Catalyst that would, while maybe not showing all the different ways there are to do it, at least walk the reader through a good, reasonable first one.

I am happy to report that *Catalyst*, by Jonathan Rockway (who, I should note, is a member of the Catalyst core team) might very well be this gentle introduction to Catalyst that we were waiting for.

For a technical book, this one is relatively slim, with nine chapters and 200 pages. But then, it has a very targeted goal: it doesn't aim to be an in-depth look at the inner mechanisms of Catalyst nor an exhaustive description of everything Catalyst and its impressive cohorts of plug-ins can achieve. Rather, it strives to be a first-steps guide that shows enough to someone who is familiar with Perl but never touched Catalyst. It aims to enable them, once the book is read and put down, to build a web application with all the expected modern bells and whistles.

The first chapter is a brief presentation of Catalyst, its design goals (Extensibility, Re-usability, Flexibility and Reliability) as well as the MVC (Model-View-Controller) architecture it is based on. It also provides guidance on how to install Catalyst, and where to go to get further help.

The second chapter walks us through the creation of a Catalyst "Hello World"-type application. The goal being to get our feet wet. Here, as well as in the subsequent chapters, Jonathan does an excellent job of explaining what the code does, but he doesn't spend a lot of time delving on why it must be that way, or what are all the options given to the programmer. The expectation is that once the Catalyst fledgling is armed with the material provided by the book, they'll be able to gather those more intricate references off Catalyst's manpages.



Catalyst
Accelerating Perl Web Application Development
by Jonathan Rockway
Packt Publishing, December 2007
ISBN 978-1-84190-95-6
187 Pages
http://www.packtpub.com/catalyst-perl-web-application/book

For the next 4 chapters, Jonathan leads us through the creation of a few, remarkably well-chosen applications—a web-based community address book, a rudimentary blog system and an IRC opinion harvester and reporter. They are all real-life applications that, with a little cosmetic work, wouldn't be out of place in production, and yet are not so overwrought as to be distracting. Using those examples, Jonathan shows us how to implement most of the typical features we requires when building a web application: configuration, templating, form generation, result paging, searching, session management, authentication, and authorization.

Of course, nowadays web pages are merely the tip of the iceberg. Any self-respecting web application is likely to also have to provide RSS feeds, contain some AJAX spiffiness and include an RPC or REST API. Right on cue, chapter 7 is dedicated to those features and shows how to spike the examples seen in the previous chapters with those.

The last two chapters deal with development details. Chapter 8 presents how to write tests for both the offline details and the online interface of your Catalyst application. And chapter 9 deals, quite appropriately, with the final task left at the end of the development cycle: the deployment of the application. It explains the different ways that the application can be hooked to a website infrastructure (via Apache's mod_perl, or via a CGI or fast_cgi).

Performance considerations are briefly addressed, not deeply enough to make a screaming banshee out of an application, but still enough to make it production-worthy. Unfortunately, there is no mention of how to promote the database engine of an application from SQLite (which is used throughout the book as the development database engine) to one of the "big boys" (MySQL, Postgres, Oracle, etc.). Probably because the switch, once one knows how, is trivial, but a quick mention of it would have been welcome.

To sum it up, this book makes an excellent introduction to the wonderful world of Catalyst. The Llama of Catalyst (CataLlama?), if you will. If you are new to Catalyst, you will love it. If you are a seasoned Catalyst user, chances are that most of the book will feel old-hat to you, although you're still quite likely to find a few new tricks hidden within its pages.

Yanick Champoux is a testing engineer at Alcatel-Lucent, but also recreationally dabbles in web development. In the days before any Catalyst book were published, his quest for a good Catalyst tutorial led him to create `Pod::Manual`.

Catalyst

Catalyst links:

Official website:
http://catalyst.perl.org

Catalyst cheatsheet:
http://refcards.com/refcard/catalyst-forda

Catalyst manual as a PDF:
http://babyl.dyndns.org/perl/pod-manual/catalyst_manual.pdf

The Catalyst modules and plug-ins
http://search.cpan.org/search?query=catalyst&mode=module

Catalyst advent calendars:

```
map
"http://catalyst.perl.org/calendar/$_",
  2005..2007
```

MVC in a Nutshell

The Model-View-Controller architecture divides an application's functionality into three distinct segments.

The Models provide an interface to the underlaying data used by the application, such as database connections, filesystems, LDAP queries, *etc.*

The Views generate output suitable for the end-user out of some data provided by the models. It can be web pages, but can also be pdf documents, XML files, text-to-speech audio tracks, GUI interface, *etc.*

The Controller is the brain of the application, and the mediator between the Models and the Views. It processes events coming from the user and responds by using the appropriate models and views.

This segregation helps reduce the complexity of the application. Models don't have to care about presentation, and Views don't have to worry where the information come from. It also provides flexibility and makes for easy refactoring. As long as they have similar interfaces, one can replace one model by another—say, the initial development SQLite database to an Oracle cluster—without changing a single line of code off the views and controller. The same goes for Views and Controllers.

Not Just for the Web

Most people think of MVC frameworks as web frameworks, but web pages are just one sort of View. Although most frameworks lean heavily toward web applications since that's what most people want, its not the only way. For instance, the `Catalyst::Engine::Wx` module provides a way to use Catalyst as the framework of a WxPerl GUI-based application. You can make desktop applications with Catalyst using the same MVC principles, and while re-using the components you created for the web version.