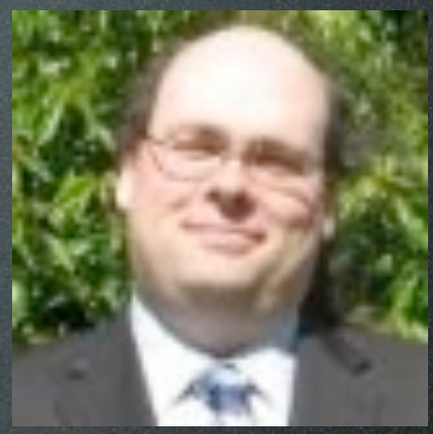
Catalyst

A Powerful MVC Web Framework for Perl

Presentation Info

Louis Erickson
<lerickson@rdwarf.net>



- Programming in Perl for 15 years.
- Currently works at NVIDIA.

Presented on 6 December 2011 for the Silicon Valley Perl user's group.

Thanks to Plug & Play Tech Center for having us tonight!

Introduction

- Introduction
  - What is Catalyst
  - Building a Catalyst App
  - Core Catalyst: Actions
  - Resources

What is Catalyst?

- Model-View-Controller (MVC) Web Framework
- Built of many modules
  - Libraries
  - Plugins
- Perl, duh.

Model-View-Controller

- MVC is a well understood way to structure an application
  - Model: Data model, stores informaiton
  - View: Visualize and/or represent the model and/or results
  - Controller: Implements business logic

Building a Catalyst App

- Install Catalyst
  - Perlbrew and cpanm will be your friends
- cpanm Catalyst::Devel
  - Dependencies usually work!

Create The App

- Catalyst provides a tool to create a new app, catalyst.pl:
- catalyst.pl Rechoarder

```
$ catalyst.pl Rechoarder
created "Rechoarder"
created "Rechoarder/script"
created "Rechoarder/lib"
created "Rechoarder/root"
created "Rechoarder/root/static"
created "Rechoarder/root/static/images"
created "Rechoarder/t"
created "Rechoarder/lib/Rechoarder"
created "Rechoarder/lib/Rechoarder/Model"
created "Rechoarder/lib/Rechoarder/View"
created "Rechoarder/lib/Rechoarder/Controller"
created "Rechoarder/rechoarder.conf"
created "Rechoarder/rechoarder.psgi"
created "Rechoarder/lib/Rechoarder.pm"
created "Rechoarder/lib/Rechoarder/Controller/Root.pm"
created "Rechoarder/README"
created "Rechoarder/Changes"
created "Rechoarder/t/01app.t"
created "Rechoarder/t/02pod.t"
created "Rechoarder/t/03podcoverage.t"
created "Rechoarder/root/static/images/catalyst_logo.png"
created "Rechoarder/root/static/images/btn_120x50_built.png"
created "Rechoarder/root/static/images/btn_120x50_built_shadow.png"
created "Rechoarder/root/static/images/btn_120x50_powered.png"
created "Rechoarder/root/static/images/btn_120x50_powered_shadow.png"
created "Rechoarder/root/static/images/btn_88x31_built.png"
created "Rechoarder/root/static/images/btn_88x31_built_shadow.png"
created "Rechoarder/root/static/images/btn_88x31_powered.png"
created "Rechoarder/root/static/images/btn_88x31_powered_shadow.png"
created "Rechoarder/root/favicon.ico"
created "Rechoarder/Makefile.PL"
created "Rechoarder/script/rechoarder_cgi.pl"
created "Rechoarder/script/rechoarder_fastcgi.pl"
created "Rechoarder/script/rechoarder_server.pl"
created "Rechoarder/script/rechoarder_test.pl"
created "Rechoarder/script/rechoarder_create.pl"
Change to application directory and Run "perl Makefile.PL" to make sure your
install is complete
```

Exploring The App

```
$ ls -F
Changes          lib/          root/
Makefile.PL      rechoarder.conf    script/
README           rechoarder.psgi    t/
```

- Ordinary module:
  - Changes
  - Makefile.PL
  - lib/
  - t/
- Catalyst stuff:
  - root/
  - script/
  - <myapp>.conf
  - <myapp>.psgi

Scripts

```
$ ls script
rechoarder_cgi.pl    rechoarder_fastcgi.pl   rechoarder_test.pl
rechoarder_create.pl    rechoarder_server.pl
```

- _create.pl: Tool to create models, views, and/or controllers
- _cgi.pl, _fastcgi.pl: Tools to run the app under webservers
- _test.pl: unit testing tool
- _server.pl: Testing web server.

root/

- “Document Root” for the webserver
- Contains a static/ for fixed resources
- Often contains templates and components

Writing the Model

- Can be any class, stored in MyApp/lib/Model
- Common to use DBIx::Class, includes a tool:

```
./rechoarder_create.pl model RechoarderDB DBIC::Schema
Rechoarder::Schema create=static
dbi:Pg:dbname=recorder_test recorder_test DB_password
```

Writing the View

- Can be anything needed, many available:
  - TemplateToolkit
  - HTML::Template
  - Excel::Template
  - GD
  - Email
- Tool will write them:

```
rechoarder_create.pl view HTML TT
```

Writing the Controller

- Controllers are the exiting part of the web app.
- Map URL to functions
- Access the Model
- Provide data for the view to render

Actions

- Catalyst controllers use special functions called “actions” to handle URL requests.
- Actions are defined by special attributes.
  - Attributes determine how the action maps to a URL, if it does.

URL Mapping

- Attributes:
- :Private
- :Local
- :Global
- :Path
- :Args
- :Regex
- :LocalRegex
- :Chained

:Private

- A :Private attribute defines a valid action that maps to no URLs.
- Can be useful for redirections and shared code.

:Local

- A :Local attribute defines an action that maps by namespace and subroutine.
- Maps from the start of the URL.

```
package MyApp::Controller::User;

sub foo :Local {
    # Called for http://localhost:3000/user/foo/...
}
```

:Global

- The :Global attribute defines an action that maps from the root of the URL space.
- Maps from the start of the URL.

```
package MyApp::Controller::User;

sub snort :Local {
        # Called for http://localhost:3000/snort/...
}
```

:Path

- The :Path attribute defines an action that contains the specified items in the URL.
- A leading / anchors at the root of the URL space, otherwise inside the controller's namespace.

:Path Examples

- :Path examples:
  - Can be used to create :Local and :Global

```
package MyApp::Controller::User;

sub bar :Path(“/wibble”) {
    # Called for http://localhost:3000/wibble/...
}

sub bing :path(“wobble”) {
    # Called for http://localhost:3000/user/wobble/...
}
```

:Args

- Modifies action to capture the desired items off the URL and add to @_.
- :Args(n) demands exactly n arguments.
- :Args(0) and :Args are different.

```
package MyApp::Controller::User;

sub snort :Local :Args(2) {
    # Called for http://localhost:3000/snort/aaa/bbb/
    # Captures aaa and bbb
}
```

:Regex

- The :Regex attribute defines an action that matches a regular expression.
- Not bound to namespace, can match any part of the URL

```
package MyApp::Controller::User;

sub snort :Regex('item(\d+)/order(\d+)'); {
    # Called for http://localhost:3000/.*item\d+/order\d+.*
}
```

:LocalRegex

- The :LocalRegex attribute defines an action that matches a regular expression.
- Bound to namespace, matching only things after the Controller's namespace.

```
package MyApp::Controller::User;

sub snort :Regex('item(\d+)/order(\d+)'); {
    # URLs: http://localhost:3000/user/.*item\d+/order\d+.*
}
```

:Chained

- The :Chained attribute assigns a function to match part of a path.
- Chained is used in different ways to start a chain, put things in a chain, or end a chain.
- Allows complex code re-use by parsing based on components of a URL.

:Chained Example

```
package MyApp::Controller::Catalog;

sub base :Chained(“/”) :PathPart(‘catalog’) :CaptureArgs(1) {
      # Starts a chain when the URL begins with ‘catalog’
      # Next item is captured, probably an id
}

sub item :Chained(“base”) :PathPart(‘item’) :CaptureArgs(1) {
      # Called after ‘base’, when there’s an ‘item’ in the URL
}

sub order :Chained(“item”) :PathPart(‘order’) :Args(0) {
      # Called after item when there’s an ‘order’ in the URL
      # Args() instead of CaptureArgs ends chain
}

# URLs like /catalog/1234/item/5678/order will trigger:
# base(1234); item(5678); order();
```

Writing Actions

- Methods in a Controller object
- Controllers loaded as plugins
- Methods called with a Context parameter, and any parameters from the URL

The Context

- First parameter to an Action
- Written a $c or $ctx or $context
- Contains the important data from Catalyst

Stuff in the Context

- $c->req - the Request
- $c->rep - the Reply
- $c->stash - Values to be used by the view

Request

- All the stuff you need know about the HTTP request the user sent.
- Catalyst::Request object
- Contains:
  - URL
  - Arguments
  - Paramaters
  - Body
  - Cookies
  - Method (POST, GET, etc.)

Response

- Filled in to provide the reply to the user. A web page goes here!
- Catalyst::Response object
- Contains:
  - Headers
  - Body
  - Cookies

Writing an Action

```
sub login :Local :Args(0) {
    my ($self, $c) = @_;

    # Get the username and password from form
    my $username = $c->request->params->{username};
    my $password = $c->request->params->{password};

    # If the username and password values were found in form
    if ($username && $password) {
        # Attempt to log the user in
        if ($c->authenticate({ username => $username, password => $password  } )) {
        # If successful, then let them use the application
        $c->response->redirect($c->uri_for($c->controller('Records')->action_for('list')));
        return;
      } else {
        # Set an error message
        $c->stash(error_msg => "Bad username or password.");
      }
    } else {
      # Set an error message
      $c->stash(error_msg => "Empty username or password.")
        unless ($c->user_exists);
    }
}
```

Resources

- Catalyst::Manual
- http://www.catalystframework.org/
- IRC
- “The Definitive Guide to Catalyst”, Diment, Trout, et al. (Apress, 2009)

Q&A

- Questions?
- Maybe a look at a real Catalyst app.