Catalyst is an elegant web application framework written in Perl (see catalyst.perl.org and the Catalyst module on CPAN). The information in this refcard is abbreviated and simplified – refer to the documentation for full details.

Creating a Catalyst Application
Catalyst includes a helper script to bootstrap a new application:

```bash
$catalyst.pl My::App
```

This creates a application directory `My-App` with skeleton module, build, and test files and a number of helper scripts, including one to launch a test server:

```bash
$ cd My-App
$ script/my_app_server.pl
... You can connect to your server at http://host.domain:3000/
```

Application Structure
The generated application module is structured like this (with comments and documentation omitted for brevity):

```perl
package My::App;
use Catalyst qw/-Debug/;

My::App->config( name => 'My::App' );
My::App->setup;

sub default : Private {
    my ( $self, $c ) = @_;
    $c->res->output('Congratulations, ' .
        'My::App is on Catalyst!');
}
```

Catalyst loads any `component` modules it finds within the subdirectories: M (or Model), V (or View), or C (or Controller).

`catalyst.pl` creates another helper script for generating skeleton components modules:

```bash
$ script/my_app_create.pl {M|V|C} Class \[helper arg ...]
```

Actions
The behaviour of a Catalyst application is defined by its `actions`, subroutines marked with attributes defined in the application or component modules:

```perl
sub action1 : Global { ... }
    top-level action mapped to application base URL
sub action2 : Local { ... }
    'namespace-prefixed' – sub name prefixed by component URL
sub action2 : Private { ... }
    only accessible within application via forward method
```
sub action3 : Path('path') { ... }  
    literal action matches specified URL
sub action4 : Regex('regex') { ... }  
    globally matches URLs that match regex

There are four built-in private actions:
sub begin : Private { ... }  
    called at start of request before matching actions
sub auto : Private { ... }  
    special action run after begin in each class from app down
sub default : Private { ... }  
    called if no other action matches
sub end : Private { ... }  
    called at end of request after all matching actions

Context methods
Each action subroutine is invoked with two parameters – the component object reference or class name and a context object (conventionally designated $c).
$obj = $c->config
$obj = $c->forward($action [, @args])
$obj = $c->log($newobj)
$obj = $c->plugin($name, $class, @args)
$req = $c->request # or $c->req
$resp = $c->response # or $c->res
$href = $c->stash

Methods for plugin writers
$obj = $c->dispatcher
$obj = $c->engine
$aref = $c->get_action($name, $namespace, $inherit)

Request methods
$req = $c->request # or $c->req
$str = $req->action
$str = $req->address
$str = $req->arguments # or $req->args
$str = $req->base
$str = $req->body
$str = $req->content_encoding
$str = $req->content_length
$str = $req->content_type
$str = $req->cookie($name)
@ary = $req->cookies
$str = $req->header
$str = $req->headers
$str = $req->hostname
$str = $req->input
$str = $req->match
$str = $req->method
valu = $req->param($name)
$href = $req->parameters # or $req->params
$str = $req->path
$str = $req->protocol
$str = $req->referer
$str = $req->secure
$str = $req->snippets
$str = $req->upload
$href = $req->uploads
$str = $req->uri
$str = $req->user
$str = $req->user_agent

Response methods
$resp = $c->response
$str = $resp->body($str)
$str = $resp->content_encoding($str)
$str = $resp->content_length($str)
$str = $resp->content_type($str)
$href = $resp->cookies
$str = $resp->header($name, $value)
$href = $resp->headers
$str = $resp->output($str) # alias for $resp->body
$str = $resp->redirect($location, $status)
$val = $resp->status($status)

Log methods
$log = $c->log
$log->levels(@levels)
$log->enable(@levels)
logfile->disable(@levels)
$bool = $log->is_debug
logfile->debug($message)
$bool = $log->is_info
logfile->info($message)
$bool = $log->is_warn
logfile->warn($message)
$bool = $c->is_error
logfile->error($message)
$bool = $log->is_fatal
logfile->fatal($message)

Utility Functions (Catalyst::Utils)
$aref = attrs($coderef)
$str = class2appclass($class)
$str = class2classprefix($class)
$str = class2classsuffix($class)
$str = class2env($class)
$str = class2prefix($class)
$str = class2tempdir($class, $create)
$str = home($class)
$str = prefix($class, $action)
$str = reflect_actions($class)
$str = request($str)

Plugins