Test::WWW::Mechanize::Catalyst::WithContext

test your Catalyst apps in context

Julien Fiegehenn (simbabque)

Catalyst

Catalyst::Test

```
use Catalyst::Test 'TestApp';
my $content  = get('index.html');           # Content as string
my $response = request('index.html');       # HTTP::Response object
my($res, $c) = ctx_request('index.html'); # HTTP::Response & context obj

use HTTP::Request::Common;
my $response = request POST '/foo', [
    bar => 'baz',
    something => 'else'
];
```

Test::WWW::Mechanize::Catalyst

```
use Test::WWW::Mechanize::Catalyst;

# To test a Catalyst application named 'Catty':
my $mech = Test::WWW::Mechanize::Catalyst->new(catalyst_app => 'Catty');

$mech->get_ok("/"); # no hostname needed
```

Lots of possibilities for unit and integration tests

Context represents the current request

- `$c` or `$ctx`
- the `Catalyst` object
- reference
- new on every request

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

    $c->session;
    $c->stash;
    $c->req;

}
```

```perl
use Test::WWW::Mechanize::Catalyst::WithContext;

my $mech = Test::WWW::Mechanize::Catalyst::WithContext->new(
    catalyst_app => 'Catty'
);

my ($res, $c) = $mech->get_context("/"); # $c is a Catalyst context
is $c->stash->{foo}, "bar", "foo got set to bar";
```

Why?!

`$c->stash`

- did we load the right template?
- initialization JSON for frontend app
- internal status that's hard to reach otherwise

```
my ($res, $c) = $mech->get_context("/internal/stuff");

is $res->code, 2000, "stuff loaded ok";

is $c->stash->{template}, "special_stuff.tt", "... with the special temp

ok $c->stash->{use_special_stuff}, "... and special flag got set";
```

`$c->session`

get to the session object without the persistence layer

- did we store the right values?
- manipulate the session
- possibly even load session fixtures

```
 (undef, my $c_before) = $mech->get_context('/');

my ( $res, $c_after ) = $mech->get_context('/change/session');

isnt $c_before->session->{foo}, $c_after->session->{foo}, 'foo got chang
```

```
(undef, my $c) = $mech->get_context('/');

$c->session->{cart}->add Article::new( sku => 1337 );

$mech->post_ok( '/checkout', $form );
```

`$c->model`, `$c->view`, `$c->controller`

- easy way to get an instance from the app with `ACCEPT_CONTEXT` and config
- do unit tests on them
- manipulate configuration without Sub::Override

`$c->custom_stuff`

- call methods and accessors from `MyApp::Catalyst`

All of these honor your session cookie

Example app in the t/ directory

Everything else that Test::WWW::Mechanize::Catalyst does

- LWP::UserAgent
- WWW::Mechanize
- Test::WWW::Mechanize
- Test::WWW::Mechanize::Catalyst
- Test::WWW::Mechanize::Catalyst::WithContext

Questions?

https://metacpan.org/pod/Test::WWW::Mechanize::Catalyst::Wi