Integrating with Facebook using Facebook::Graph

Ticketmaster – Michael Olson

August 31, 2011

Facebook::Graph

- Introduction
- How to extend Facebook::Graph

Ticketmaster backend

- Facebook requests
- Example: Verifying a Facebook session
- Challenges with FB integration
- Getting data for many FB users

Introduction to Facebook::Graph

Facebook::Graph - A fast and easy way to integrate your apps with Facebook.

- `http://search.cpan.org/~rizen/Facebook-Graph/`
- `http://github.com/rizen/Facebook-Graph`

Introduction to Facebook::Graph

Strengths:

- Supports most Facebook Graph API featuers
- Uses Moose/Mouse which makes it relatively easy to extend

Weaknesses:

- No support for FQL (though largely unnecessary because Graph API covers almost everything you would need FQL for)

How to extend Facebook::Graph

```
package My::Facebook::Graph::AppAccessToken;

use Any::Moose;
use Facebook::Graph;
with 'Facebook::Graph::Role::Uri';

has new_field => (is => 'ro', required => 1);

sub uri { ... }
sub uri_as_string { ... }

no Any::Moose;
__PACKAGE__->meta->make_immutable;
1;
```

Facebook requests

Front-end will request:

- Single user
- Info about all people tagged on an event
- Friends of user (requires Facebook session to be passed)

Verifying a Facebook session

```
use Digest::MD5 qw(md5_hex);

my ( $self, $session ) = @_;
return 0 if ref $session ne 'HASH' ||
            !defined $session->{sig} ||
            !defined $APP_SECRET ||
            (defined $session->{expires} &&
             $session->{expires} + 600 < time);
my $sig = join('', map { "$_=" . $session->{$_} }
       grep { $_ ne 'sig' } sort keys %$session);
$sig .= $APP_SECRET;
return (md5_hex($sig) eq $session->{sig});
```

Challenges: Cache

- Problem: Facebook doesn't like it if you perform too many requests in a certain timeframe; especially same user.
- So we need to cache the data in some way.
- Option 1: Put per-user results in a database (or noSQL solution)
  - Able to fully minimize FB calls
  - Works well with FB Realtime API
- Option 2: Put per-request results in cache
  - Quicker to implement
  - No additional DB load

Challenges: Queue

- Problem: Need to be careful not to hold too many open connections on application box.
- Solution: Use intermediary queue server to hold the connection open, and periodically do front-end initiated poll.
- We set aside a field in the backend response called 'retry'; this field is used to store encrypted continuation data (token) to perform a poll on queue.
- Since there can be multiple FB requests in a single backend request, we use a hash to indicate the kind of request and its token.

Requesting multiple facebook users

- Problem: We need to have first name, last name, picture of all Facebook users that are tagged to a certain event.
- If we were storing FB data in a DB, this could be a simple join.
- As it is, we need to request multiple FB users at once.
- We cache all of these users according to the event ID, and request any that we may not have info about yet.

Facebook call for multi-user

Example Facebook::Graph request for this info

```
$fb->query()
   ->select_fields(qw( id name picture ))
   ->where_ids(@missing)->request;
```

- Makes a call to '/', setting `ids` query arg to long list of IDs.
- List is so long that we need to use a POST.

POST request

```
$response = post(
    body => 'method=get&' . $query,
    content_type =>
      'application/x-www-form-urlencoded');
```

- Include 'method=get' in query params so that Facebook API can treat it like a GET request.
- FB API
  - GET: Request info
  - POST: Write info

Facebook on ISM

Example of Facebook integration on Ticketmaster.com:

- ticketmaster.com/event/0B0046C1D30F4734
- (Distant Worlds: Music from Final Fantasy)