HTTP::Async

Colin Bradford

The problem

- Company manages multiple platforms for clients, each independent
- Platforms expose web service for status
- Overview tools need data from all platforms
  - Status pages are relatively heavy on the database
  - Response time needs to be quick

The solution

- Fetch status data in parallel
- Combine incoming data into a single structure
- Display to user

Advantages of parallel fetch

- Queries run in parallel
- Timeout is limited to the maximum service timeout, not the sum of the service timeouts

Disadvantages of parallel fetch

- Heavier load on shared infrastructure
- Error handling is more complex

How to fetch in parallel?

- HTTP::Async
- Same request/response objects from LWP
- Usage:
  - Register a number of requests
  - Then either
    - Loop and process responses as they come back
    - Or poll for responses to arrive
    - Or integrate into an event loop

Useful features

- Limit on number of simultaneous requests
- Timeout on individual requests

Code: the multiplexor

```
sub getResponseFromAllEndpoints {
    my ($self, @extraArgs) = @_;
    my %responseData;    my %requestMap;
    my $fetcher = HTTP::Async->new();

    foreach my $systemId (@ { $self->getListOfSystemIds }) {
        my $requestId = $fetcher->add($self->createRequestObject($systemId, @extraArgs));
        $requestMap{$requestId} = $systemId;
    }
    while (my ($responseObject, $requestId) = $fetcher->wait_for_next_response) {
        my $response = $self->processResponseFromEndpoint($responseObject);
        if ($response->{status} eq 'OK') {
            $self->mergeResponseData(\%responseData, $response->{data});
        }
    }
    return \%responseData;
}
```

Code: create the request

```
sub createRequestObject {
    my ($self, $systemId, @extraargs) = @_;

    my %args = ( method  => 'GET', @extraargs );
    my $baseURL = $self->getURLForService($systemId);

    my $url = join('/', $baseURL, $args{service}, $systemId);

    my $request = HTTP::Request->new($args{method} => $url);

    $request->content_type('application/json');
    if (defined $args{data}) {
        $request->content(JSON::XS::encode_json($args{data}));
    }
    return $request;
}
```

Code: handle each response

```
sub processResponseFromEndpoint {
    my ($self, $response) = @_;


    if ($response->is_success() ) {
        my $responseData = JSON::XS::decode_json($response->content);
        return { data => $responseData, status => 'OK', };
    }
    $log->debug("failed request: ".$response->code);
    return { status  => 'FAIL', code => $response->code, };
}
```

Code: merge data from responses

```
sub mergeResponseData {
    my ($self, $currentData, $newData) = @_;

    foreach my $key (keys %$newData) {
        # Magic key - if it's the {data} key, and it's an array, then push the data. Otherwise, fall through
        if ( ( ($key eq 'data') || ($key eq 'filter') )
             && (ref($newData->{data}) eq 'ARRAY')) {
            push (@{$currentData->{$key}}, @{$newData->{$key}});
            next;
        }
        if (!exists $currentData->{$key}) {
            $currentData->{$key} = $newData->{$key};
        }
    }
}
```

Questions