Templates

(as a service)

Who Am I?

Logan Bell
twitter: @epochbell
IRC: logie
https://metacpan.org/author/LOGIE
Work for Shutterstock

Why was the JavaScript developer sad?

Because he didn't Node how to Express himself

Things to cover

- What is Swig.js?
- Swig in Perl
- Swig as a service
- Why?
- How to?
- Questions

What is Swig

- Inspired by Django/Twig style templates
- Written by Paul Armstrong
- Written in JavaScript
- It’s hip with the kids

Example

```
{% for item in items %}

  <div class="item">{{ item }}</div>

{% endfor %}


<div class="price">{{ price }}</div>
```

What is Swig

- Supports tags
- Object Oriented template inheritance
- express.js compatible
- Also works with node.js

Webstack

- Platform team
- Need for unified templates across multiple languages
- Dancer and Modern Perl
- Moving away from Mason

Perl and JavaScript

- Javascript templates and Perl backend?
- But we want to use Dancer
- But we want to share Templates
- We also want to use Swig

Template::Swig

- Perl wrapper that compiled JavaScript
- JavaScript::V8
- Upsides:
  - Fast
- Downsides:
  - Difficult to upgrade
  - Difficult to keep custom tags in-line

JavaScript::V8

```
use JavaScript::V8;

my $context = JavaScript::V8::Context->new;

$context->bind_function(yo => sub { print @_ });

$context->eval(q{

   // This is javascript
   yo(“word up");

});
```

JavaScript::V8

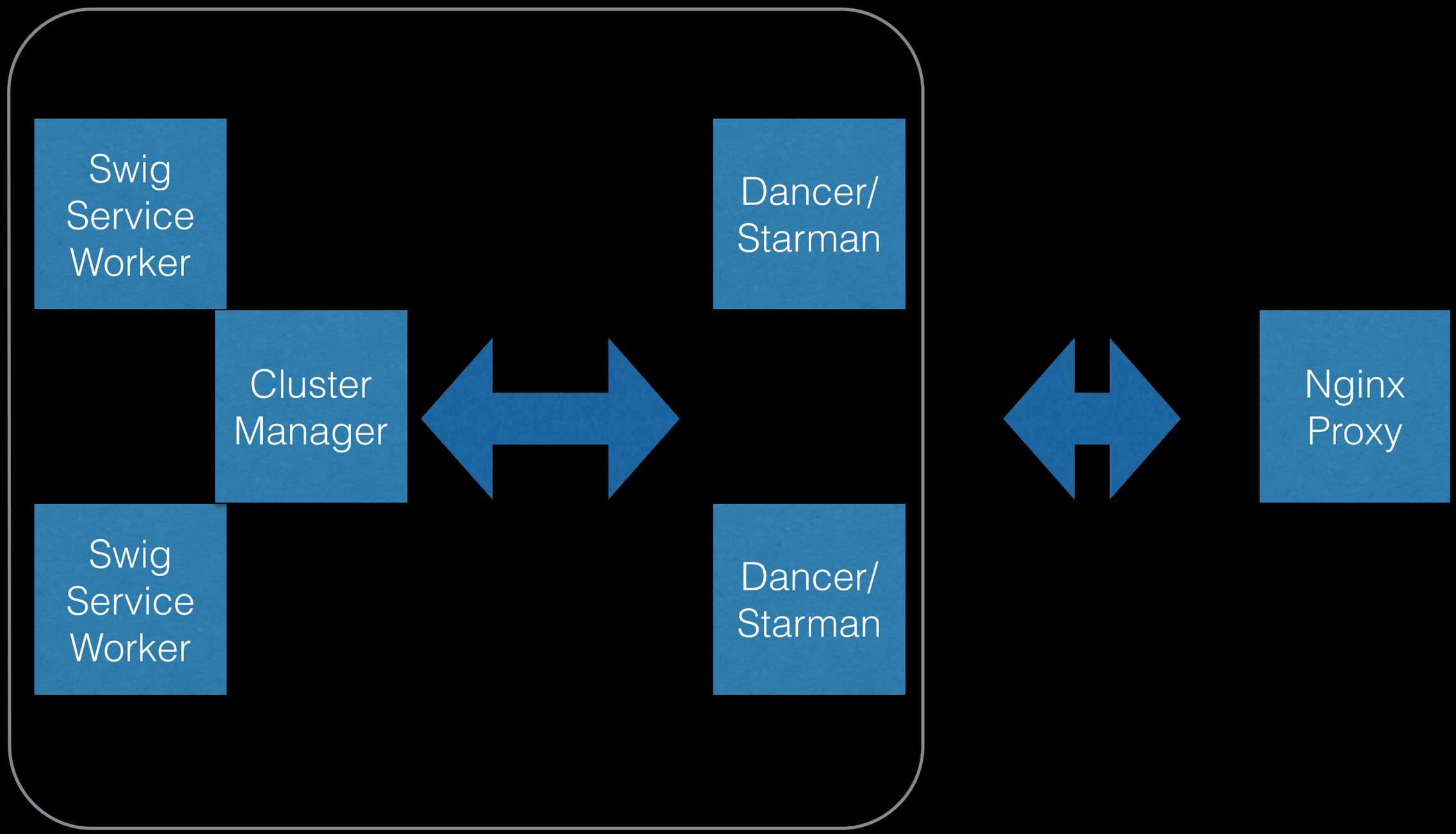
```
use JavaScript::V8;

my $context = JavaScript::V8::Context->new;

$context->bind_function(yo => sub { print @_ });

$context->eval(q{

   // This is javascript
   yo(“word up");

});
```

Rage

Let’s make it a service!

- We’ll use node.js
- Build it with express.js
- We can use i18n
- Perl/Ruby/Php can share templates!

This is what it looks like



WebService::SwigClient

- Uses WWW::Curl under the hood
- Meant to be fast, caches connection

WebService::SwigClient

```
use WebService::SwigClient;

my $client = WebService::SwigClient->new(
   service_url   => http://localhost:8999,
   error_handler => sub {
      my ($error, $curl_object) = @_;
      warn $error;
    },
);

$client->render('foo.html', { param => 1, param => 2});
```

We also need the service

git clone git@github.com:shutterstock/a-swig-service.git

npm install

bin/run-swig —port=8999 --templates=./templates

Let’s learn to dance

Dancer::Plugin::Swig

- Wrapper around WebService::SwigClient
- Easy to setup and get started with

How to use it

- Setup a dancer application:
  perl -a Dancer MySwig::App

- Install the swig module:
  cpanm Dancer::Plugin::Swig

The Config

```
plugins:
    Swig:
      service_url: "http://localhost:8999"
```

Route File

```
package NewApp;
use Dancer ':syntax';
use Dancer::Plugin::Swig;

our $VERSION = '0.1';

get '/' => sub {
    render 'index.html', { hello_world => 'howdy' };
};

true;
```

Route File

```
package NewApp;
use Dancer ':syntax';
use Dancer::Plugin::Swig;

our $VERSION = '0.1';

get '/' => sub {
    render 'index.html', { hello_world => 'howdy' };
};

true;
```

The Template File

```
<html>
  <head>
    <title>Hi</title>
  </head>
  <body>
    <span>{{ hello_world }}</span>
  </body>
</html>
```

Run the service

bin/run-swig —port=8999 --templates=./templates

Try it

open http://my.dancer.application.com/

```
<html>
  <head>
    <title>Hi</title>
  </head>
  <body>
    <span>howdy</span>
  </body>
</html>
```

Basic i18n support

- Built in swig tag {% i18n %}
- Can watch for a translation file and load it appropriately
- Uses a npm module yacm (yet another cluster manger)

i18n tag

{% i18n TAG_NAME %} Default {% eni18n %}

{% i18n TAG_NAME key:value %}
Default key
{% end18n %}

The Template File

```
<html>
  <head>
    <title>Hi</title>
  </head>
  <body>
     <span>
         {% i18n hello_world %}Not found{% endi18n %}
     </span>
  </body>
</html>
```

Route File

```
package NewApp;
use Dancer ':syntax';
use Dancer::Plugin::Swig;

our $VERSION = '0.1';

get '/' => sub {
        render 'index.html', {
            i18n          => { language => ‘es’ },
            hello_world => ‘HELLO_WORLD'
         };

true;
```

Translation File

```
{
  "HELLO_WORLD": {
    "en": "howdy",
    "es": "hola mundo",
    "fr": "bonjour tout le monde"
  }
}
```

Run the service

```
bin/run-swig —port=[port] \
    —templates=./templates \
    —translations =./translations.json
```

Try it

open http://my.dancer.application.com/

```
<html>
  <head>
    <title>Hi</title>
  </head>
  <body>
    <span>hola mundo</span>
  </body>
</html>
```

Heavy Load, No Problem

```
bin/run-swig —port=[port] \
        —templates=./templates \
        —translations =./translations.json \
        —workers = 2
```

Workers

- Using node clusters, which create network processes that share the same port.
- We use yacm to manage our workers.
- Watches for changes in files and reloads into memory

Other Interesting Features

- ruby/sinatra client
- it's faster than Mason 1

Where to learn more

- https://github.com/shutterstock/a-swig-service
- https://metacpan.org/pod/WebService::SwigClient
- https://metacpan.org/pod/release/LOGIE/Dancer-Plugin-Swig-0.02/README.pod

Thank You!

- Shutterstock: For both employment and the images for this talk
- Webstack Team: Nikolay, Vishal, Adam, Kevin, and Belden
- My wife and two girls: Amy, Chloe and Kassidy! Love all of ya!
- The site where I stole my joke: http://www.elijahmanor.com/front-end-web-dev-jokes/

Questions