Building APIs with Silex

Cal Evans
cal@calevans.com

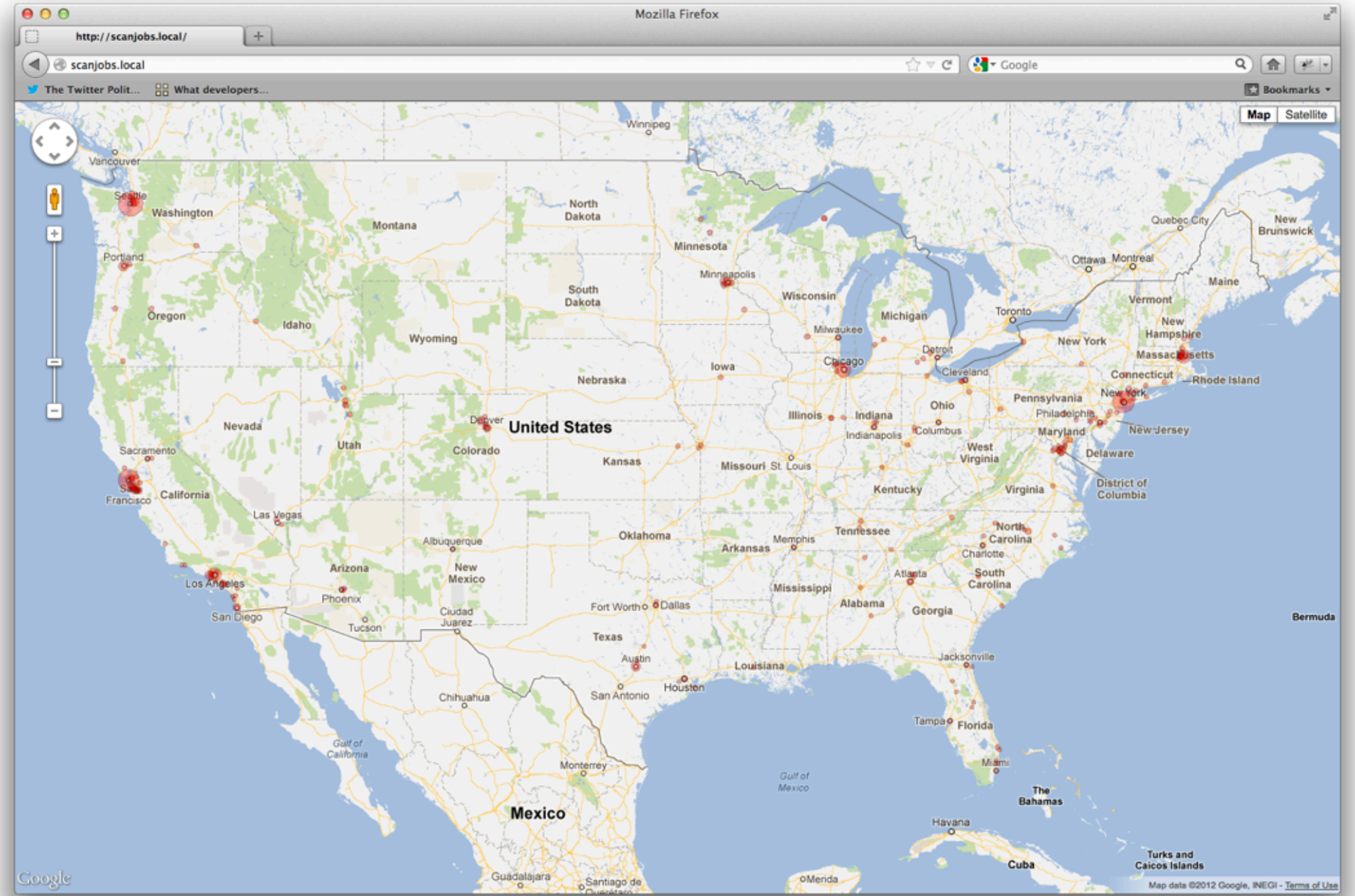
@calevans

The Sample Application

https://github.com/calevans/scanjobs

https://joind.in/7094

The Sample Application



Introduction to Silex

- Not a micro-framework

The MicroPHP Manifesto

http://funkatron.com/posts/the-microphp-manifesto.html

Introduction to Silex

```
$ cloc vendor
    1073 text files.
    1063 unique files.
     667 files ignored.

http://cloc.sourceforge.net v 1.56  T=6.0 s (157.5 files/s, 20622.2 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
PHP                            919          17360          37173          68674
XML                             12             32             28            257
Bourne Shell                     2             18              3             66
YAML                            10              9              0             52
XSD                              1              8              0             36
Python                           1              5              0             12
-------------------------------------------------------------------------------
SUM:                           945          17432          37204          69097
-------------------------------------------------------------------------------
```

Introduction to Silex

- Not a micro-framework
- What it is good for

Introduction to Silex

- Not a micro-framework
- What it is good for
- What is isn't good for

Introduction to Composer

- Composer is not PEAR
- Composer is useful
- Creating your composer.json
  - Require
  - autoloaders

composer.json

```
{
    "name": "scanJobs",
    "minimum-stability": "dev",
    "require": {
      "silex/silex": "1.0.*",
      "doctrine/dbal": ">=2.2",
      "knplabs/console-service-provider": ">=1.0.0",
      "igorw/config-service-provider": "*.*.*"
      },
    "autoload": {
    "psr-0": {
      "CalEvans": "src/",
      "ScanJobs": "app/"}
     }
}
```

Introduction to Composer

- Composer is not PEAR
- Composer is useful
- Creating your composer.json
  - Require
  - autoloaders
- Composer & git

Basic Project Structure

- No required structure for Silex
- Composer has required structure

Sample Application Structure

```
scanJobs
    |-app                    |-app
    |-config                 |---ScanJobs
    |-data                   |-----Command
    |-logs                   |-----Controller
    |-public                 |-----Model
    |-scripts
    |-src
    |-tmp
    |-vendor
```

Sample Application Structure

```
scanJobs
    |-app
    |-config
    |-data
    |-logs
    |-public
    |-scripts
    |-src
    |-tmp
    |-vendor
```

```
|-app
|---ScanJobs
|-----Command
|-----Controller
|-----Model
```

Sample Application Structure

```
scanJobs
    |-app                    |-app
    |-config                 |---ScanJobs
    |-data                   |-----Command
    |-logs                   |-----Controller
    |-public                 |-----Model
    |-scripts
    |-src
    |-tmp
    |-vendor
```

Sample Application Structure

```
scanJobs
    |-app                    |-app
    |-config                 |---ScanJobs
    |-data                   |-----Command
    |-logs                   |-----Controller
    |-public                 |-----Model
    |-scripts
    |-src
    |-tmp
    |-vendor
```

Sample Application Structure

```
scanJobs
    |-app
    |-config
    |-data
    |-logs
    |-public
    |-scripts
    |-src
    |-tmp
    |-vendor
```

```
|-app
|---ScanJobs
|-----Command
|-----Controller
|-----Model
```

Sample Application Structure

```
scanJobs
    |-app
    |-config
    |-data
    |-logs
    |-public
    |-scripts
    |-src
    |-tmp
    |-vendor
```

```
|-app
|---ScanJobs
|-----Command
|-----Controller
|-----Model
```

Sample Application Structure

```
scanJobs
    |-app
    |-config
    |-data
    |-logs
    |-public
    |-scripts
    |-src
    |-tmp
    |-vendor
```

```
|-app
|---ScanJobs
|-----Command
|-----Controller
|-----Model
```

Basic Project Structure

- No required structure for Silex
- Composer has required structure
- Application root vs. Document Root

Configuration

- What I did not choose
- What I did choose

Configuration

- What I did not choose
- What I did choose
- Editing JSON by hand is stupid

The Bootstrap

- Why a Bootstrap?
- DRY
- Security

Bootstrap.php

app/Bootstrap.php

```
<?PHP
$env = getenv('APP_ENV') ?: 'prod';
$app = new Silex\Application();
$app->register(new Igorw\Silex
\ConfigServiceProvider(__DIR__."/../config/{$env}.json"));
$app->register(new Silex\Provider
\DoctrineServiceProvider(),
                array('db.options' => $app['database']));
return $app;
```

Bootstrap.php

app/Bootstrap.php

```
<?PHP
$env = getenv('APP_ENV') ?: 'prod';
$app = new Silex\Application();
$app->register(new Igorw\Silex
\ConfigServiceProvider(__DIR__."/../config/{$env}.json"));
$app->register(new Silex\Provider
\DoctrineServiceProvider(),
                array('db.options' => $app['database']));
return $app;
```

Bootstrap.php

app/Bootstrap.php

```
<?PHP
$env = getenv('APP_ENV') ?: 'prod';
$app = new Silex\Application();
$app->register(new Igorw\Silex
\ConfigServiceProvider(__DIR__."/../config/{$env}.json"));
$app->register(new Silex\Provider
\DoctrineServiceProvider(),
                array('db.options' => $app['database']));
return $app;
```

Bootstrap.php

app/Bootstrap.php

```
<?PHP
$env = getenv('APP_ENV') ?: 'prod';
$app = new Silex\Application();
$app->register(new Igorw\Silex
\ConfigServiceProvider(__DIR__."/../config/{$env}.json"));
$app->register(new Silex\Provider
\DoctrineServiceProvider(),
                array('db.options' => $app['database']));
return $app;
```

Bootstrap.php

app/Bootstrap.php

```
<?PHP
$env = getenv('APP_ENV') ?: 'prod';
$app = new Silex\Application();
$app->register(new Igorw\Silex
\ConfigServiceProvider(__DIR__."/../config/{$env}.json"));
$app->register(new Silex\Provider
\DoctrineServiceProvider(),
                array('db.options' => $app['database']));
return $app;
```

The Bootstrap

- Why a Bootstrap?
  - DRY
  - Security
- Uses APP_ENV
- Makes 1 assumption
- Creates our database connection

The Web Dispatch Script

public/index.php

```
<?PHP
require_once __DIR__.'/../vendor/autoload.php';

use ScanJobs\Controllers;

$app = require '../app/Bootstrap.php';

$app->mount('/jobs', new Controllers\JobsController());
$app->mount('/cities', new Controllers\CityController());
$app->mount('/companies', new Controllers
\CompanyController());
$app->mount('/', new Controllers\IndexController());

$app->run();
```

The Web Dispatch Script

public/index.php

```
<?PHP
require_once __DIR__.'/../vendor/autoload.php';

use ScanJobs\Controllers;

$app = require '../app/Bootstrap.php';

$app->mount('/jobs', new Controllers\JobsController());
$app->mount('/cities', new Controllers\CityController());
$app->mount('/companies', new Controllers
\CompanyController());
$app->mount('/', new Controllers\IndexController());

$app->run();
```

The Web Dispatch Script

public/index.php

```
<?PHP
require_once __DIR__.'/../vendor/autoload.php';

use ScanJobs\Controllers;

$app = require '../app/Bootstrap.php';

$app->mount('/jobs', new Controllers\JobsController());
$app->mount('/cities', new Controllers\CityController());
$app->mount('/companies', new Controllers
\CompanyController());
$app->mount('/', new Controllers\IndexController());

$app->run();
```

The Web Dispatch Script

public/index.php

```
<?PHP
require_once __DIR__.'/../vendor/autoload.php';

use ScanJobs\Controllers;

$app = require '../app/Bootstrap.php';

$app->mount('/jobs', new Controllers\JobsController());
$app->mount('/cities', new Controllers\CityController());
$app->mount('/companies', new Controllers
\CompanyController());
$app->mount('/', new Controllers\IndexController());

$app->run();
```

The Web Dispatch Script

public/index.php

```
<?PHP
require_once __DIR__.'/../vendor/autoload.php';

use ScanJobs\Controllers;

$app = require '../app/Bootstrap.php';

$app->mount('/jobs', new Controllers\JobsController());
$app->mount('/cities', new Controllers\CityController());
$app->mount('/companies', new Controllers
\CompanyController());
$app->mount('/', new Controllers\IndexController());

$app->run();
```

The CLI Dispatch Script

app/console.sh

```
#!/usr/bin/env php
<?php
require_once __DIR__.'/../vendor/autoload.php';

use ScanJobs\Command;
use CalEvans\Google\Geocode as Geocode;
use Knp\Provider\ConsoleServiceProvider;

$app = require 'Bootstrap.php';

$app->register(new ConsoleServiceProvider(),
                array('console.name'              => 'Console',
                      'console.version'           => '1.0.0',
                      'console.project_directory' => __DIR__.'/..'));
$application = $app['console'];
$scan = new Command\ScanJobsCommand();
$scan->addGeocodeer(new Geocode());
$application->add($scan);
$application->add(new Command\NewDatabaseCommand());
$application->add(new Command\WorkCommand());
$application->run();
```

The CLI Dispatch Script

app/console.sh

```
#!/usr/bin/env php
<?php
require_once __DIR__.'/../vendor/autoload.php';

use ScanJobs\Command;
use CalEvans\Google\Geocode as Geocode;
use Knp\Provider\ConsoleServiceProvider;

$app = require 'Bootstrap.php';

$app->register(new ConsoleServiceProvider(),
                array('console.name'              => 'Console',
                      'console.version'           => '1.0.0',
                      'console.project_directory' => __DIR__.'/..'));
$application = $app['console'];
$scan = new Command\ScanJobsCommand();
$scan->addGeocodeer(new Geocode());
$application->add($scan);
$application->add(new Command\NewDatabaseCommand());
$application->add(new Command\WorkCommand());
$application->run();
```

The CLI Dispatch Script

app/console.sh

```
#!/usr/bin/env php
<?php
require_once __DIR__.'/../vendor/autoload.php';

use ScanJobs\Command;
use CalEvans\Google\Geocode as Geocode;
use Knp\Provider\ConsoleServiceProvider;

$app = require 'Bootstrap.php';

$app->register(new ConsoleServiceProvider(),
                array('console.name'              => 'Console',
                      'console.version'           => '1.0.0',
                      'console.project_directory' => __DIR__.'/..'));
$application = $app['console'];
$scan = new Command\ScanJobsCommand();
$scan->addGeocodeer(new Geocode());
$application->add($scan);
$application->add(new Command\NewDatabaseCommand());
$application->add(new Command\WorkCommand());
$application->run();
```

The CLI Dispatch Script

app/console.sh

```
#!/usr/bin/env php
<?php
require_once __DIR__.'/../vendor/autoload.php';

use ScanJobs\Command;
use CalEvans\Google\Geocode as Geocode;
use Knp\Provider\ConsoleServiceProvider;

$app = require 'Bootstrap.php';

$app->register(new ConsoleServiceProvider(),
                array('console.name'              => 'Console',
                      'console.version'           => '1.0.0',
                      'console.project_directory' => __DIR__.'/..'));
$application = $app['console'];
$scan = new Command\ScanJobsCommand();
$scan->addGeocodeer(new Geocode());
$application->add($scan);
$application->add(new Command\NewDatabaseCommand());
$application->add(new Command\WorkCommand());
$application->run();
```

The CLI Dispatch Script

app/console.sh

```
#!/usr/bin/env php
<?php
require_once __DIR__.'/../vendor/autoload.php';

use ScanJobs\Command;
use CalEvans\Google\Geocode as Geocode;
use Knp\Provider\ConsoleServiceProvider;

$app = require 'Bootstrap.php';

$app->register(new ConsoleServiceProvider(),
                array('console.name'              => 'Console',
                      'console.version'           => '1.0.0',
                      'console.project_directory' => __DIR__.'/..'));
$application = $app['console'];
$scan = new Command\ScanJobsCommand();
$scan->addGeocodeer(new Geocode());
$application->add($scan);
$application->add(new Command\NewDatabaseCommand());
$application->add(new Command\WorkCommand());
$application->run();
```

The CLI Dispatch Script

app/console.sh

```
#!/usr/bin/env php
<?php
require_once __DIR__.'/../vendor/autoload.php';

use ScanJobs\Command;
use CalEvans\Google\Geocode as Geocode;
use Knp\Provider\ConsoleServiceProvider;

$app = require 'Bootstrap.php';

$app->register(new ConsoleServiceProvider(),
                array('console.name'              => 'Console',
                      'console.version'           => '1.0.0',
                      'console.project_directory' => __DIR__.'/..'));
$application = $app['console'];
$scan = new Command\ScanJobsCommand();
$scan->addGeocodeer(new Geocode());
$application->add($scan);
$application->add(new Command\NewDatabaseCommand());
$application->add(new Command\WorkCommand());
$application->run();
```

The CLI Dispatch Script

app/console.sh

```
#!/usr/bin/env php
<?php
require_once __DIR__.'/../vendor/autoload.php';

use ScanJobs\Command;
use CalEvans\Google\Geocode as Geocode;
use Knp\Provider\ConsoleServiceProvider;

$app = require 'Bootstrap.php';

$app->register(new ConsoleServiceProvider(),
                array('console.name'              => 'Console',
                      'console.version'           => '1.0.0',
                      'console.project_directory' => __DIR__.'/..'));
$application = $app['console'];
$scan = new Command\ScanJobsCommand();
$scan->addGeocodeer(new Geocode());
$application->add($scan);
$application->add(new Command\NewDatabaseCommand());
$application->add(new Command\WorkCommand());
$application->run();
```

The CLI Dispatch Script

app/console.sh

```
#!/usr/bin/env php
<?php
require_once __DIR__.'/../vendor/autoload.php';

use ScanJobs\Command;
use CalEvans\Google\Geocode as Geocode;
use Knp\Provider\ConsoleServiceProvider;

$app = require 'Bootstrap.php';

$app->register(new ConsoleServiceProvider(),
                array('console.name'              => 'Console',
                      'console.version'           => '1.0.0',
                      'console.project_directory' => __DIR__.'/..'));
$application = $app['console'];
$scan = new Command\ScanJobsCommand();
$scan->addGeocodeer(new Geocode());
$application->add($scan);
$application->add(new Command\NewDatabaseCommand());
$application->add(new Command\WorkCommand());
$application->run();
```

You and your DB

- Connection is created in the Bootstrap
- Finding your connection
- Create the database

Creating the DB Connection

app/Bootstrap.php

```
$app->register(new Silex\Provider\DoctrineServiceProvider(),
               array('db.options' => $app['database']));
```

The CLI

- Why CLI?
  - Code reuse
  - Utility scripts
- The CLI Provider
  - Creating Commands

WorkCommand

app/ScanJobs/Command/WorkCommand.php

```
class WorkCommand extends Command
{
    protected function configure()
    {
        $this->setName('work')
        ->setDescription('Generic all-purpose work script');
    }

    protected function execute(\Symfony\Component\Console\Input\InputInterface $input,
                               \Symfony\Component\Console\Output\OutputInterface $output)
    {
        // Do something clever here
        $output->writeln('Done');
        return;
    }
}
```

WorkCommand

app/ScanJobs/Command/WorkCommand.php

```
class WorkCommand extends Command
{
    protected function configure()
    {
        $this->setName('work')
        ->setDescription('Generic all-purpose work script');
    }

    protected function execute(\Symfony\Component\Console\Input\InputInterface $input,
                               \Symfony\Component\Console\Output\OutputInterface $output)
    {
        // Do something clever here
        $output->writeln('Done');
        return;
    }
}
```

The CLI

- Why CLI?
  - Code reuse
  - Utility scripts
- The CLI Provider
  - Creating Commands
  - Registering Commands

Registering a command

```
$application->add(new Command\WorkCommand());
```

The API

- The heart of what we are doing
- This application has very little in the way of front-end. it is mostly backend.

The API

- The heart of what we are doing
- This application has very little in the way of front-end
- Several ways to create an API in Silex
  - ->get(), ->post(), ->delete(), ->update()

The API

```
$app->get('/info',function () {phpinfo();});
$app->put('/info',function () {phpinfo();});
$app->post('/info',function () {phpinfo();});
$app->delete('/info',function () {phpinfo();});
```

The API

- The heart of what we are doing
- This application has very little in the way of Frontend
- Several ways to create an API in Silex
  - ->get(), ->post(), ->delete(), ->update()
  - ->match()->method()

The API

```
$app->match('/info',function() {phpinfo();})
    ->method('GET|PUT|POST|DELETE|PATCH|HEAD|OPTIONS');

$app->match('/info',function() {phpinfo();})
    ->method('GET');
$app->match('/info',function() {phpinfo();})
    ->method('POST');
$app->match('/info',function() {phpinfo();})
    ->method('DELETE|PUT');
$app->match('/info',function() {phpinfo();})
    ->method('PATCH|HEAD|OPTIONS');
```

The API

- The heart of what we are doing
- This application has very little in the way of front-end
- Several ways to create an API in Silex
  - ->get(), ->post(), ->delete(), ->update()
  - ->match()->method()
- Route Variables

The API

```
$app->match('/helloworld/{firstName}',
   function($firstName)
   {
      echo "<h2>Hello {$firstName}</h2>\n";
})
->method('GET');
```

The API

- The heart of what we are doing
- This application has very little in the way of front-end
- Several ways to create an API in Silex
  - ->get(), ->post(), ->delete(), ->update()
  - ->match()->method()
- Route Variables
- Mounts vs. Routes

The API

```
$app->mount('/jobs',
            new Controller\JobsController());
```

The API

app/ScanJobs/Controller/JobsController.php

```
<?PHP
namespace ScanJobs\Controller;

use Silex\Application;
use Silex\ControllerProviderInterface;
use Silex\ControllerCollection;
use ScanJobs\Model\Job;

class JobsController implements ControllerProviderInterface
{
    protected $app;

    public function connect(Application $app)
    {
        $this->app = $app;

        $getJobsList = function()
        {
            return $this->getJobsList();
        };

        $controller = $app['controllers_factory'];
        $controller->get('/',$getJobsList);

        return $controller;
    }
```

The API

app/ScanJobs/Controller/JobsController.php

```
<?PHP
namespace ScanJobs\Controller;

use Silex\Application;
use Silex\ControllerProviderInterface;
use Silex\ControllerCollection;
use ScanJobs\Model\Job;

class JobsController implements ControllerProviderInterface
{
    protected $app;

    public function connect(Application $app)
    {
        $this->app = $app;

        $getJobsList = function()
        {
            return $this->getJobsList();
        };

        $controller = $app['controllers_factory'];
        $controller->get('/',$getJobsList);

        return $controller;
    }
```

The API

app/ScanJobs/Controller/JobsController.php

```php
<?PHP
namespace ScanJobs\Controller;

use Silex\Application;
use Silex\ControllerProviderInterface;
use Silex\ControllerCollection;
use ScanJobs\Model\Job;

class JobsController implements ControllerProviderInterface
{
    protected $app;

    public function connect(Application $app)
    {
        $this->app = $app;

        $getJobsList = function()
        {
            return $this->getJobsList();
        };

        $controller = $app['controllers_factory'];
        $controller->get('/',$getJobsList);

        return $controller;
    }
```

The API

app/ScanJobs/Controller/JobsController.php

```
<?PHP
namespace ScanJobs\Controller;

use Silex\Application;
use Silex\ControllerProviderInterface;
use Silex\ControllerCollection;
use ScanJobs\Model\Job;

class JobsController implements ControllerProviderInterface
{
    protected $app;

    public function connect(Application $app)
    {
        $this->app = $app;

        $getJobsList = function()
        {
            return $this->getJobsList();
        };

        $controller = $app['controllers_factory'];
        $controller->get('/',$getJobsList);

        return $controller;
    }
```

The API

app/ScanJobs/Controller/JobsController.php

```
<?PHP
namespace ScanJobs\Controller;

use Silex\Application;
use Silex\ControllerProviderInterface;
use Silex\ControllerCollection;
use ScanJobs\Model\Job;

class JobsController implements ControllerProviderInterface
{
    protected $app;

    public function connect(Application $app)
    {
        $this->app = $app;

        $getJobsList = function()
        {
            return $this->getJobsList();
        };

        $controller = $app['controllers_factory'];
        $controller->get('/',$getJobsList);

        return $controller;
    }
```

The API

app/ScanJobs/Controller/JobsController.php

```
<?PHP
namespace ScanJobs\Controller;

use Silex\Application;
use Silex\ControllerProviderInterface;
use Silex\ControllerCollection;
use ScanJobs\Model\Job;

class JobsController implements ControllerProviderInterface
{
    protected $app;

    public function connect(Application $app)
    {
        $this->app = $app;

        $getJobsList = function()
        {
            return $this->getJobsList();
        };

        $controller = $app['controllers_factory'];
        $controller->get('/',$getJobsList);

        return $controller;
    }
```

The API

app/ScanJobs/Controller/JobsController.php (continued)

```
    protected function getJobsList()
    {
                $results = Job::fetchJobsList($this->app['db'],'US');
                $payload = array('results' => $results);
                return $this->app->json($payload,200);

            }

        }
```

The API

app/ScanJobs/Controller/JobsController.php (continued)

```
    protected function getJobsList()
    {
                $results = Job::fetchJobsList($this->app['db'],'US');
                $payload = array('results' => $results);
                return $this->app->json($payload,200);

            }

        }
```

The API

- Step 1 - Think it through
- Step 2 - Group your routes
- Step 3 - Build your controllers

Packaging For Deployment

- GitHub is easy
- Deployment to a normal physical or virtual) server is easy
- Deploying to a PaaS is a PITA
- When it doubt, script it.

Packaging For Deployment

```
#!/bin/bash

REPO_NAME="external"
BRANCH_NAME="deploy"

git branch -D $BRANCH_NAME
git checkout -b $BRANCH_NAME

cat .gitignore | grep -v vendor > .gitignore.new
mv .gitignore.new .gitignore

for gitdir in `find ./vendor -name ".git"`
do
        BASENAME=$(dirname $gitdir)
        rm -rf $BASENAME"/.git"
        rm -rf $BASENAME"/.gitignore"
done

chmod -R 775 vendor
git add vendor
git commit -a -m "Integrate vendors for deploy"

git push -f $REPO_NAME $BRANCH_NAME

git checkout master
git branch -D $BRANCH_NAME

composer.phar install
```

Packaging For Deployment

```
#!/bin/bash

REPO_NAME="external"
BRANCH_NAME="deploy"

git branch -D $BRANCH_NAME
git checkout -b $BRANCH_NAME

cat .gitignore | grep -v vendor > .gitignore.new
mv .gitignore.new .gitignore

for gitdir in `find ./vendor -name ".git"`
do
        BASENAME=$(dirname $gitdir)
        rm -rf $BASENAME"/.git"
        rm -rf $BASENAME"/.gitignore"
done

chmod -R 775 vendor
git add vendor
git commit -a -m "Integrate vendors for deploy"

git push -f $REPO_NAME $BRANCH_NAME

git checkout master
git branch -D $BRANCH_NAME

composer.phar install
```

Packaging For Deployment

```
#!/bin/bash

REPO_NAME="external"
BRANCH_NAME="deploy"

git branch -D $BRANCH_NAME
git checkout -b $BRANCH_NAME

cat .gitignore | grep -v vendor > .gitignore.new
mv .gitignore.new .gitignore

for gitdir in `find ./vendor -name ".git"`
do
        BASENAME=$(dirname $gitdir)
        rm -rf $BASENAME"/.git"
        rm -rf $BASENAME"/.gitignore"
done

chmod -R 775 vendor
git add vendor
git commit -a -m "Integrate vendors for deploy"

git push -f $REPO_NAME $BRANCH_NAME

git checkout master
git branch -D $BRANCH_NAME

composer.phar install
```

Packaging For Deployment

```
#!/bin/bash

REPO_NAME="external"
BRANCH_NAME="deploy"

git branch -D $BRANCH_NAME
git checkout -b $BRANCH_NAME

cat .gitignore | grep -v vendor > .gitignore.new
mv .gitignore.new .gitignore

for gitdir in `find ./vendor -name ".git"`
do
        BASENAME=$(dirname $gitdir)
        rm -rf $BASENAME"/.git"
        rm -rf $BASENAME"/.gitignore"
done

chmod -R 775 vendor
git add vendor
git commit -a -m "Integrate vendors for deploy"

git push -f $REPO_NAME $BRANCH_NAME

git checkout master
git branch -D $BRANCH_NAME

composer.phar install
```

Packaging For Deployment

```
#!/bin/bash

REPO_NAME="external"
BRANCH_NAME="deploy"

git branch -D $BRANCH_NAME
git checkout -b $BRANCH_NAME

cat .gitignore | grep -v vendor > .gitignore.new
mv .gitignore.new .gitignore

for gitdir in `find ./vendor -name ".git"`
do
        BASENAME=$(dirname $gitdir)
        rm -rf $BASENAME"/.git"
        rm -rf $BASENAME"/.gitignore"
done

chmod -R 775 vendor
git add vendor
git commit -a -m "Integrate vendors for deploy"

git push -f $REPO_NAME $BRANCH_NAME

git checkout master
git branch -D $BRANCH_NAME

composer.phar install
```

Packaging For Deployment

```
#!/bin/bash

REPO_NAME="external"
BRANCH_NAME="deploy"

git branch -D $BRANCH_NAME
git checkout -b $BRANCH_NAME

cat .gitignore | grep -v vendor > .gitignore.new
mv .gitignore.new .gitignore

for gitdir in `find ./vendor -name ".git"`
do
        BASENAME=$(dirname $gitdir)
        rm -rf $BASENAME"/.git"
        rm -rf $BASENAME"/.gitignore"
done

chmod -R 775 vendor
git add vendor
git commit -a -m "Integrate vendors for deploy"

git push -f $REPO_NAME $BRANCH_NAME

git checkout master
git branch -D $BRANCH_NAME

composer.phar install
```

Packaging For Deployment

```
#!/bin/bash

REPO_NAME="external"
BRANCH_NAME="deploy"

git branch -D $BRANCH_NAME
git checkout -b $BRANCH_NAME

cat .gitignore | grep -v vendor > .gitignore.new
mv .gitignore.new .gitignore

for gitdir in `find ./vendor -name ".git"`
do
        BASENAME=$(dirname $gitdir)
        rm -rf $BASENAME"/.git"
        rm -rf $BASENAME"/.gitignore"
done

chmod -R 775 vendor
git add vendor
git commit -a -m "Integrate vendors for deploy"

git push -f $REPO_NAME $BRANCH_NAME

git checkout master
git branch -D $BRANCH_NAME

composer.phar install
```

Packaging For Deployment

```
#!/bin/bash

REPO_NAME="external"
BRANCH_NAME="deploy"

git branch -D $BRANCH_NAME
git checkout -b $BRANCH_NAME

cat .gitignore | grep -v vendor > .gitignore.new
mv .gitignore.new .gitignore

for gitdir in `find ./vendor -name ".git"`
do
        BASENAME=$(dirname $gitdir)
        rm -rf $BASENAME"/.git"
        rm -rf $BASENAME"/.gitignore"
done

chmod -R 775 vendor
git add vendor
git commit -a -m "Integrate vendors for deploy"

git push -f $REPO_NAME $BRANCH_NAME

git checkout master
git branch -D $BRANCH_NAME

composer.phar install
```

Wrap Up

- Silex is still not a micro-framework
- Silex is useful
- Silex is powerful
- Silex is not **the** answer.
  (but it is one of the answers)

Questions?

Cal Evans

```
{
 "e":"cal@calevans.com",
 "t":"@calevans",
 "b":"http:\/\/blog.calevans.com"
}
```

https://github.com/calevans/scanjobs

https://joind.in/7094