Feature Flags with Symfony

Symfony Live Berlin 2014

Benjamin Eberlei <benjamin@qafoo.com>

30.10.2014

Qafoo

passion for software quality

Me

- Working at Qafoo

We promote high quality code with trainings and consulting

`http://qafoo.com`

- Doctrine Developer
- Symfony Contributor
- Twitter `@beberlei` and `@qafoo`

Outline

Introduction

```php
<?php

if (is_feature_enabled('awesome_sauce')) {
        awesome_sauce();
} else {
        boring_sauce();
}
```

One Year later

```
<?php

if (is_fea     enabled('      ome_sauce')) {
    /* if (i     ture_en      d('awesome_sauce_v3')) {
        fancy      ce()
    }*/
    if (is_featu       led('awesome_sauce_v4')) {
        if (is_fe      _enabled('crazy_sauce')) {
            cra           ();
        } els
               y_sauc
        }
    } els
           ome_sauce();
    }
} else {
    boring_sauce();
}
```

Qafoo
passion for software quality

Lets start from the beginning!

- ”Flipping Out” by Flickr (2009)
- ”FeatureToggle” by Martin Fowler (2010)
- Names
    - Flags
    - Toggles
    - Flippers
    - Switches

Definition

Martin Fowler on FeatureToggles:

The basic idea is to have a ***configuration file*** *that defines a* ***bunch of toggles*** *for various* ***pending features****.*
The ***running application*** *then uses these toggles in order to decide whether or not to show the new feature.*

Feature Flags are branching

Branching?

```
$ git branch awesome_sauce master
$ git checkout awesome_sauce
```

Branches for Features Flags

```php
<?php
// branch "master"
boring_sauce();
```

```php
<?php
// branch "awesome_sauce"
awesome_sauce();
```

```php
<?php
// branch "crazy_sauce"
crazy_sauce();
```

Qafoo
passion for software quality

Feature Flags vs VCS Branches

Feature Flags allow arbitrary combination of branches

VCS don't have this flexibility!

- Allow trunk-based development
- Increase complexity

Outline

API for Feature Flags

```php
<?php

interface FeatureFlags
{
    function isEnabled($flag);
}
```

Hardcoded Feature Flags

```php
<?php
class HardcodedFlags implements FeatureFlags
{
    public function isEnabled($flag)
    {
        if ($flag === 'billing') {
            return true;
        }

        return false;
    }
}
```

passion for software quality

Implementation

- Symfony Configuration
- SQL-Database
- Redis
- Any kind of implementation is usually simple.

Feature Flags Service

```xml
<service
    id="feature_flags"
    class="Acme\DemoBundle\Util\EnvFlags">

   <argument>%kernel.environment%</argument>
</service>
```

Outline

Design Considerations

- Avoid if/elseif/else hell
- Maintainable Solution
  - Cleanup old code
  - Cleanup deprecated flags
- Integrate nicely into Symfony
- Reusable, generic solutions preferred

Solution

Move all toggle decisions outside of your code

Integration Points

- Twig Templates
- Routing
- Controllers
- Services
- Event Listeners

Twig Templates

Decide what a user can see

- Show Links
- Load Sub-Controllers

Twig Templates

```
{% if is_feature_enabled('billing') %}
    <a href="{{ path('billing') }}">Pay</a>
{% endif %}
```

Twig Templates

```
{% if is_feature_enabled('billing') %}
    {{ render(controller(
        "AcmeDemoBundle:Billing:show"))
    }}
{% endif %}
```

Routing

Decide what a user can access

- Conditional routes
- Show 404 if it the feature is disabled

Routing

```
my_bundle_awesome_sauce:
    pattern: /awesome
    defaults:
        _feature_flag: awesome_sauce
```

passion for software quality

Routing: EventListener

```php
<?php
public function onKernelRequest($event)
{
    $request = $event>getRequest();
    $flag = $request->attributes
        ->get('_feature_flag');

    if (!$this->features->isEnabled($flag)) {
        throw new NotFoundHttpException();
    }
}
```

passion for software quality

Selecting Controllers

Decide what controller is called

- Execute different actions based on flags
- Manipulate Controller Resolver

Deciding about Controllers

```
hello:
  pattern: /hello/{name}
  defaults:
    _controller: "AcmeDemoBundle:Default:hello"
    _alternative: "AcmeSuperBundle:Default:hello"
    _when_feature: super_hello
```

Deciding about Controllers

```php
<?php

public function onKernelRequest($event)
{
    // ...
    if ($this->features->isEnabled($whenFlag)) {
        $request->attributes->set(
            '_controller',
            $alternative
        );
    }
}
```

passion for software quality

Deciding about Services

Decide what business logic is called

- Construct different services based on feature flags
- Requires a common interface the services implement
- Interface Segregation (SOLI**D** principles)

Symfony Dependency Injection

- Delegate construction of a service to a factory
- Use `factory-service` and `factory-method`
- Implement a generic Factory for the task only once

Feature Flag Service Factory

```php
<?php
class FeatureFlagFactory
{
    private $container;

    public function create($when, $then, $else)
    {
        return $this->flags->isEnabled($when)
            ? $this->container->get($then)
            : $this->container->get($else);
    }
}
```

Feature Flag Service Definition

```xml
<service id="feature_flag_factory"
    class="Acme\DemoBundle\FeatureFlagFactory">

    <argument type="service"
        id="service_container" />
</service>
```

passion for software quality

Feature Flag Service

```
<service id="my_service" class=".."
      factory-service="feature_flag_factory"
      factory-method="create">

      <argument>awesome_sauce</argument>
      <argument>my_service.awesome_sauce</argument>
      <argument>my_service.boring_sauce</argument>
</service>
```

Using the Feature Flag Service

```php
<?php

public function helloAction()
{
    $service = $this->get('my_service');
    // ...
}
```

passion for software quality

Deciding about EventListeners

Decide what event listeners are called

- Add a custom event attribute tag for feature flags.
- Make sure listeners are only called when flag is enabled.

Event Listener Tags

```
<service id="my_event_listener" class="..">
    <!-- ... -->

    <tag name="kernel.event_listener"
        event="kernel.request"
        if-feature-enabled="awesome_sauce" />

</service>
```

Event Listener

Homework!

Hint: It is quite complicated to do this generically.

Simple Solution

```
class AwesomeListener
{
    public function onKernelRequest($event)
    {
        if (!$this->features->isEnabled('awesome')) {
            return;
        }

        // ...
    }
}
```

Outline

passion for software quality

What about Context?

- A dynamic feature flag system needs context.
  - User Information
  - Request Information
- Gather very early in `kernel.request` event.
- Obviously before any dynamic feature flag is used.

API with Context

```php
<?php

interface FeatureFlags
{
    function setContext($variable, $value);
    function isEnabled($flag);
}
```

Gather Context

```php
<?php

public function onKernelRequest($event)
{
    // ...
    $this->featureFlags->setContext(
        'user_id',
        $user->getId()
    );
    $this->featureFlags->setContext(
        'ip_address',
        $request->getClientIp()
    );
}
```

Qafoo
passion for software quality

Outline

Introduction

Building the Foundation

Using Feature Flags

Context

Advanced Topics

A/B Testing

- Consider feature toggles activation of a small experiment
- Let 50% of users see the new feature
- Measure success of the new variant compared to the old
- Decide to keep the old or switch to the new variant

A/B Testing: Technical Requirements

- Feature Toggles need to be at least user group based
- Measurable success criteria for new feature
- Multi-Armed bandit algorithm for evaluation

Circuit Breaker

- Consider dynamic features toggle to deactive defunct backends
- Example: Deactivate Search when Elasticsearch is down
- Requires feature toggle to be always present in code
- Requires datastorage to measure number of failures of backend services.

Links

- http://code.flickr.net/2009/12/02/flipping-out/
- http://martinfowler.com/bliki/FeatureToggle.html
- http://labs.qandidate.com/blog/2014/09/04/feature-toggles-in-symfony2/

Qafoo
passion for software quality

https://joind.in/10287

Qafoo
passion for software quality

THANK YOU

Rent a quality expert
qafoo.com