



Multi-site et routing

Alexandre Salomé - Symfony Live 2014 - Paris



Alexandre Salomé

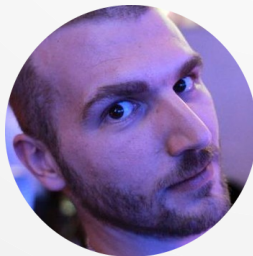
Consultant technique pour SensioLabs Lille

2 ans de symfony1

3 ans de Symfony2

@alexandresalome

sensiolabs.com



Motoblouz

Leader de la vente en ligne de matériel pour **moto**

Nouveau site en Symfony2

SensioLabs pour la formation et l'audit

<http://motoblouz.com>

Multi-sites

- Plusieurs marques
- Sur chaque marque, une ou plusieurs langues disponibles
- Spécificités pour chaque site

Notre cas pratique

- Velojazz
 - Site français : velojazz.fr
 - Site anglais : velojazz.com/uk
 - Site allemand : velojazz.com/de
- Bikomatic
 - Site français : fr.bikomatic.com
 - Site anglais : en.bikomatic.com
- Roulibre
 - Site allemand : roulibre.de

Des URLs différentes

- <http://velojazz.fr/inscription>
- <http://velojazz.com/uk/register>
- <http://velojazz.com/de/registrieren>

Des vues différentes

- Feuilles de style spécifiques
- Blocs différents (en-tête, pied de page, pages de contenu)

Des comportements différents

- Sur Roulibre, l'inscription n'est pas disponible
- Sur Bikomatic, le mail de contact est différent

THINKING...



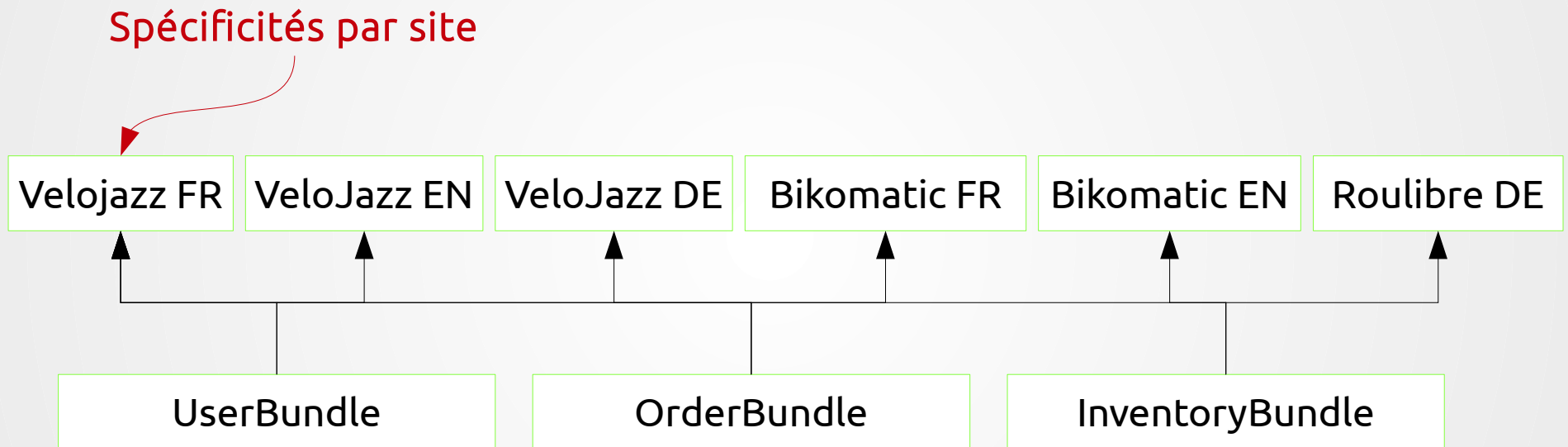
Please wait...

Étude

Stratégie A : plusieurs applications

- Bundle = Fonctionnalité commune aux différents sites
- Une Application = Une marque
 - Surcharge au niveau de l'application
 - Configuration pour les spécificités de chaque site

Stratégie A : plusieurs applications



Stratégie A : plusieurs applications

- 1 nouveau site = 1 nouvelle application
- Traitements inter-applicatifs pénibles
 - Génération des assets
 - Notifications par e-mail
- Déploiements pénibles
- Sur-découpage des dépôts

Danger !

Stratégie B : une seule application !

- Multi-site configuré dans l'application
- Un seul dépôt = Meilleur cohésion
- N'empêche pas le découplage et la réutilisabilité
- Meilleures maintenance et déploiements

**Très bon
qualité !**

Stratégie B : une seule application !

Spécificités par site

Velojazz – Bikomatic – Roulibre

UserBundle

OrderBundle

InventoryBundle

MultisiteBundle



MultisiteBundle

Disponible sur Github



<http://github.com/alexandresalome/multisite-bundle>

Fournisseur / Consommateur

- Permet de séparer notre problème en deux parties



README

Configurez vos sites dans *config.yml*:

```
alex_multisite:
  default_branding: velojazz
  default_locale:   fr_FR
  brandings:
    velojazz:
      fr_FR: { host: velojazz.fr }
      en_GB: { host: velojazz.com, prefix: /uk }
      de_DE: { host: velojazz.com, prefix: /de }
```

README

```
# config.yml
alex_multisite:
  default_branding: velojazz
  default_locale:   fr_FR
  brandings:
    velojazz:
      fr_FR: { host: %host_fr% }
      en_GB: { host: %host_others%, prefix: /uk }
      de_DE: { host: %host_others%, prefix: /de }

# parameters.yml
host_fr:      velojazz.fr
host_others:  velojazz.com
```

README

Utilisez l'annotation suivante :

```
/**
 * @Route(name="_demo", paths={
 *     "fr_FR"="/ceci-est-un-exemple",
 *     "en_GB"="/this-is-an-example",
 *     "de_DE"="/dies-ist-ein-beispiel"
 * })
 */
public function demoAction()
{
    return array();
}
```

```
/**
 * @Route(name="_demo", paths={
 *     "velojazz"= {
 *         "fr_FR"="/velojazz-et-vous",
 *         "en_GB"="/velojazz-and-you",
 *         "de_DE"="/velojazz-und-du"
 *     },
 *     "bikomatic"= ...
 *     "roulibre"= ...
 * })
 */
```

Utilisation depuis Twig

Générer des URLs vers des sites différents :

```
{# same site, same locale #}  
{{ path('_demo') }}
```

```
{# different site #}  
{{ path('_demo', {_branding: 'roulibre'}) }}
```

```
{# different locale #}  
{{ path('_demo', {_locale: 'de_DE'}) }}
```

Bootstrap du MultisiteBundle

- README pour expliquer le fonctionnement
- Classe `AlexMultisiteBundle`
- `DependencyInjection\`
 - `Configuration` pour contraindre la configuration
 - `AlexMultisiteExtension` pour transformer en définition de services
- `Branding\`
 - `Branding` pour représenter une marque
 - `SiteContext` pour l'accès au modèle multi-sites

```
$velojazz = new Branding('velojazz', array(
    'fr_FR' => array('host' => 'velojazz.fr'),
    'en_GB' => array('host' => 'velojazz.com', 'prefix' => '/uk'),
    'de_DE' => array('host' => 'velojazz.com', 'prefix' => '/de'),
));

# ...

$siteContext = new SiteContext(
    array($velojazz, $bikomatic, $roulibre),
    'velojazz',
    'fr_FR'
);

$siteContext->getCurrentBrandingName();
$siteContext->getCurrentLocale();
```

Bilan

- README = notre contrat
- Transformation de la configuration en service
 - Voir [DependencyInjection\AlexMultisiteExtension](#)
- Un service principal : `site_context`
 - Paramètres par défaut
 - Configuration des sites
 - Notre point d'entrée dans le modèle multisites



Étendre le routing

JMSI18nRoutingBundle

- Surcharge du routing
 - Préfixe __<LOCALE>__ sur les routes
 - Chargement des routes par annotations, de Yaml et Xml
 - Génération des URLs
- Transparent à l'utilisation
- Voir le [JMSI18nRoutingBundle](#)

Inspirons-nous en !

Créer une classe d'annotation

- Étendre l'annotation standard
 - Héritage de l'existant
- Ajout d'une nouvelle option *paths*
 - Tableau à 1 niveau = locale → path
 - Tableau à 2 niveaux = branding → locale → path
- Voir [Annotation\Route](#)

Chargement des annotations

- Fait par sensio/framework-extra-bundle
 - Transformation Annotation → Route
- Router\Loader\AnnotatedRouteControllerLoader

Surcharge du FrameworkExtraBundle

On crée un fichier de service dans *MultisiteBundle* avec dedans la surcharge de la classe de chargement des annotations :

```
<!-- Resources/config/framework_extra.xml -->
<parameters>
    <parameter key="sensio_framework_extra.routing.loader.annot_class.class">
        Alex\MultisiteBundle\Router\Loader\AnnotatedRouteControllerLoader
    </parameter>
</parameters>
```

```
namespace Alex\MultisiteBundle\Router\Loader;

use Sensio\...\AnnotatedRouteControllerLoader as BaseAnnotatedRouteControllerLoader;
use Alex\MultisiteBundle\Annotation\Route as RouteAnnotation;

class AnnotatedRouteControllerLoader extends BaseAnnotatedRouteControllerLoader
{
    protected function addRoute(RouteCollection $collection, $annot, ...)
    {
        // If the annotation is not the Multisite's one, call parent method
        if (!$annot instanceof RouteAnnotation) {
            return parent::addRoute($collection, $annot, $globals, $class, $method);
        }

        // our business here
    }
}
```

```
namespace Alex\MultisiteBundle\Router\Loader;
// ...
class AnnotatedRouteControllerLoader extends BaseAnnotatedRouteControllerLoader
{
    protected $siteContext;

    /**
     * @return AnnotatedRouteControllerLoader
     */
    public function setSiteContext(SiteContext $siteContext)
    {
        $this->siteContext = $siteContext;
    }

    // ...
}
```

Injection du contexte dans les services

- Passe du conteneur de services
 - InjectSiteContextPass
- Injection du service site_context

```
$container  
->getDefinition('sensio_framework_extra.routing.loader.annot_class')  
->addMethodCall('setSiteContext', array(new Reference('site_context')))  
;
```

Extension du routeur

- `match(string $pathinfo)`
 - Injecter dans *SiteContext* le site actuel
- `generate($name, $parameters, $absolute)`
 - Préfixer le nom de route
- `getRouteCollection()`
 - Grouper par branding et par locale
- Voir **MultisiteRouter**

```

if (preg_match('#^foo\\.example\\.org$#s', $host, $hostMatches)) {
    if (0 === strpos($pathinfo, '/fr')) {
        // _i18n__foo_fr_FR__test_locale
        if ($pathinfo === '/fr/page-fr') {
            return array ( '_branding' => 'foo', '_locale' => 'fr_FR', '_controller' => 'Alex\\Mu
        }

        // _i18n__foo_fr_FR__test_branding_locale
        if ($pathinfo === '/fr/test-foo-fr') {
            return array ( '_branding' => 'foo', '_locale' => 'fr_FR', '_controller' => 'Alex\\Mu
        }
    }

    // _i18n__foo_en_GB__test_locale
    if ($pathinfo === '/page-en') {
        return array ( '_branding' => 'foo', '_locale' => 'en_GB', '_controller' => 'Alex\\Multis
    }

    // _i18n__foo_en_GB__test_branding_locale
    if ($pathinfo === '/test-foo-en') {
        return array ( '_branding' => 'foo', '_locale' => 'en_GB', '_controller' => 'Alex\\Multis
    }
}

```

Voilà !

Nos routes sont internationalisées



Configuration par site

Spécification

- Paramètres différents pour chaque site
 - Nom du site, e-mail de contact, ...
 - Activation/ désactivation de fonctionnalités
- 3 niveaux de configuration
 - Configuration globale
 - Configuration par *branding*
 - Configuration par site

Utilisation

```
alex_multisite:
  brandings:
    defaults:
      feature_register: true
      contact_email:    contact-team@velojazz.com
  roulibre:
    defaults:
      contact_email:    roulibre-team@velojazz.com
  de_DE: { host: ..., feature_register: false }
```

Ajout dans SiteContext

- « host » et « prefix » deviennent des options
- L'utilisateur peut donner autant d'options qu'il veut
- Ajout des méthodes sur le service **SiteContext** :
 - `$ctx->getOption($name)`
 - `$ctx->getBranding($branding)->getOption($locale, $name);`
- À la construction du conteneur
 - Résolution de la cascade
 - Injection dans les objets Branding

Utilisation

```
{# Dans un template #}  
{{ site_context.option('title') }}  
  
// Dans un contrôleur  
$this  
    ->get('site_context')  
    ->getOption('title')  
;
```

Bilan

- Extension du modèle existant (ajout d'options)
- API d'utilisation simple et concise
- Cascade de configuration faite à la construction du conteneur
- Peu de code !

Templating

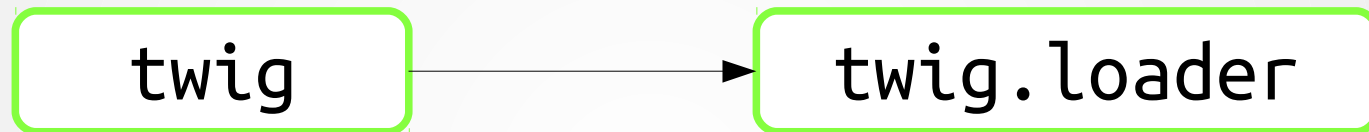
Stratégie A : Loader spécifique

- Les templates sont préfixés s'ils sont « spéciaux »
 - Template classique : `AcmeDemoBundle::layout.html.twig`
 - Template « spécial » : `site://logo.html.twig`
- De là, définir la logique de chargement
 - *Twig_LoaderInterface*
- Création d'un service dans le conteneur + tag
- Voir http://symfony.com/doc/current/reference/dic_tags.html#twig-loader

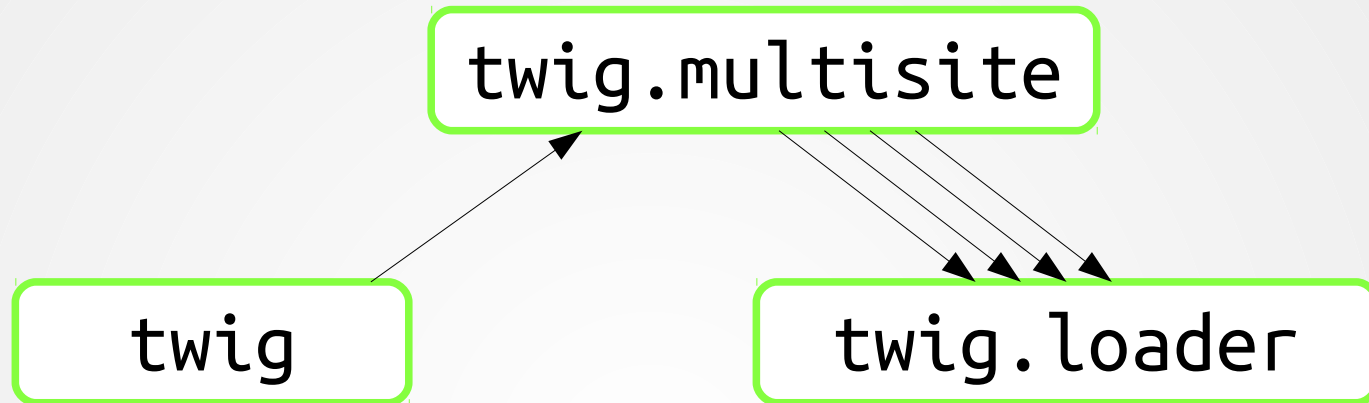
Stratégie B : Surcharger le loader

- Pour chaque template, on essaie de trouver des templates spécifiques pour la marque/la locale actuelle :
 - AcmeDemoBundle::logo.html.twig
 - AcmeDemoBundle::_velojazz_fr_FR/logo.html.twig
 - AcmeDemoBundle::_roulibre_/logo.html.twig
 - AcmeDemoBundle::__fr_FR/contact.html.twig
- Préfixe *_branding_media*

Surcharge du service Twig



Surcharge du service Twig



Surcharge du service Twig

- Création d'une classe d'encapsulation
 - Router\MultisiteLoader
 - Se charge de distribuer l'appel
- Modification par passe de compilateur
 - Twig\TwigLoaderPass

Bilan

- Léger surcoût à l'utilisation de Twig
- Peu de lignes de code



<https://flic.kr/p/aTB844>

Assurance qualité

Couverture des tests

- Tests fonctionnels (via AppKernel + BrowserKit)
 - `TemplatingTest`
 - `RoutingTest`
 - `ConfigTest`
- Tests unitaires
 - Modèle de site : `BrandingTest` et `SiteContextTest`
- Travis CI

Conclusion

- Utilisez la logique fournisseur/consommateur
- Simplifiez les fonctionnalités transverses
 - Keep It Simple, Stupid
- Inspirez-vous des bundles similaires
- Testez de manière agnostique du métier

Questions ?

Merci !

Crédits

- **JMSI18nRoutingBundle** pour l'inspiration
<https://github.com/schmittjoh/JMSI18nRoutingBundle>
- **Photos** creative commons
 - Randolph Rautenberg - <https://flic.kr/p/6d1iRf>
 - Oran Viriyincy - <https://flic.kr/p/7Z98DL>
 - Riccardo Bandiera - <https://flic.kr/p/2a8G18>
 - Chris Waits - <http://flic.kr/p/avSWXM>
 - Canned Tuna – <https://flic.kr/p/aTB844>
 - Touring Club - <https://flic.kr/p/afHQxo>