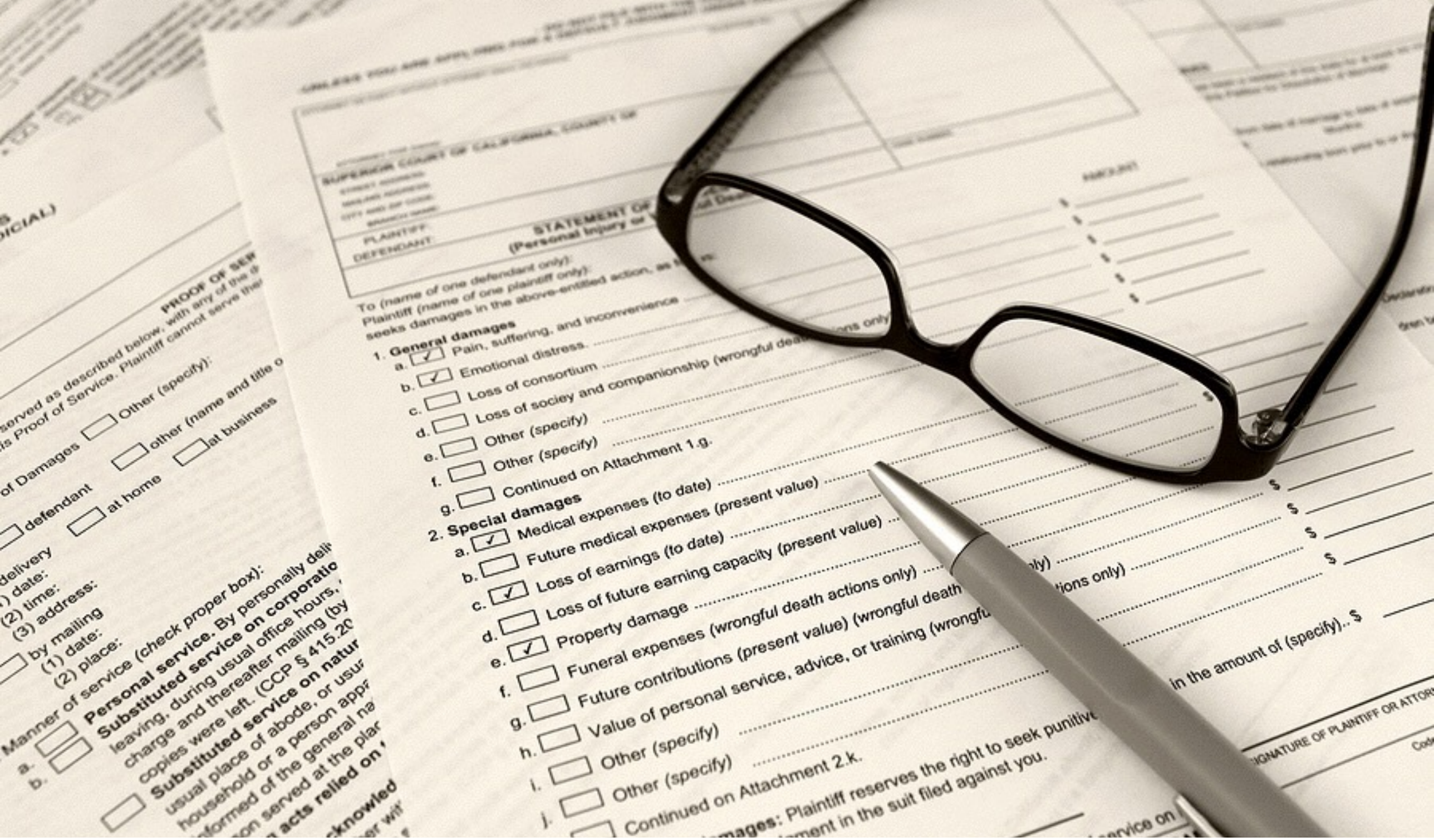


https://www.flickr.com/photos/orlandolawyer/6120103619/sizes/o/in/photostream/

Techniques Avancées avec les formulaires Symfony

SymfonyLive PARIS 2015 9-10 AVRIL

SensioLabs

Au Programme

- Types de formulaire en service
- Héritage dynamique versus héritage statique
- Convertisseurs de données
- Événements / Écouteurs / Event Subscriber
- Extension de Types de formulaire
- Formulaires imbriqués et
- Rendu avec Twig

Problématiques

Conception d'un formulaire

On le traduit comme ça

Objet métier (domain object pour les intimes)

```php
namespace AppBundle\Entity;

class Article
{
    private $title;
    private $tags;
    private $publishedAt;

    public function __construct($title, array $tags, $publishedAt)
    {
        $this->title = $title;
        $this->tags  = $tags;

        if (!$publishedAt instanceof \DateTime) {
            $publishedAt = new \DateTime($publishedAt);
        }

        $this->publishedAt = $publishedAt;
    }
}
```

ArticleType (définition des éléments du formulaire)

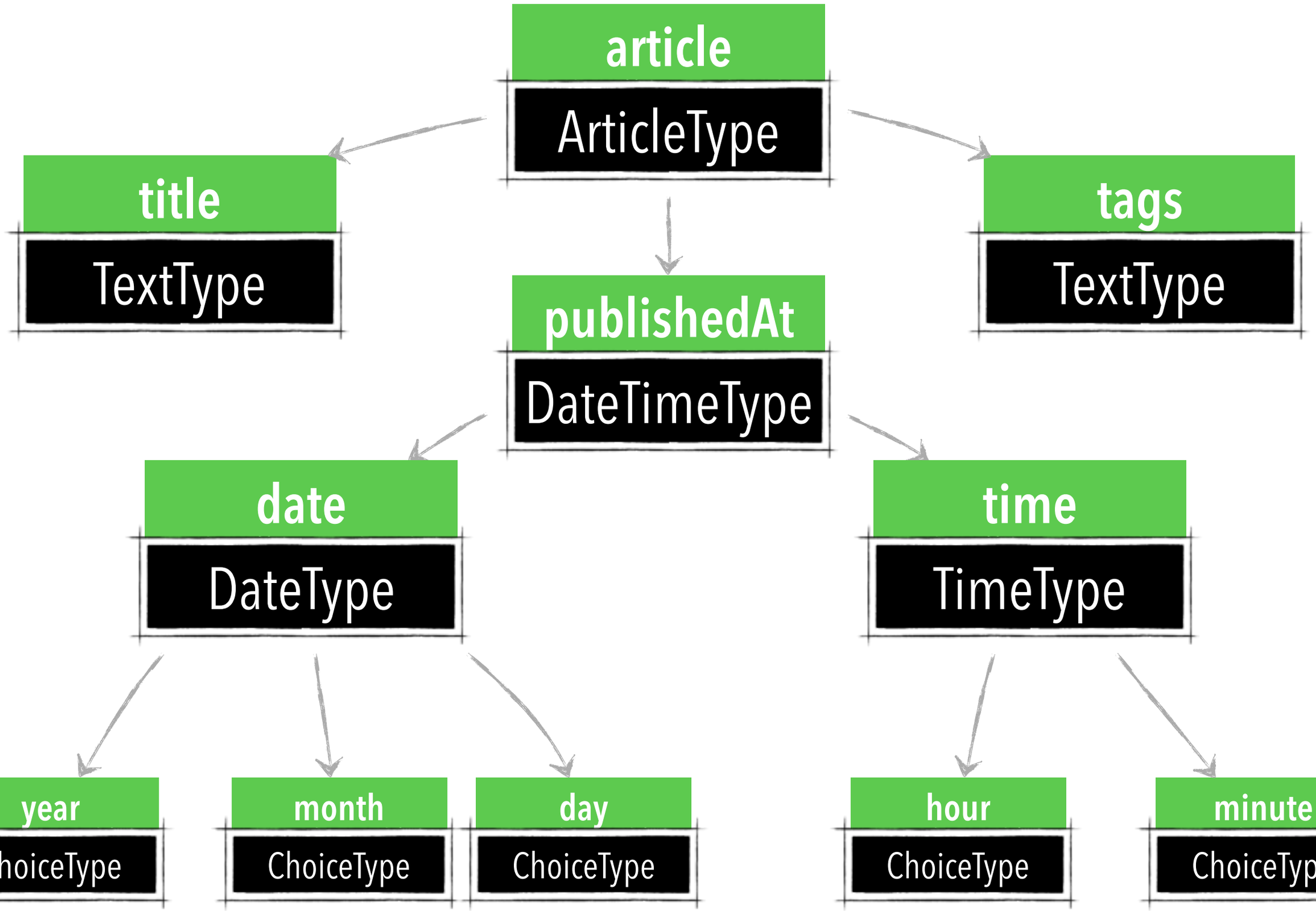
```php
namespace AppBundle\Form;

use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;

class ArticleType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('title')
            ->add('tags')
            ->add('publishedAt', 'datetime')
        ;
    }

    // ...
}
```

Le contrôleur (utilisation du formulaire)

```php
namespace AppBundle\Controller;

// use ...
class ArticleController extends Controller
{
    /** @Template */
    public function newAction(Request $request)
    {
        $article = new Article('The Symfony Book', [ 'php', 'http' ], '2014-04-08 12:00');

        $form = $this->createForm(new ArticleType(), $article);
        $form->handleRequest($request);

        if ($form->isValid()) {
            // process the data
        }

        return [ 'form' => $form->createView() ];
    }
}
```

Du point de vue de l'humain

Du point de vue de Symfony



Construction d'un formulaire

La notion de type

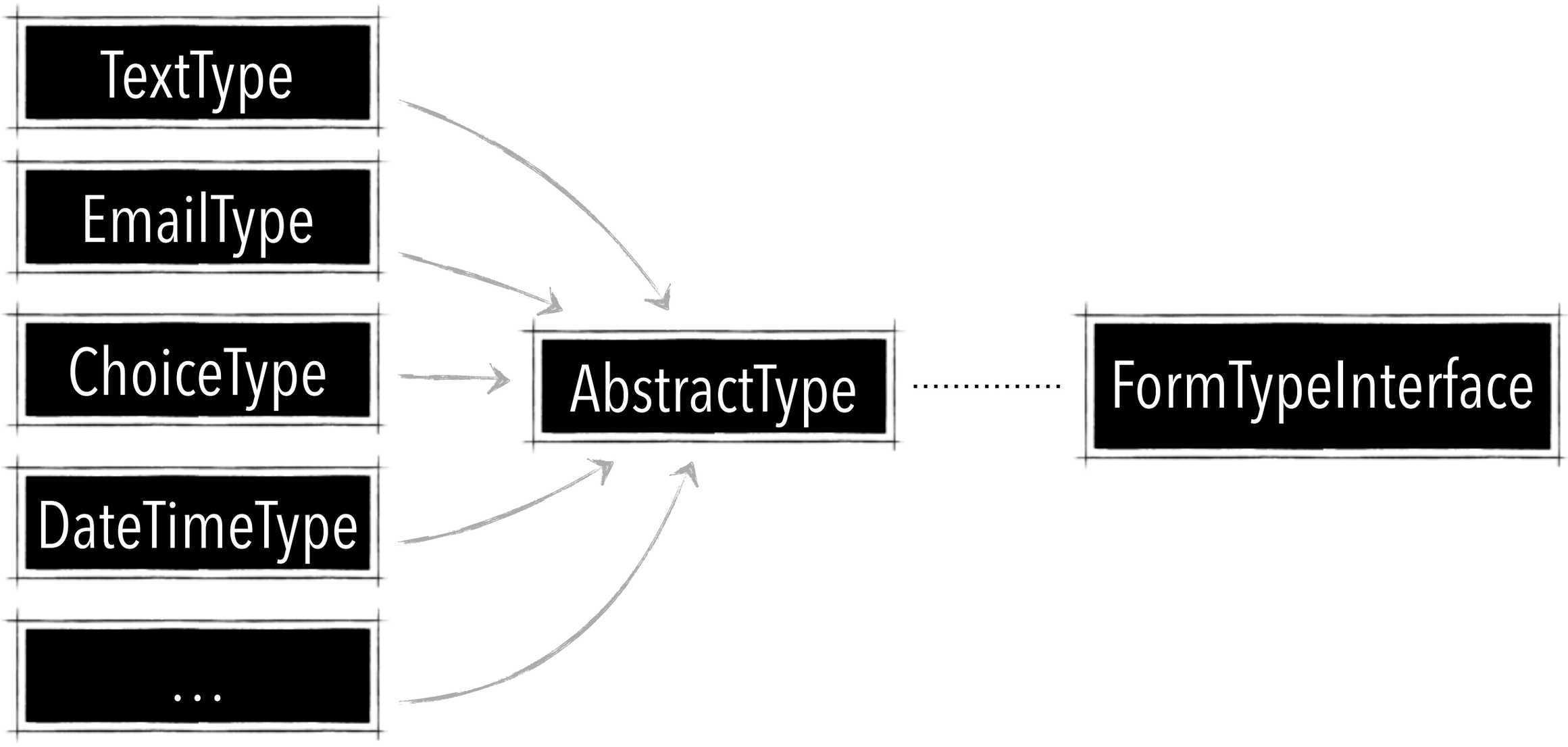
Un **type** est une structure unique qui **définit** et **configure** un **élément** du formulaire

$ php app/console container:debug | grep form.type

```
form.type.birthday          container Symfony\Component\Form\Extension\Core\Type\BirthdayType
form.type.button            container Symfony\Component\Form\Extension\Core\Type\ButtonType
form.type.checkbox          container Symfony\Component\Form\Extension\Core\Type\CheckboxType
form.type.choice            container Symfony\Component\Form\Extension\Core\Type\ChoiceType
form.type.collection        container Symfony\Component\Form\Extension\Core\Type\CollectionType
form.type.country           container Symfony\Component\Form\Extension\Core\Type\CountryType
form.type.currency          container Symfony\Component\Form\Extension\Core\Type\CurrencyType
form.type.date              container Symfony\Component\Form\Extension\Core\Type\DateType
form.type.datetime          container Symfony\Component\Form\Extension\Core\Type\DateTimeType
form.type.email             container Symfony\Component\Form\Extension\Core\Type\EmailType
form.type.entity            container Symfony\Bridge\Doctrine\Form\Type\EntityType
form.type.file              container Symfony\Component\Form\Extension\Core\Type\FileType
form.type.form              container Symfony\Component\Form\Extension\Core\Type\FormType
form.type.hidden            container Symfony\Component\Form\Extension\Core\Type\HiddenType
form.type.integer           container Symfony\Component\Form\Extension\Core\Type\IntegerType
form.type.language          container Symfony\Component\Form\Extension\Core\Type\LanguageType
form.type.locale            container Symfony\Component\Form\Extension\Core\Type\LocaleType
form.type.money             container Symfony\Component\Form\Extension\Core\Type\MoneyType
form.type.number            container Symfony\Component\Form\Extension\Core\Type\NumberType
form.type.password          container Symfony\Component\Form\Extension\Core\Type\PasswordType
form.type.percent           container Symfony\Component\Form\Extension\Core\Type\PercentType
form.type.radio             container Symfony\Component\Form\Extension\Core\Type\RadioType
form.type.repeated          container Symfony\Component\Form\Extension\Core\Type\RepeatedType
form.type.reset             container Symfony\Component\Form\Extension\Core\Type\ResetType
form.type.search            container Symfony\Component\Form\Extension\Core\Type\SearchType
form.type.submit            container Symfony\Component\Form\Extension\Core\Type\SubmitType
form.type.text              container Symfony\Component\Form\Extension\Core\Type\TextType
form.type.textarea          container Symfony\Component\Form\Extension\Core\Type\TextareaType
form.type.time              container Symfony\Component\Form\Extension\Core\Type\TimeType
form.type.timezone          container Symfony\Component\Form\Extension\Core\Type\TimezoneType
form.type.url               container Symfony\Component\Form\Extension\Core\Type\UrlType
```

Les types natifs sont enregistrés en tant que services avec le tag « ***form.type*** »

Héritage statique : Structure des types



Héritage dynamique de type

```
class DateType extends AbstractType
{
    public function getParent()
    {
        return 'form';
    }
}

class BirthdayType extends AbstractType
{
    public function getParent()
    {
        return 'date';
    }
}
```

http://cdn-parismatch.ladmedia.fr

Comment ça fonctionne ?

Revenons à notre contrôleur

```php
class ArticleController extends Controller
{
    /** @Template */
    public function newAction(Request $request)
    {
        $article = new Article('...');

        $form = $this->createForm(new ArticleType(), $article);
        $form->handleRequest($request);

        // ...
    }
}
```

Dans un premier temps

```php
$form = $this->createForm(new ArticleType(), $article);
```

ou

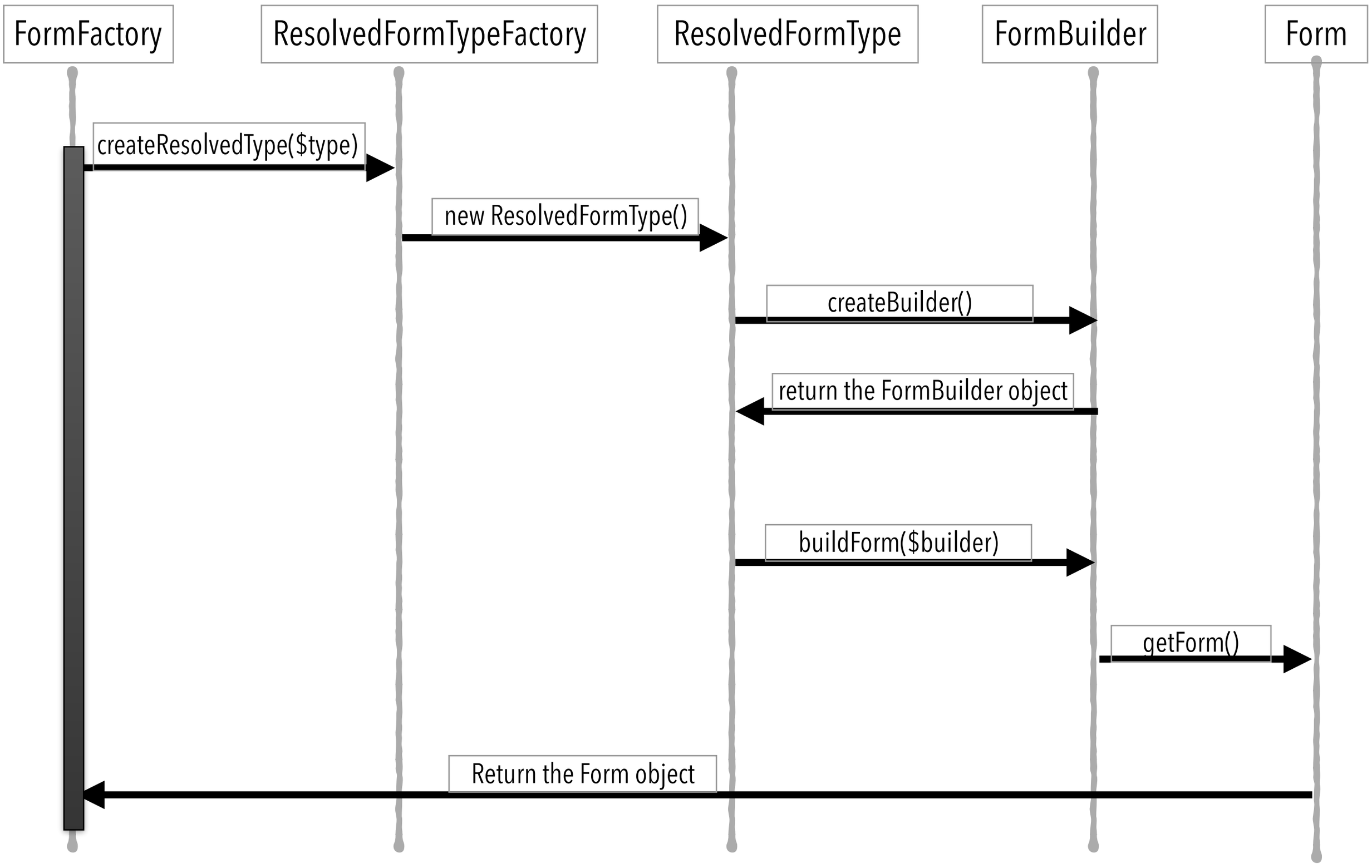
```php
$form = $this->createForm('article', $article);
```

ou

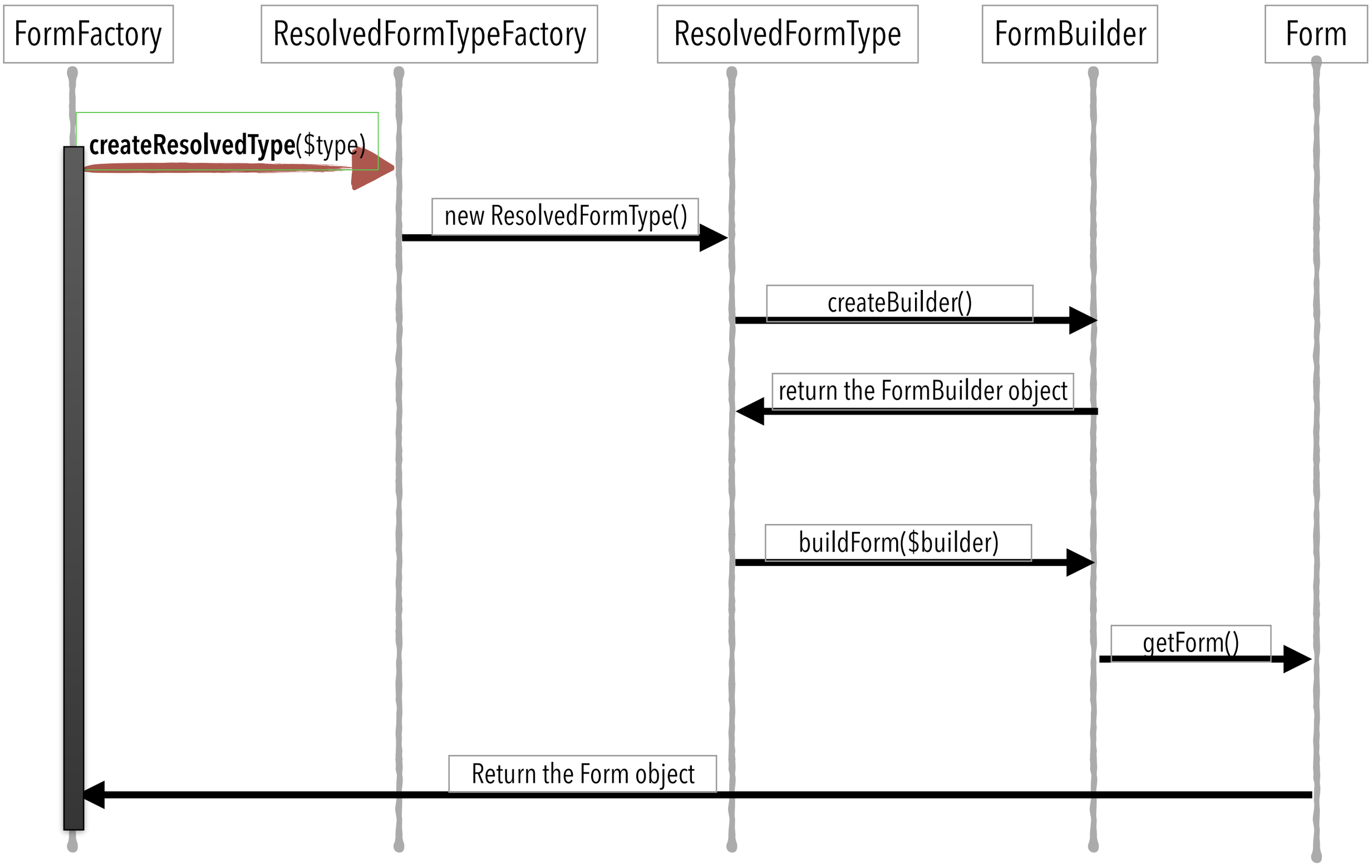
```php
$this->container->get('form.factory')->create($type, $data, $options);
```

Symfony\Component\Form\FormFactory

Vue d'ensemble



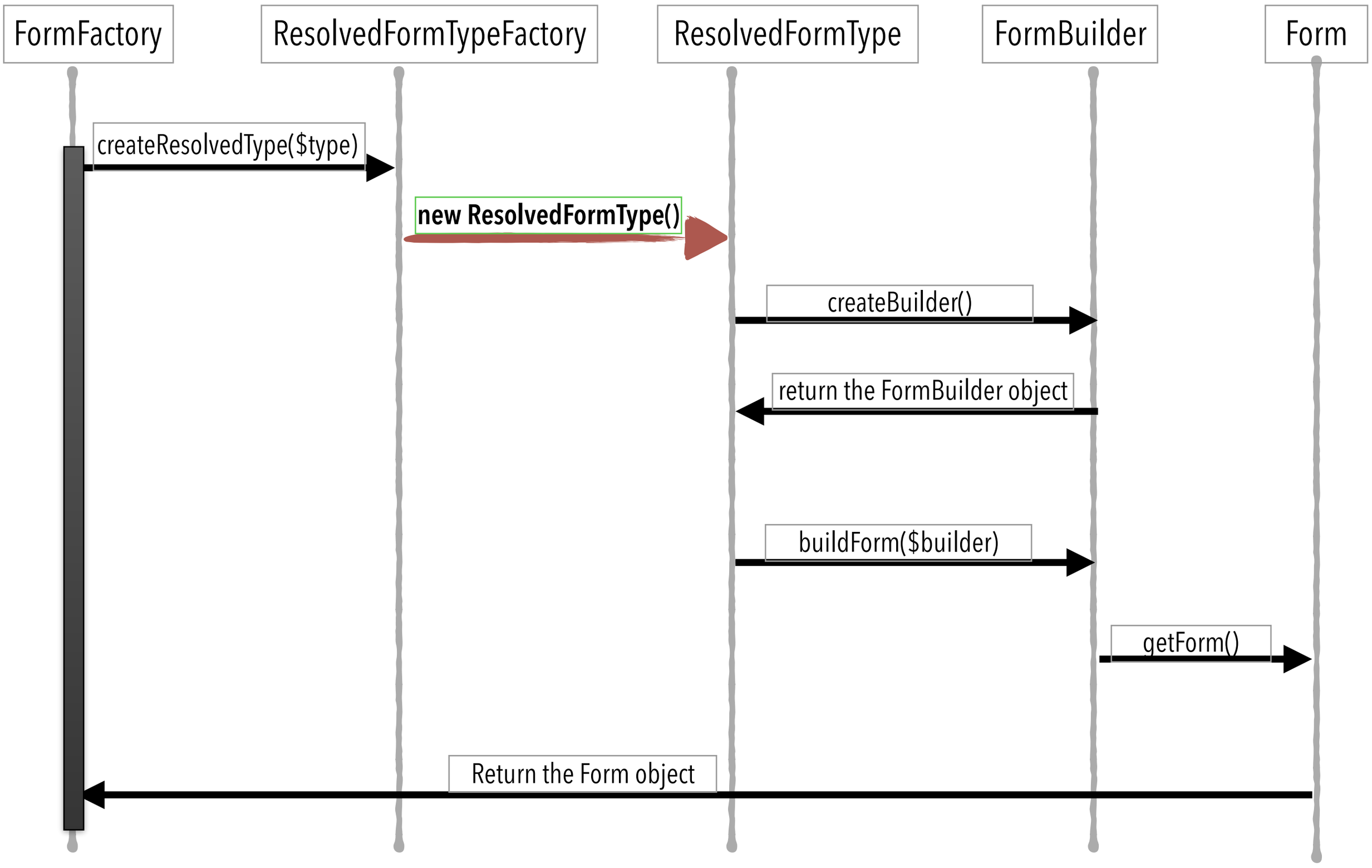
Vue d'ensemble



Résolution dynamique d'un type

```php
namespace Symfony\Component\Form;

class ResolvedFormTypeFactory implements ResolvedFormTypeFactoryInterface
{
    public function createResolvedType(
        FormTypeInterface $type,
        array $typeExtensions,
        ResolvedFormTypeInterface $parent = null
    )
    {
        return new ResolvedFormType($type, $typeExtensions, $parent);
    }
}
```

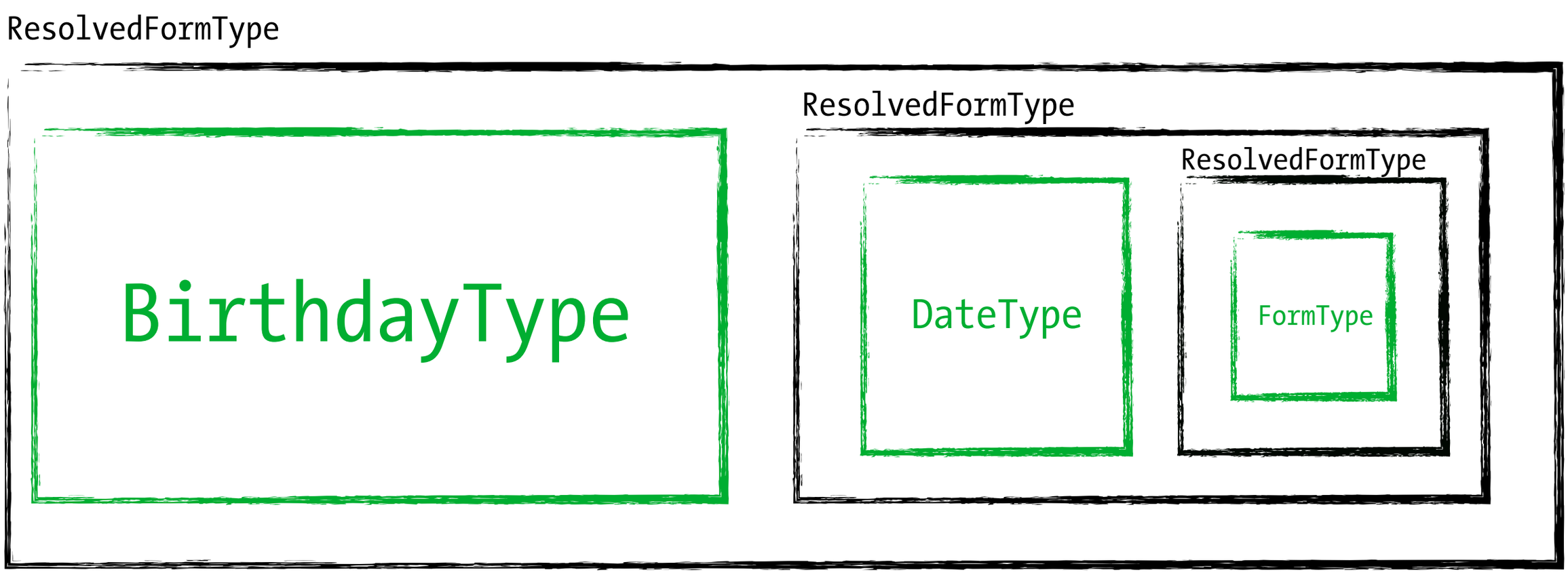
Vue d'ensemble



Composition d'un type résolu

```php
// birthday type
$type = new ResolvedFormType(
    new BirthdayType(), [], new ResolvedFormType(
        new DateType(), [], new ResolvedFormType(
            new FormType(), []
        )
    )
);
```

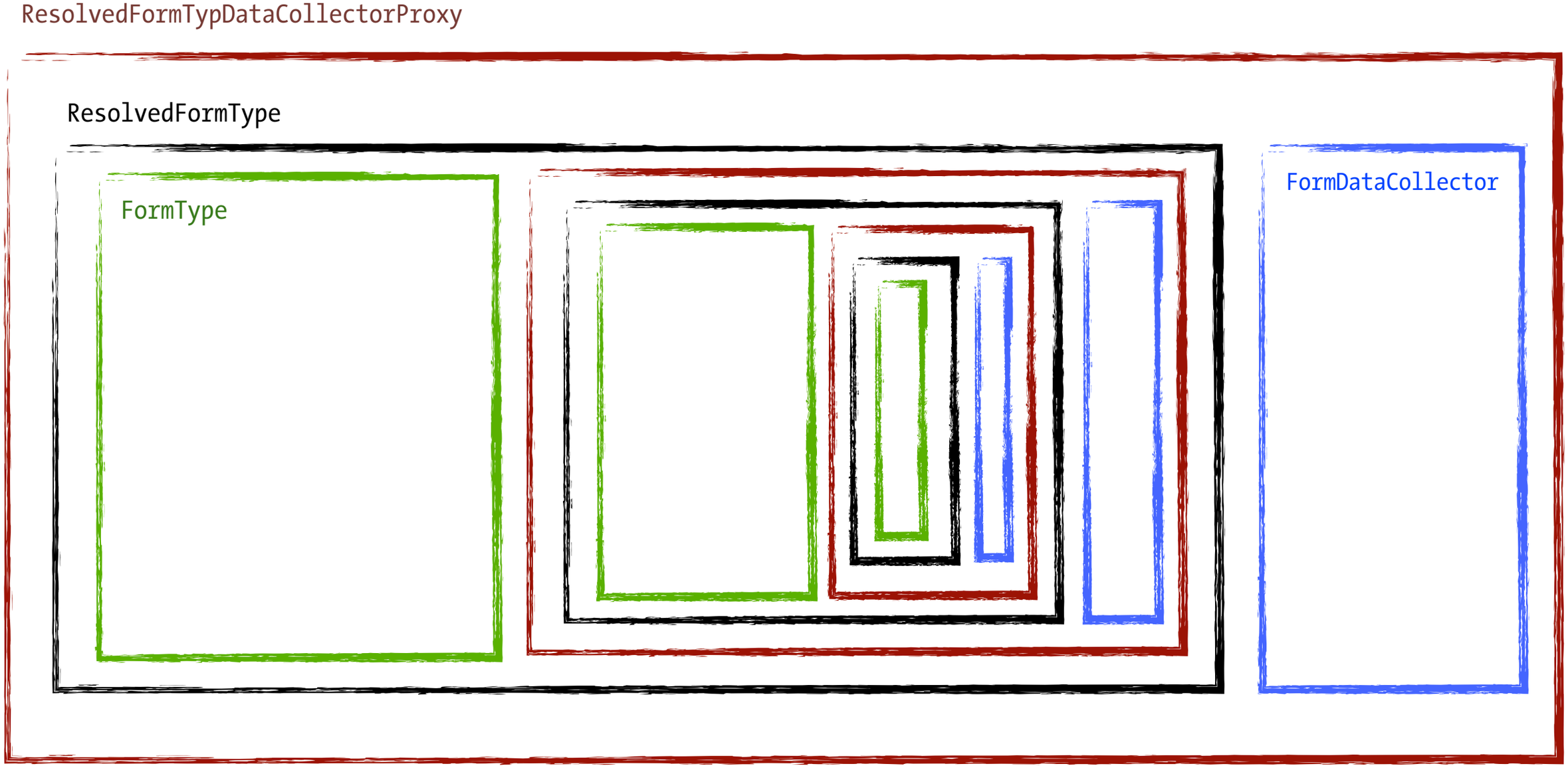
Composition d'un type résolu



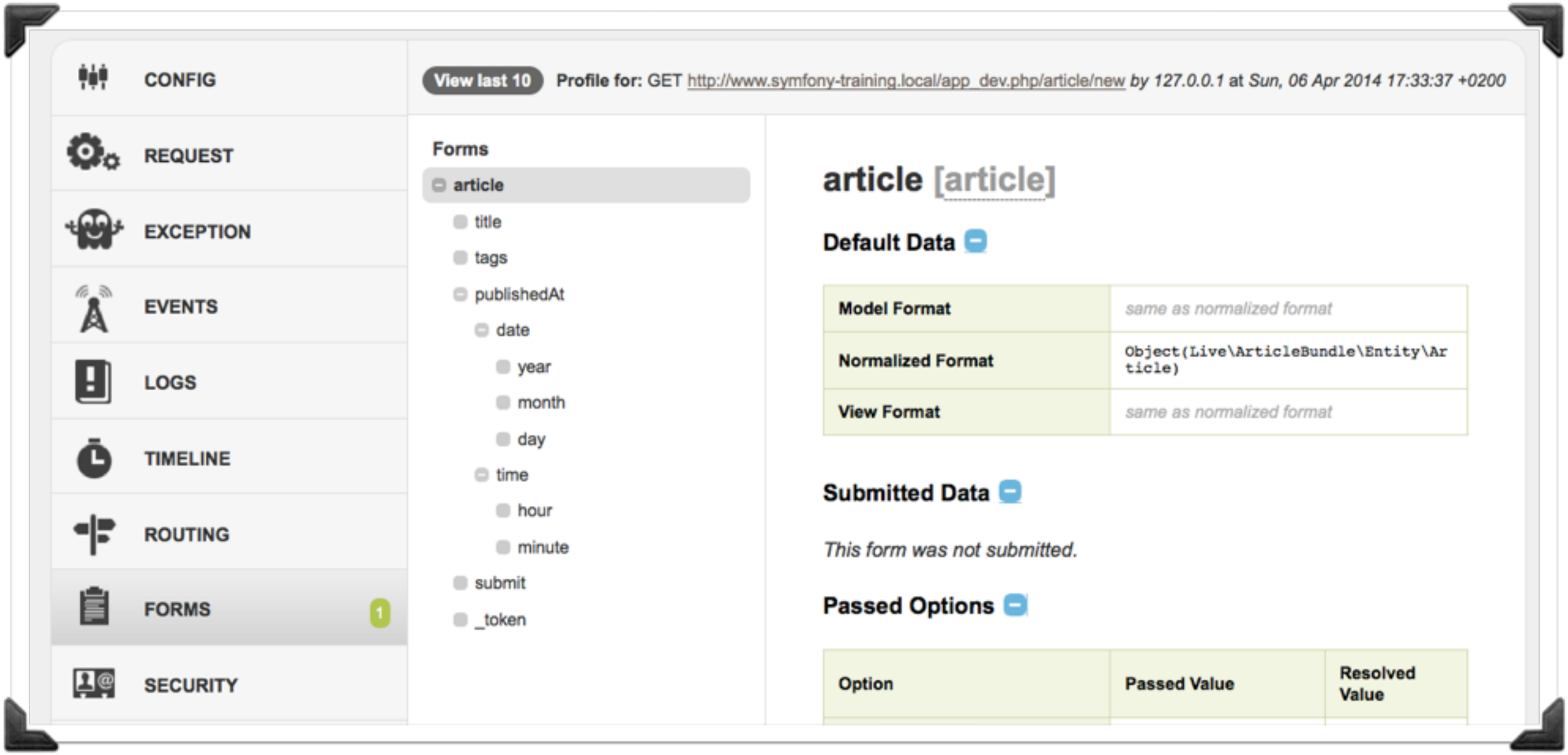
Collecter les infos de chaque type

```php
// text type
$collector = new FormDataCollector(…);

$type = new ResolvedTypeDataCollectorProxy(
    new ResolvedFormType(
        new TextType(),
        [],
        new ResolvedTypeDataCollectorProxy(
            new ResolvedFormType(new FormType(), []),
            $collector
        ),
    ),
    $collector
);
```

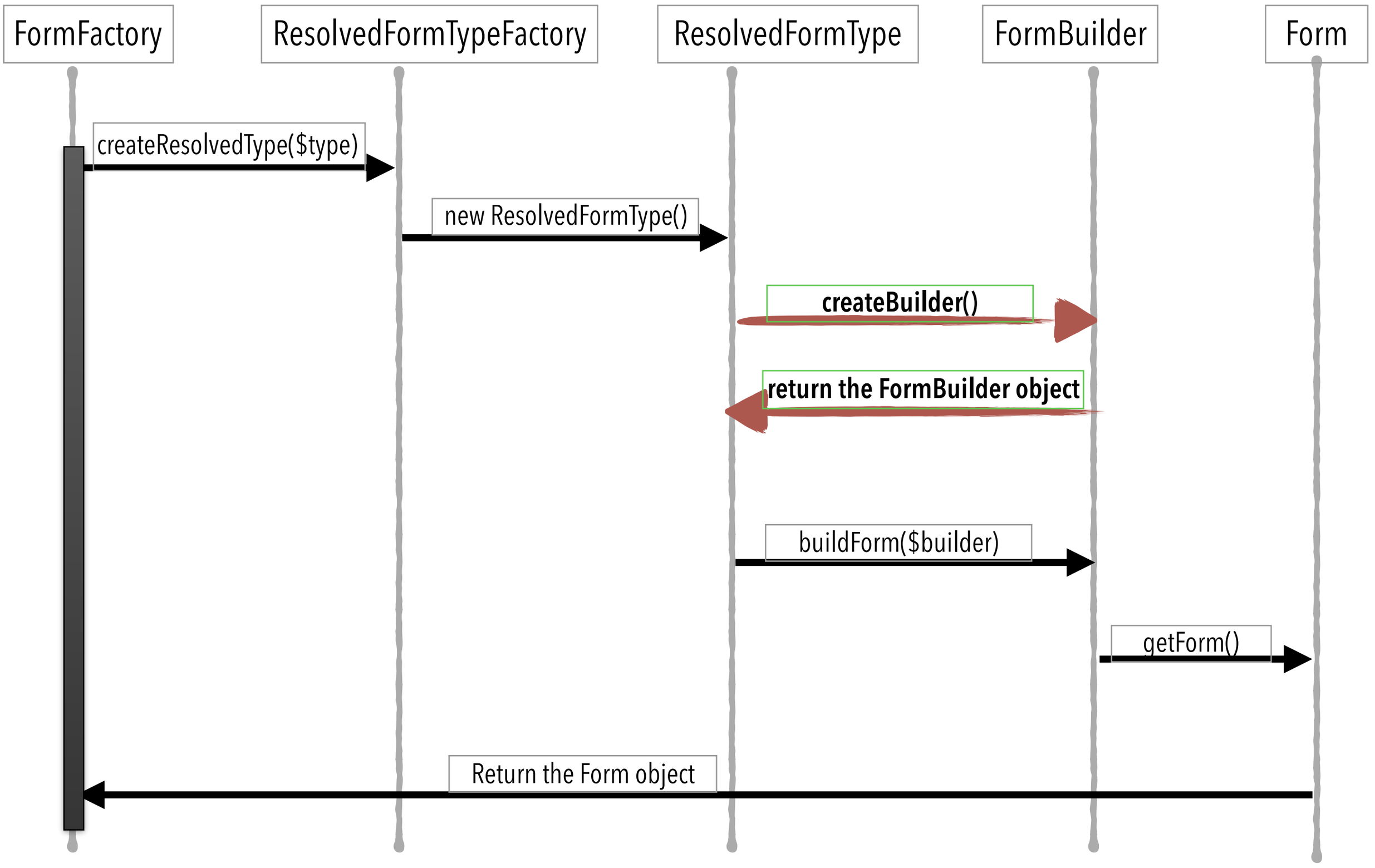
Composition d'un type résolu proxy



Débogage des formulaires



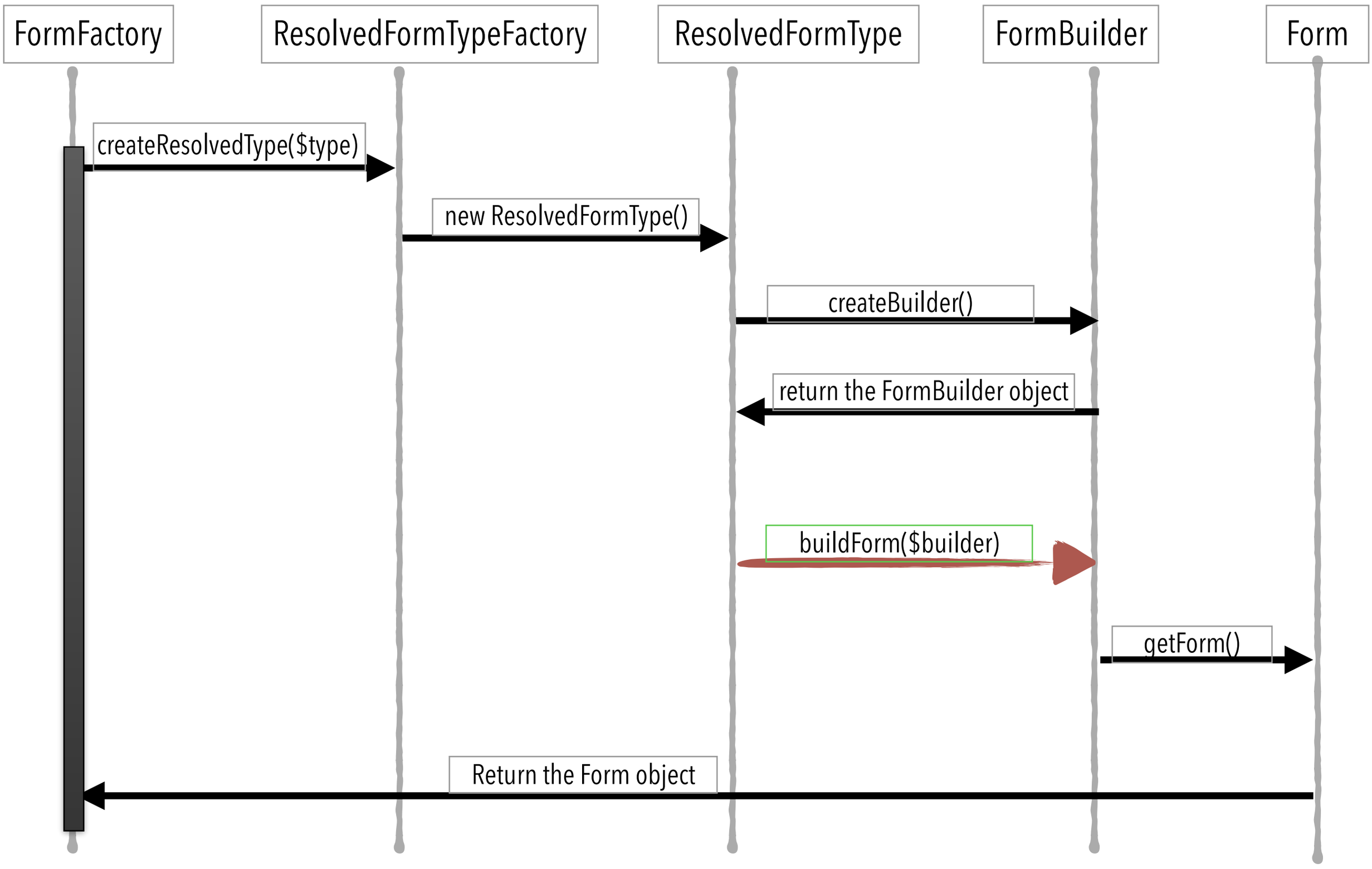
Vue d'ensemble



Création du FormBuilder

- **Construit** et **configure** l'instance de formulaire
- **Résout** toutes les options avec **l'OptionsResolver**
- **Enregistre** les écouteurs et les convertisseurs de données
- 1 ResolvedFormType → 1 FormBuilder → 1 EventDispatcher

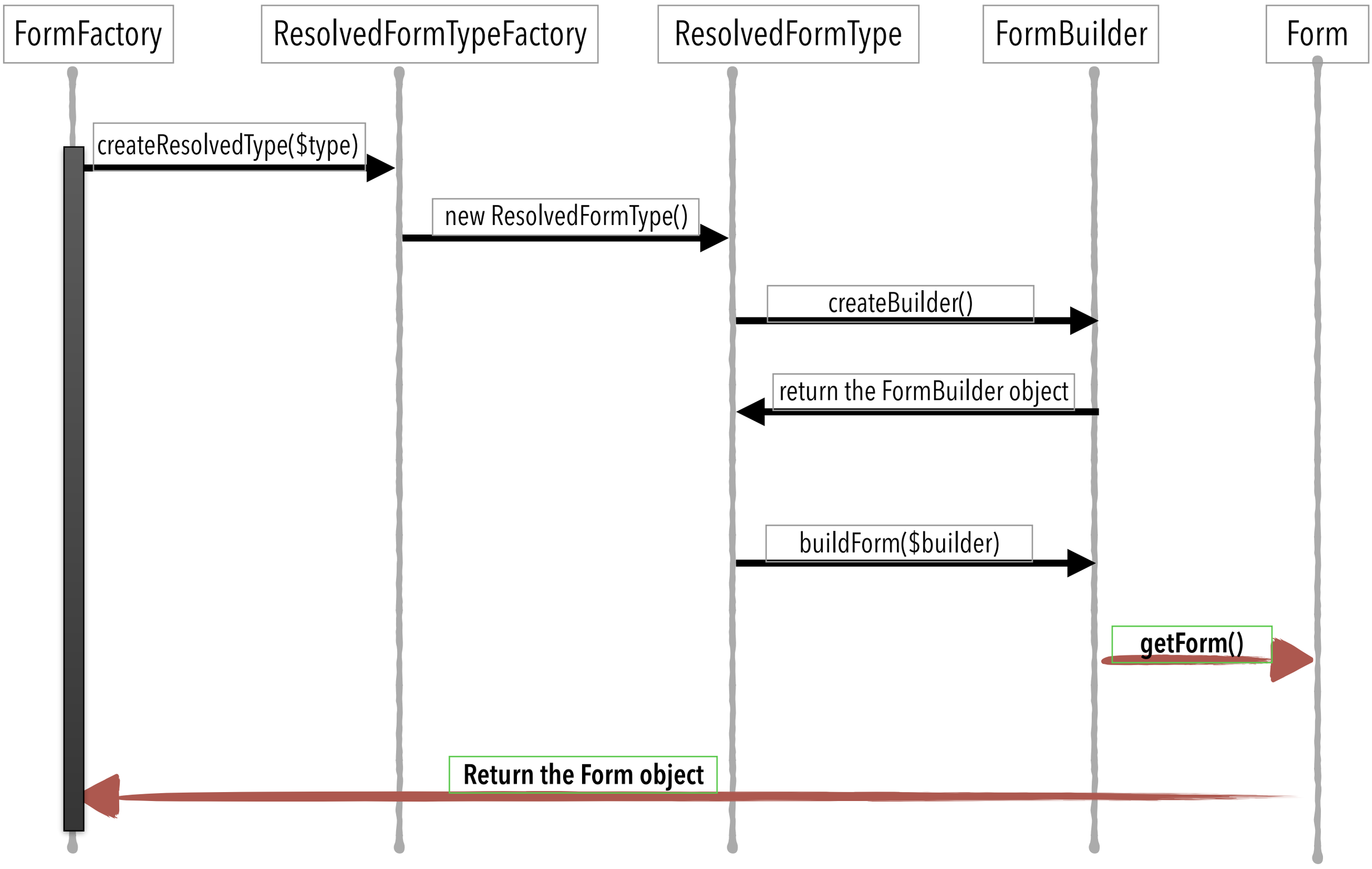
Vue d'ensemble



Préparation du FormBuilder

```php
namespace AppBundle\Form;

use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;

class ArticleType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('title')
            ->add('tags')
            ->add('publishedAt', 'datetime')
        ;
    }

    // ...
}
```

Vue d'ensemble



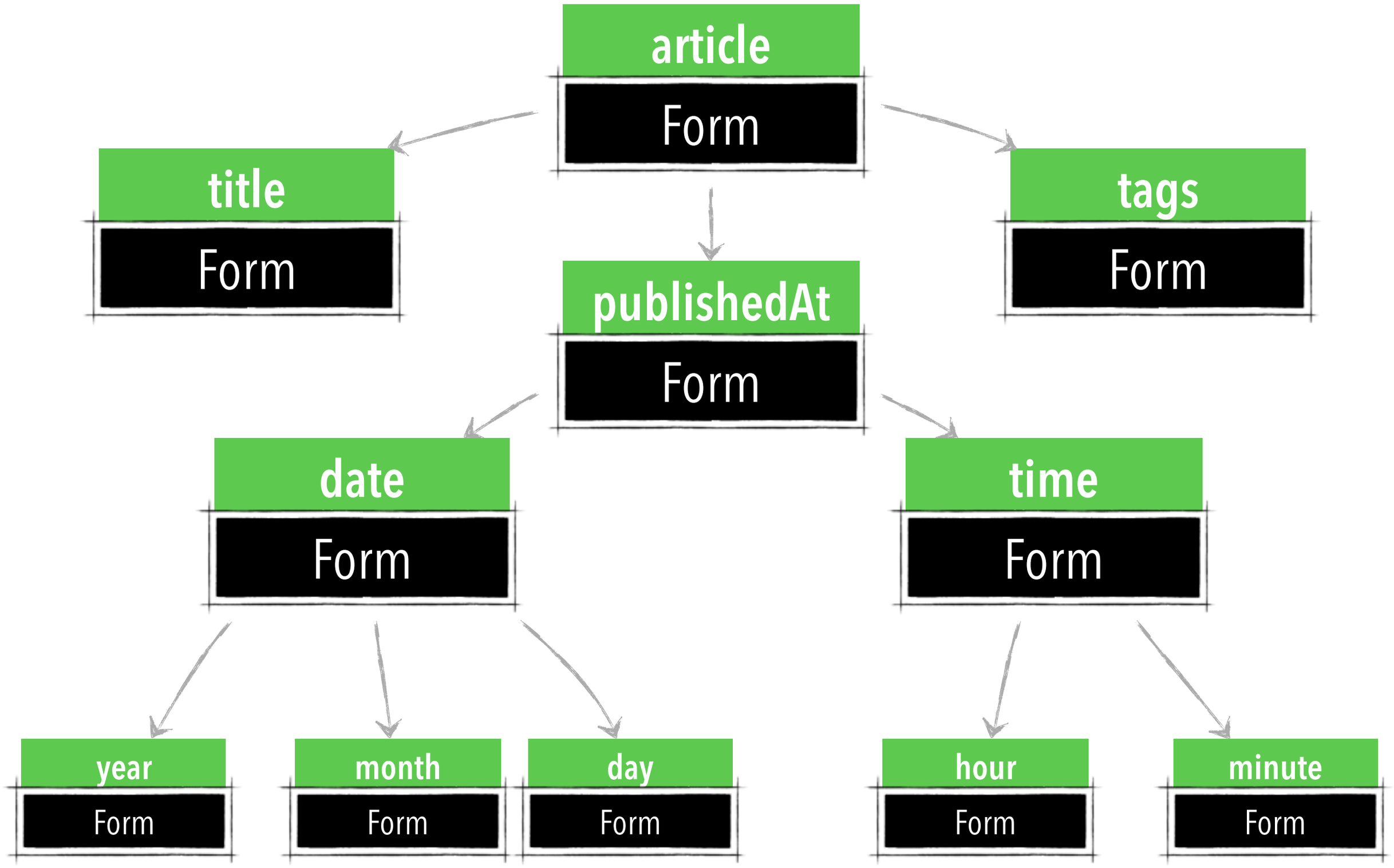
Dans la méthode FormBuilder::getForm()

```php
// Création du formulaire
$form = new Form($formConfig);
foreach ($this->getChildren() as $child) {
    $form->add($child);
}

// Initialisation des valeurs par défaut
$form->setData($article);
```

Création du formulaire

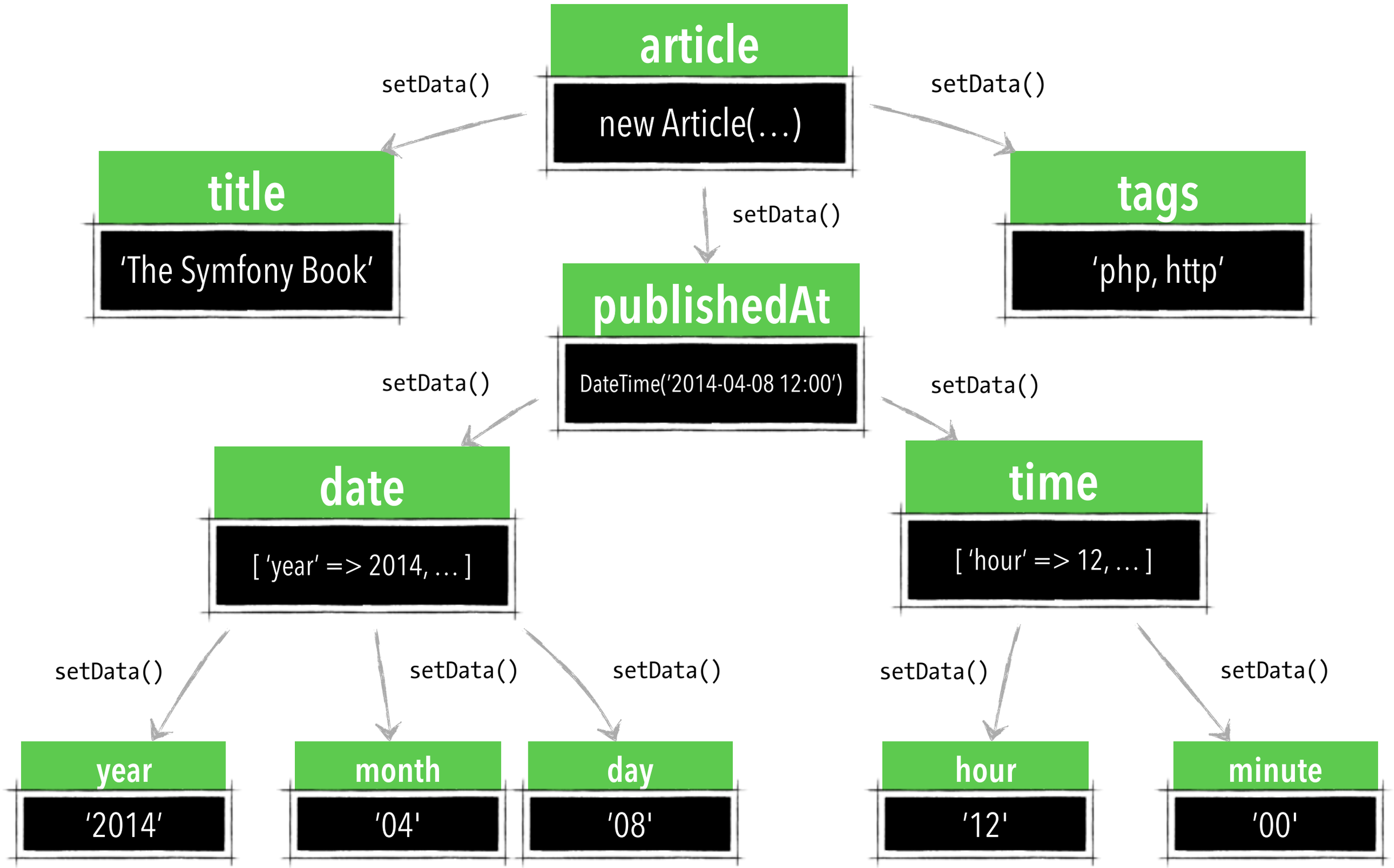
```php
// Création du formulaire
$form = new Form($formConfig);
foreach ($this->getChildren() as $child) {
    $form->add($child);
}

// Initialisation des valeurs par défaut
$form->setData($article);
```

Et voilà ! Le formulaire

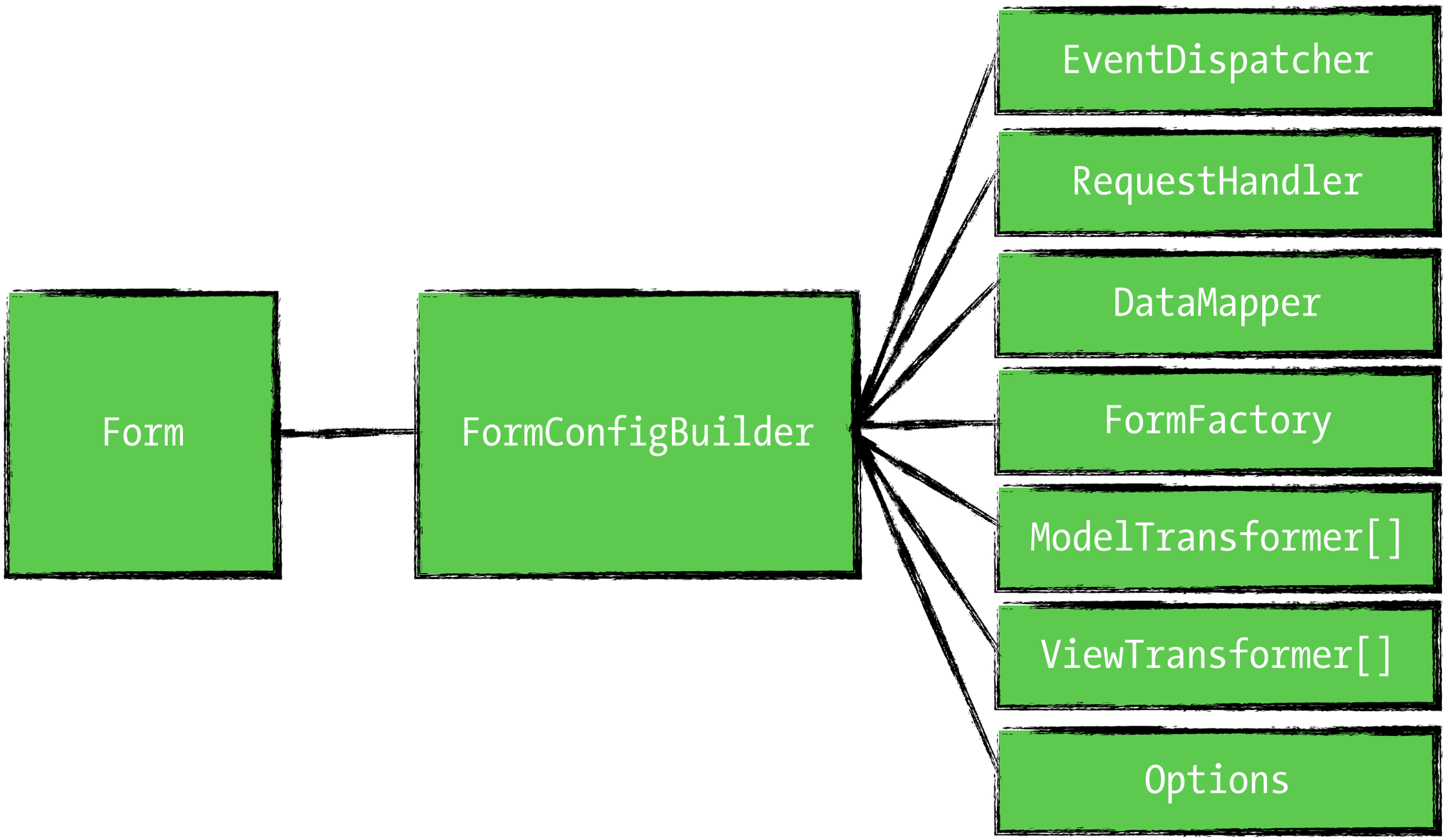


Initialisation des valeurs par défaut

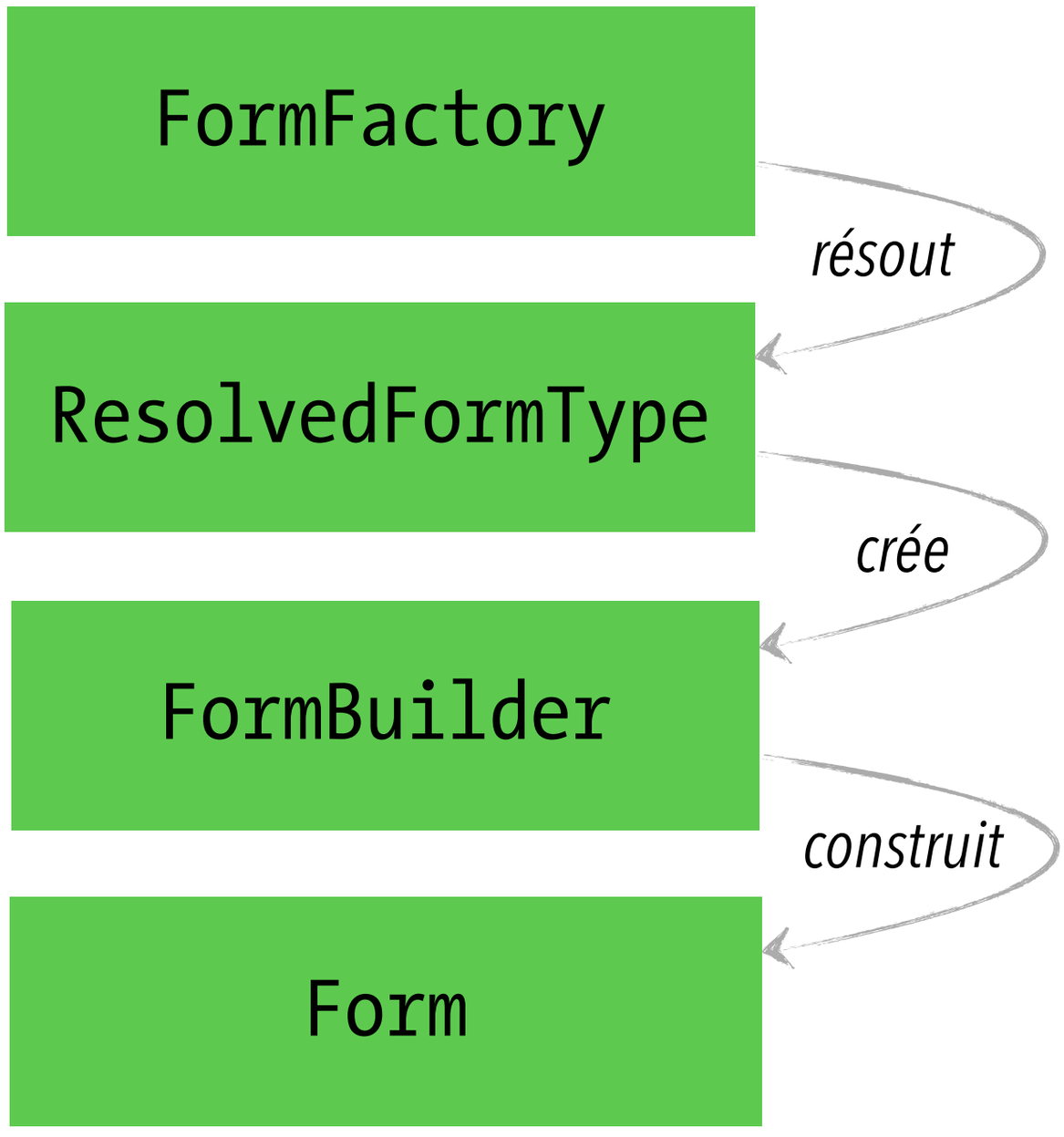
```
$form->setData($article);
```


article
new Article(...)
setData()
setData()
setData()
title
'The Symfony Book'
tags
'php, http'
publishedAt
DateTime('2014-04-08 12:00')
setData()
setData()
date
['year' => 2014, ...]
time
['hour' => 12, ...]
setData()
setData()
setData()
setData()
setData()
year
'2014'
month
'04'
day
'08'
hour
'12'
minute
'00'

Composition d'un formulaire



Récapitulons



Fabrique des instances de **formulaires** à partir de **types** de formulaires

Type résolu contenant :

- ✓ les **types de formulaire assemblés**
- ✓ options résolues.

Configure un seul formulaire

Instance du **formulaire** chargée de **traiter les données** qui lui sont fournies

https://www.flickr.com/photos/infofestival

Normalisation des données

SensioLabs

3 représentations des données

Model Data

Les **données** appartenant au **modèle** de données

NormData

Les **données normalisées** permettent le **passage** d'une **représentation à une autre**.

View Data

Les **données** utilisables pour **l'affichage** des informations dans le navigateur.

2 règles d'Or

Les **données normalisées** doivent contenir le **maximum d'informations.**

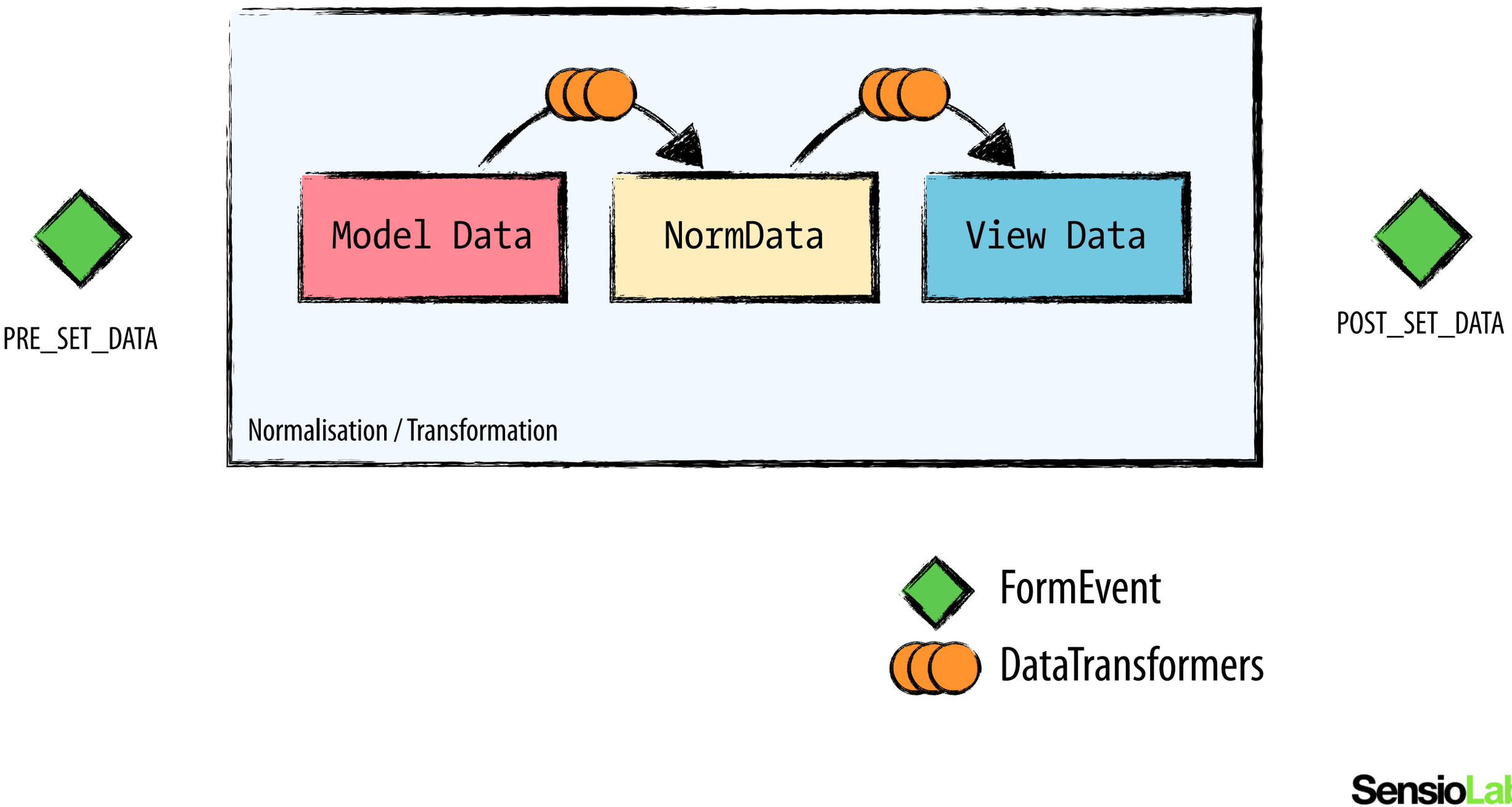
Ne pas changer le contenu de la donnée mais **seulement sa représentation**

Changer la représentation

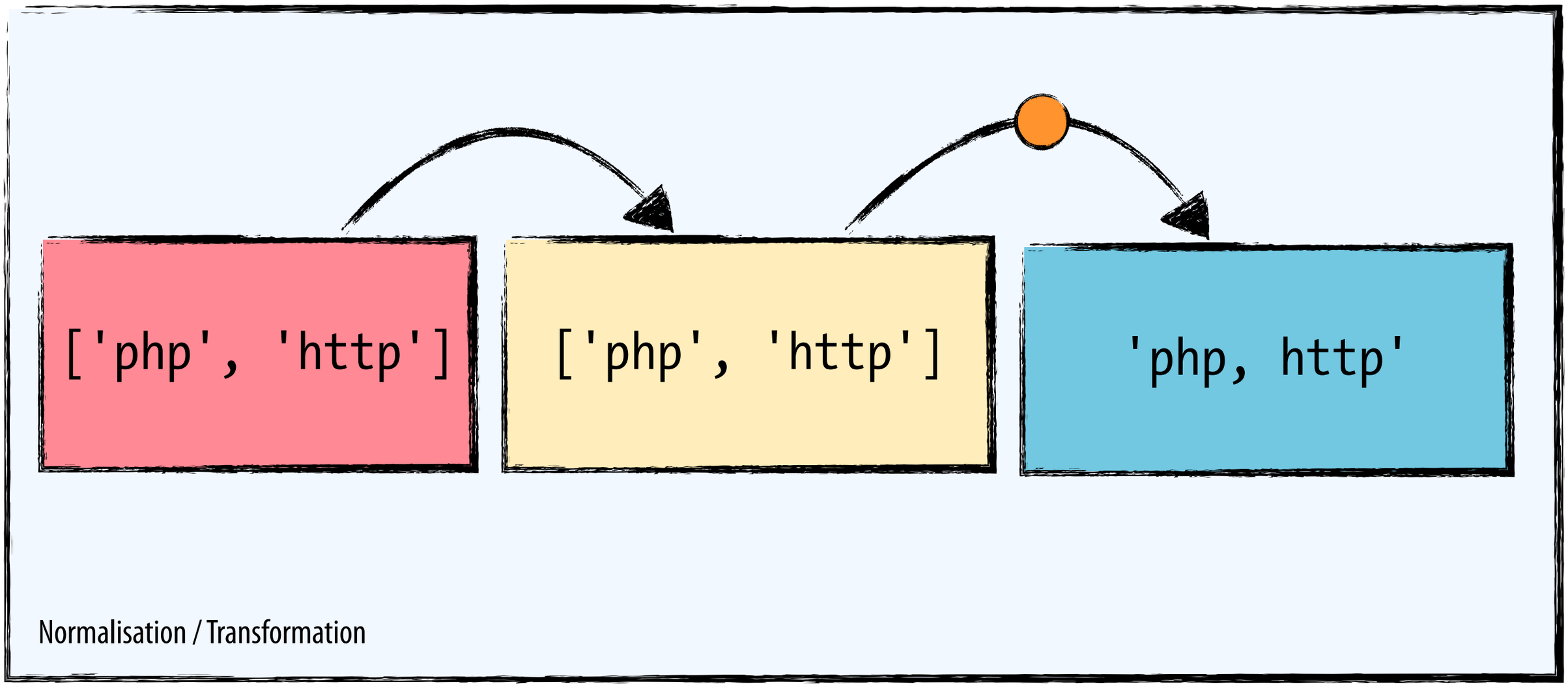
~~'2014-04-08' => '2015-09-10'~~

```
'2014-04-08' => DateTime('2014-04-08')
```

Définition des valeurs par défaut du formulaire



Normaliser les tags de l'article



DataTransformers

Les convertisseurs de données

Les **convertisseurs** de données aka « DataTransformer » **modifient** la **représentation** des données.

Créer un convertisseur de données

```php
namespace Live\ArticleBundle\Form\DataTransformer;

use Symfony\Component\Form\DataTransformerInterface;
use Symfony\Component\Form\Exception\TransformationFailedException;

class ArrayToDelimitedStringTransformer implements DataTransformerInterface
{
    public function transform($value)
    {
        if (!is_array($value)) {
            throw new TransformationFailedException();
        }

        if (0 === count($value)) {
            return '';
        }

        return implode(', ', $value);
    }
}
```

Enregistrer un convertisseur de données

```
$builder->addViewTransformer(
        new ArrayToDelimitedStringTransformer()
);
```

```
$builder->addModelTransformer(
        new My\Custom\ModelDataTransformer()
);
```

Quelques convertisseurs natifs

```
DataTransformer
├── ArrayToPartsTransformer.php
├── BaseDateTimeTransformer.php
├── BooleanToStringTransformer.php
├── ChoiceToBooleanArrayTransformer.php
├── ChoiceToValueTransformer.php
├── ChoicesToBooleanArrayTransformer.php
├── ChoicesToValuesTransformer.php
├── DataTransformerChain.php
├── DateTimeToArrayTransformer.php
├── DateTimeToLocalizedStringTransformer.php
├── DateTimeToRfc3339Transformer.php
├── DateTimeToStringTransformer.php
├── DateTimeToTimestampTransformer.php
├── IntegerToLocalizedStringTransformer.php
├── MoneyToLocalizedStringTransformer.php
├── NumberToLocalizedStringTransformer.php
├── PercentToLocalizedStringTransformer.php
└── ValueToDuplicatesTransformer.php
```

https://www.flickr.com/photos/safaguezguez/4782575498/sizes/l/in/photostream/

Soumission des données

Vous vous souvenez ?

```php
class ArticleController extends Controller
{
    /** @Template */
    public function newAction(Request $request)
    {
        $article = new Article('...');

        $form = $this->createForm(new ArticleType(), $article);

        $form->handleRequest($request);

        // ...
    }
}
```

C'est reparti !

```php
class ArticleController extends Controller
{
    /** @Template */
    public function newAction(Request $request)
    {
        $article = new Article('...');

        $form = $this->createForm(new ArticleType(), $article);

        $form->handleRequest($request);

        // ...
    }
}
```

Soumission du formulaire

```
$form->submit($request->request->get('article'));
```

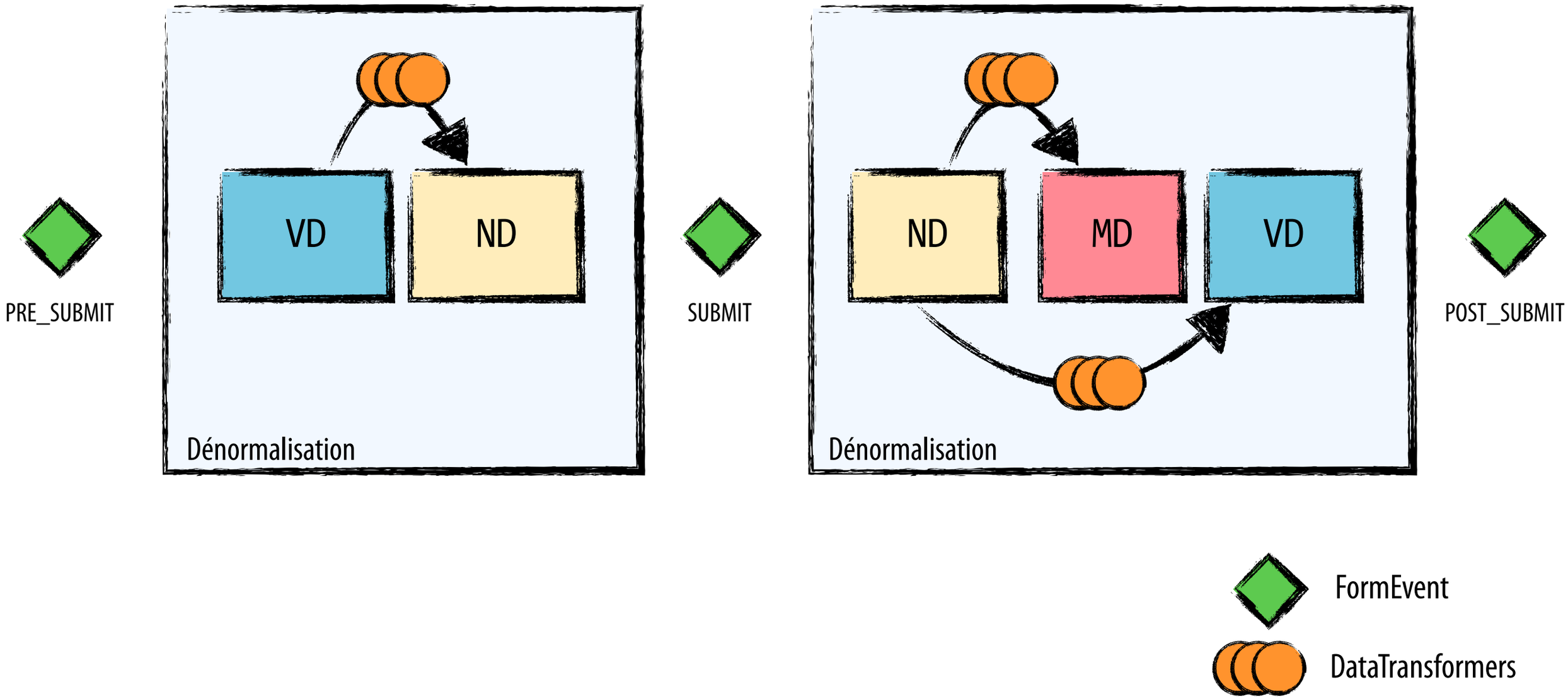
ou

```
$form->handleRequest($request);
```

HttpFoundationRequestHandler

- V**érifie** que le **formulaire** est **soumis** avec le bon type de **requête**
- Récupère les données depuis la requête
- *$request->**query*** | *$request->**request*** | *$request->**files***
- Appelle la méthode ***Form::submit()***

Réintégration des données au modèle



Form::submit() est appelé sur tous les **objets** Form **fils**

Dénormaliser une chaîne de tags

```php
class ArrayToDelimitedStringTransformer implements DataTransformerInterface
{
    public function reverseTransform($value)
     {
         if (!is_string($value)) {
             throw new TransformationFailedException();
         }

         if (empty($value)) {
             return array();
         }

         return explode(', ', $value);
     }
}
```

Les événements

Les **événements** du formulaire permettent de **modifier** le **contenu de la donnée** ou bien de modifier la **structure du formulaire**.

Utilisation des événements

Constante	Nom de l'événement	Objet de l'événement
PRE_SET_DATA	form.pre_set_data	FormEvent($form, $modelData)
POST_SET_DATA	form.post_set_data	FormEvent($form, $modelData)
PRE_SUBMIT	form.pre_bind	FormEvent($form, $viewData)
SUBMIT	form.bind	FormEvent($form, $normData)
POST_SUBMIT	form.post_bind	FormEvent($form, $viewData)

Quels usages ?

- Ajouter / Supprimer des champs dans une collection
- Ajouter des champs en fonction de l'utilisateur
- Modifier / Filter le contenu donné

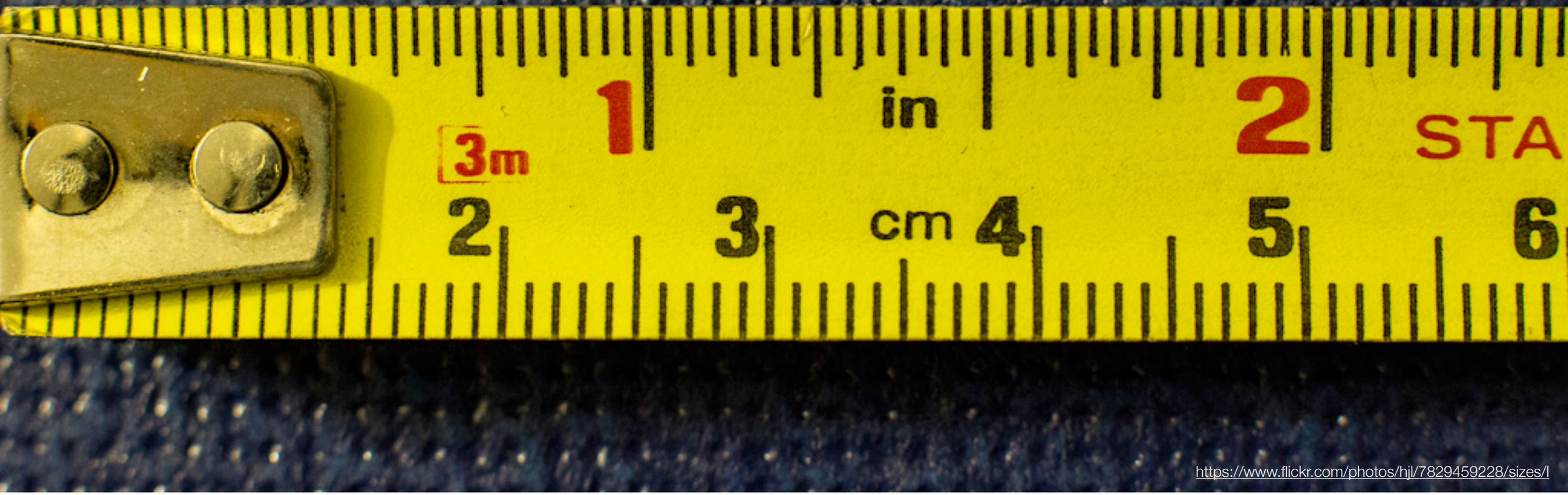
- **Exemple** : *changer la casse d'une chaîne de caractères*
- **Exemple** : *alimenter des listes déroulantes liées*

Mettre le titre de l'article en « title case »

```php
// ArticleType.php
use Symfony\Component\Form\FormEvent;
use Symfony\Component\Form\FormEvents;

// Dans la méthode ArticleType::buildForm()
$titleBuilder = $builder->create('title');
$titleBuilder->addEventListener(
    FormEvents::PRE_SUBMIT, function (FormEvent $event) {
        $title = ucwords($event->getData());
        $event->setData($title);
    }
);

$builder->add($titleBuilder);
```

Enregistrer un « event subscriber »

```
$builder->add(
    $builder
        ->create('title')
        ->addEventSubscriber(
          new StringUtilEventSubscriber()
        )
);
```



https://www.flickr.com/photos/hjl/7829459228/sizes/l

Extension de Type de Formulaire

SensioLabs

Les extension de types

Une **extension de type de formulaire** est une structure qui permet **d'étendre** et **configurer** un **groupe de types** de formulaire.

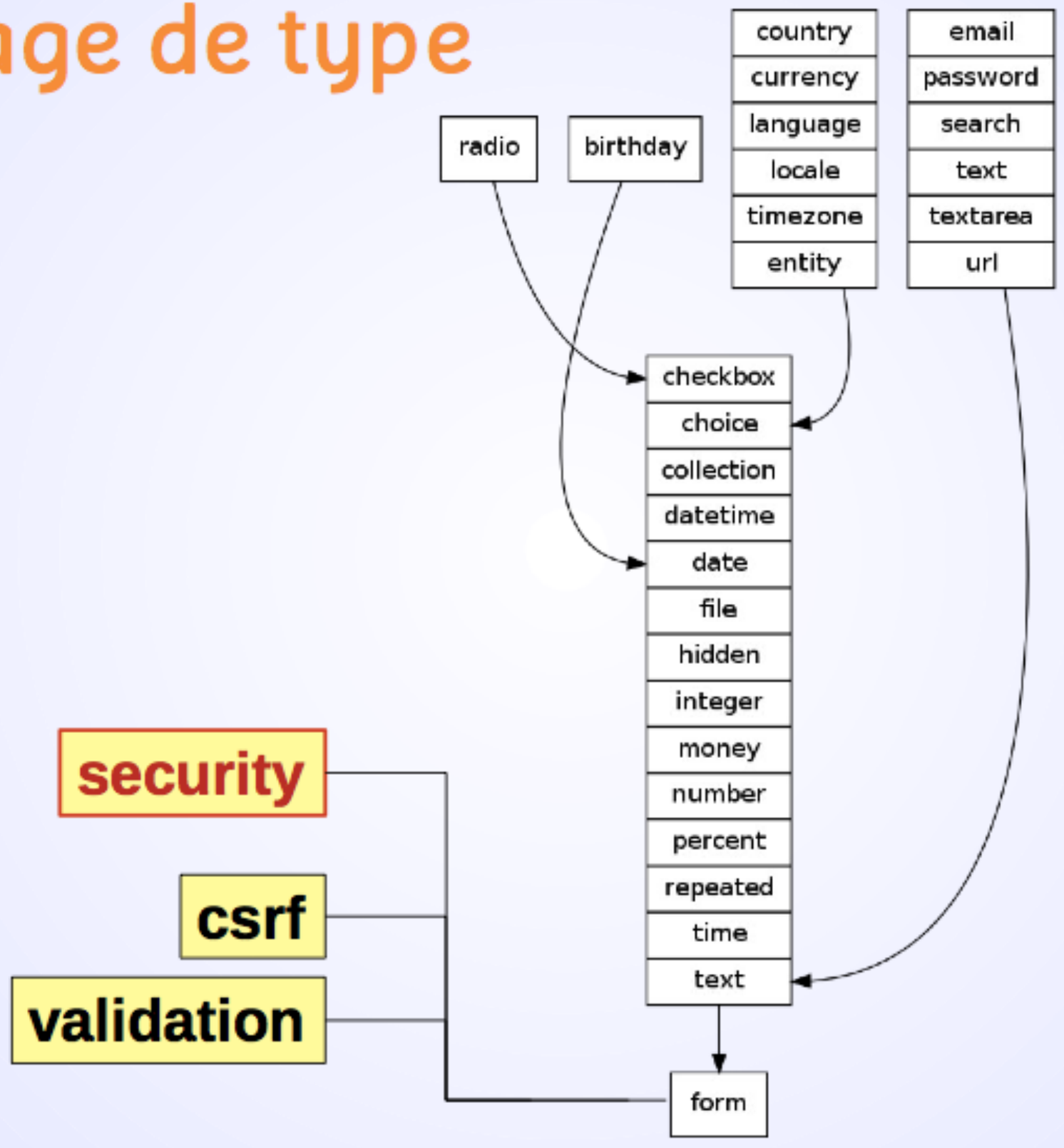
Dans quels buts?

- Enregistrer de nouvelles options globales
- Passer des variables à la vue
- Enregistrer de nouveaux types de formulaire
- Enregistrer de nouveaux convertisseurs de données
- Enregistrer de nouveaux écouteurs d'événements

Exemples

- FormTypeCsrfExtension
- ValidatorTypeExtension

Héritage de type



Créer une extension de type de formulaire

```php
namespace AppBundle\Form\Extension;

use Symfony\Component\Form\AbstractTypeExtension;
use Symfony\Component\Security\Core\Authorization\AuthorizationCheckerInterface;

class AuthorizationTypeExtension extends AbstractTypeExtension
{
    private $authorizer;

    public function __construct(AuthorizationCheckerInterface $authorizer)
    {
        $this->authorizer = $authorizer;
    }

    public function getExtendedType()
    {
        return 'form';
    }
}
```

Enregistrer de nouvelles options

```php
namespace AppBundle\Form\Extension;

// ...
use Symfony\Component\OptionsResolver\OptionsResolverInterface;

class AuthorizationTypeExtension extends AbstractTypeExtension
{
    // ...

    public function setDefaultOptions(OptionsResolverInterface $resolver)
    {
        $resolver->setDefaults([
            'role' => null,
        ]);
    }
}
```

Enregister un écouteur d'événement

```php
// ...
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\Form\FormEvent;
use Symfony\Component\Form\FormEvents;

class AuthorizationTypeExtension extends AbstractTypeExtension
{
    // ...
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        // If no permissions requirement or user is authorized, skip this event listener.
        if (null === $options['role'] || $this->authorizer->isGranted($options['role'])) {
            return;
        }

        // Otherwise, remove the form element from its parent form.
        $builder->addEventListener(FormEvents::PRE_SET_DATA, function (FormEvent $event) {
            $form = $event->getForm();
            if (!$form->isRoot()) {
                $form->getParent()->remove($form->getName());
            }
        });
    }
}
```

Enregister la nouvelle extension

```yaml
# app/config/services.yml
form.extension.authorization:
    class: AppBundle\Form\Extension\AuthorizationTypeExtension
    tags:
        - { name: form.type_extension, alias: form }
```

Utiliser la nouvelle option

```
class ArticleType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, …)
    {
        $builder
            ->add('title')
            ->add('tags')
            ->add('publishedAt', 'datetime')
            ->add('slug', 'text', [ 'role' => 'ROLE_ADMIN' ])
        ;
    }
    // ...
}
```

https://www.flickr.com/photos/nesto

Fin

training@sensiolabs.com

SensioLabs