

(S) OFFICIAL

PROOF OF SERVICE

Served as described below, with any of the following methods of service. Plaintiff cannot serve the defendant at home.

delivery of damages Other (specify): _____

(1) date: _____ (2) time: _____ (3) address: _____

by mailing (1) date: _____ (2) place: _____

Manner of service (check proper box):

a. Personal service. By personally delivering, during usual office hours, copies were left. (CCP § 415.20)

b. Substituted service on corporation, usual place of abode, or usual household of the general nature of the acts relied on, informed served at the place where the acts relied on, acknowledged with signature.

PROOF OF SERVICE Plaintiff cannot serve the defendant at home.

other (name and title of plaintiff): _____ at business _____

STATEMENT OF PERSONAL INJURY OR DEATH DAMAGE

To (name of one defendant only): Plaintiff (name of one plaintiff only): seeks damages in the above-entitled action, as follows:

1. General damages
 - a. Pain, suffering, and inconvenience
 - b. Emotional distress
 - c. Loss of consortium
 - d. Loss of society and companionship (wrongful death actions only)
 - e. Other (specify) _____
 - f. Other (specify) _____
 - g. Continued on Attachment 1.g.
2. Special damages
 - a. Medical expenses (to date) _____
 - b. Future medical expenses (present value) _____
 - c. Loss of earning capacity (present value) _____
 - d. Loss of future earning capacity (actions only) _____
 - e. Property damage _____
 - f. Funeral expenses (wrongful death actions only) _____
 - g. Future contributions (present value) (wrongful death actions only) _____
 - h. Value of personal service, advice, or training (wrongful death actions only) _____
 - i. Other (specify) _____
 - j. Other (specify) _____
 - k. Continued on Attachment 2.k.

Plaintiff reserves the right to seek punitive damages in the suit filed against you.

In the amount of (specify). \$ _____

SIGNATURE OF PLAINTIFF OR ATTORNEY

Code: _____

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

Ingénierie inversée du composant Form

Sarah Khalil
Hugo Hamon



SymfonyLive
PARIS 2014
7-8 AVRIL

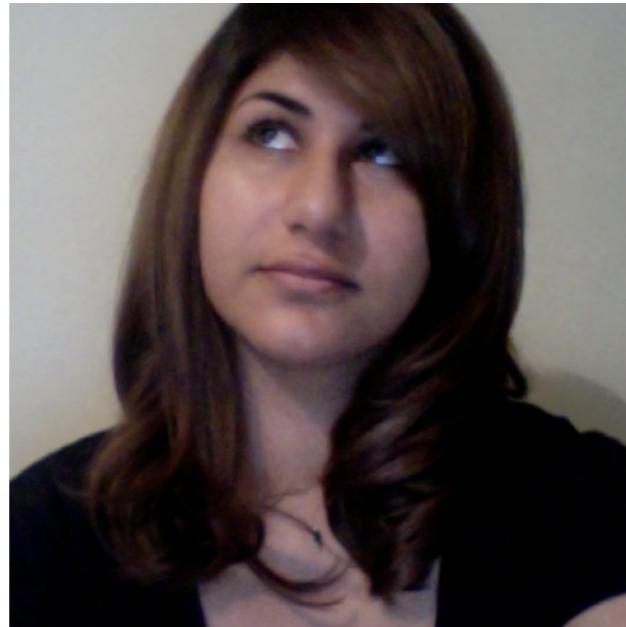
SensioLabs

Vos humbles serviteurs

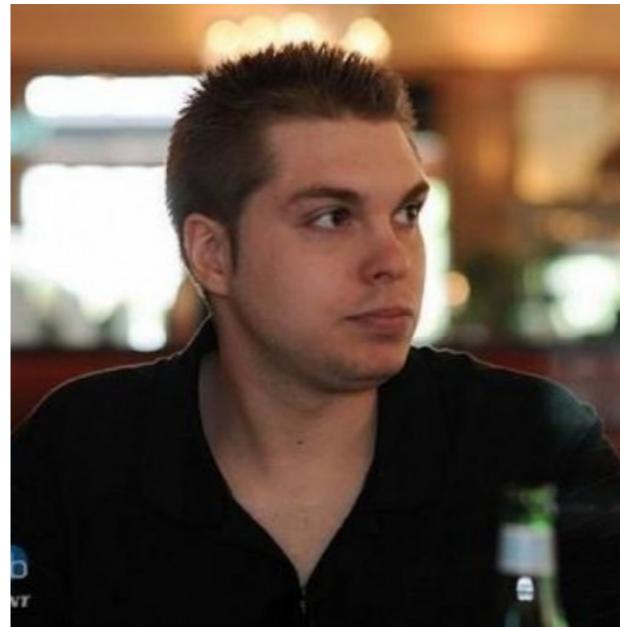
Sarah Khalil

Cultivatrice de projets

@ **SensioLabs**



@saro0h



@hhamon

Hugo Hamon

Responsable des formations

@ **SensioLabs**

De quoi allons-nous vous parler?

Au menu

I. Les formulaires au quotidien

II. Construction d'un formulaire

III. Traitement du formulaire

Quelques chiffres

Quelques chiffres

45 568/311 349 LOC

52 Dossiers

292 Fichiers

53 Namespaces

30 Interfaces

272 Classes

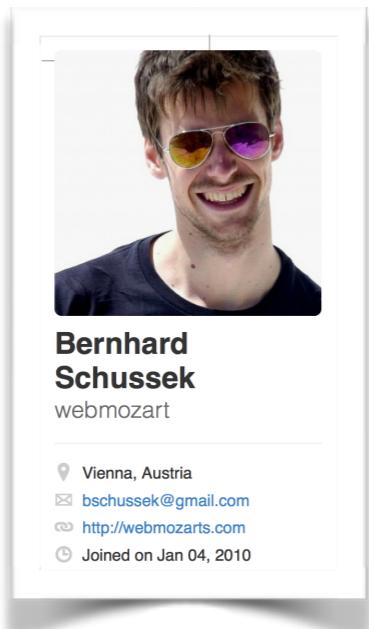
38 Constantes

2 312 Méthodes

Bridge	BrowserKit
Bundle	ClassLoader
Component	Config
	Console
	CssSelector
	Debug
	DependencyInjection
	DomCrawler
	EventDispatcher
	ExpressionLanguage
	Filesystem
	Finder
	Form
	HttpFoundation
	HttpKernel
	Intl
	Locale
	OptionsResolver
	Process
	PropertyAccess
	Routing
	Security
	Serializer
	Stopwatch
	Templating
	Translation
	Validator
	Yaml
	AbstractTypeExtension.php
	Button.php
	ButtonBuilder.php
	ButtonTypeInterface.php
	CallbackTransformer.php
	CHANGELOG.md
	ClickableInterface.php
	composer.json
	DataMapperInterface.php
	DataTransformerInterface.php
	Exception
	Extension
	Form.php
	FormBuilder.php
	FormBuilderInterface.php
	FormConfigBuilder.php
	FormConfigBuilderInterface.php
	FormConfigInterface.php
	FormError.php
	FormEvent.php
	FormEvents.php
	FormExtensionInterface.php
	FormFactory.php
	FormFactoryBuilder.php
	FormFactoryBuilderInterface.php
	FormFactoryInterface.php
	FormInterface.php
	FormRegistry.php
	FormRegistryInterface.php
	FormRenderer.php
	FormRendererEngineInterface.php
	FormRendererInterface.php
	Forms.php
	FormTypeExtensionInterface.php
	FormTypeGuesserChain.php
	FormTypeGuesserInterface.php
	FormTypeInterface.php
	FormView.php
	Guess
	LICENSE
	NativeRequestHandler.php
	phpunit.xml.dist
	PreloadedExtension.php
	README.md
	RequestHandlerInterface.php
	ResolvedFormType.php
	ResolvedFormTypeFactory.php
	ResolvedFormTypeFactoryInterface.php
	ResolvedFormTypeInterface.php
	Resources
	ReversedTransformer.php
	SubmitButton.php

Sur Github

- Récupérer le composant :
- via Composer
- <https://github.com/symfony/form>
- +200 issues ouvertes



Les dépendances

Obligatoires

Event Dispatcher
OptionsResolver
Property Access
Intl

Facultatives

HttpKernel
HttpFoundation
Validator
Dependency Injection
Twig Bridge
Security (CSRF)



Problématiques

On nous demande ceci

new article

Title	<input type="text" value="The Symfony Book"/>
Tags	<input type="text" value="php, http"/>
Published at	<input type="text" value="8 avr. 2014 12:00"/>
<input type="button" value="Submit"/>	

On le traduit comme ça

Objet métier

(domain object pour les intimes)

```
namespace Live\ArticleBundle\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)

```
namespace Live\ArticleBundle\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)

```
namespace Live\ArticleBundle\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
        }

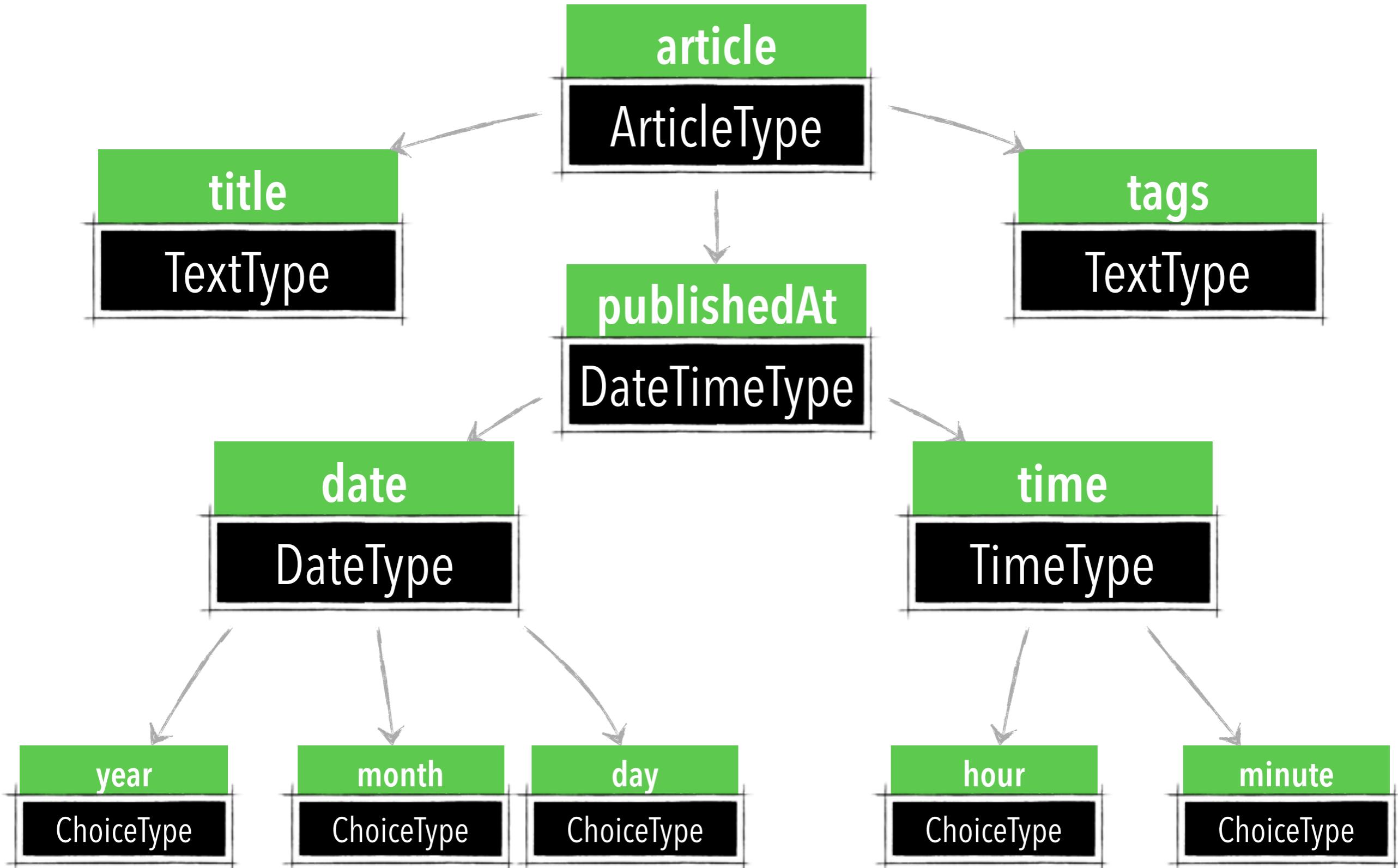
        return [ 'form' => $form->createView() ];
    }
}
```

Du point de vue de l'humain

new article

Title	<input type="text" value="The Symfony Book"/>
Tags	<input type="text" value="php, http"/>
Published at	<input type="text" value="8 avr. 2014 12:00"/>
<input type="button" value="Submit"/>	

Du point de vue de Symfony



Ce qu'on pense communément du

Composant Form



Keyvan Akbary

@keyvanakbary

Follow

If you need a bachelor thesis to explain how the
#Symfony #Form component works *_maybe_* is
because is tooo complex
webmozarts.com/bsc-thesis.pdf

6:46 PM - 27 Jan 2014

1 RETWEET 2 FAVORITES





Bernhard Schussek

@webmozart

 Follow

[@keyvanakbary](#) maybe it is also because form processing in general is pretty complex? ;)

12:19 AM - 30 Jan 2014





Nicolò Pignatelli

@nicolopigna

Follow

**#Symfony #form component really is like a woman:
very complex, but learn to deal with it and a lot of
love will be returned!**

5:01 PM - 14 Sep 2012

4 RETWEETS



SymfonyLive
PARIS 2014
7-8 AVRIL

SensioLabs



Construction d'un formulaire

La notion de type

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

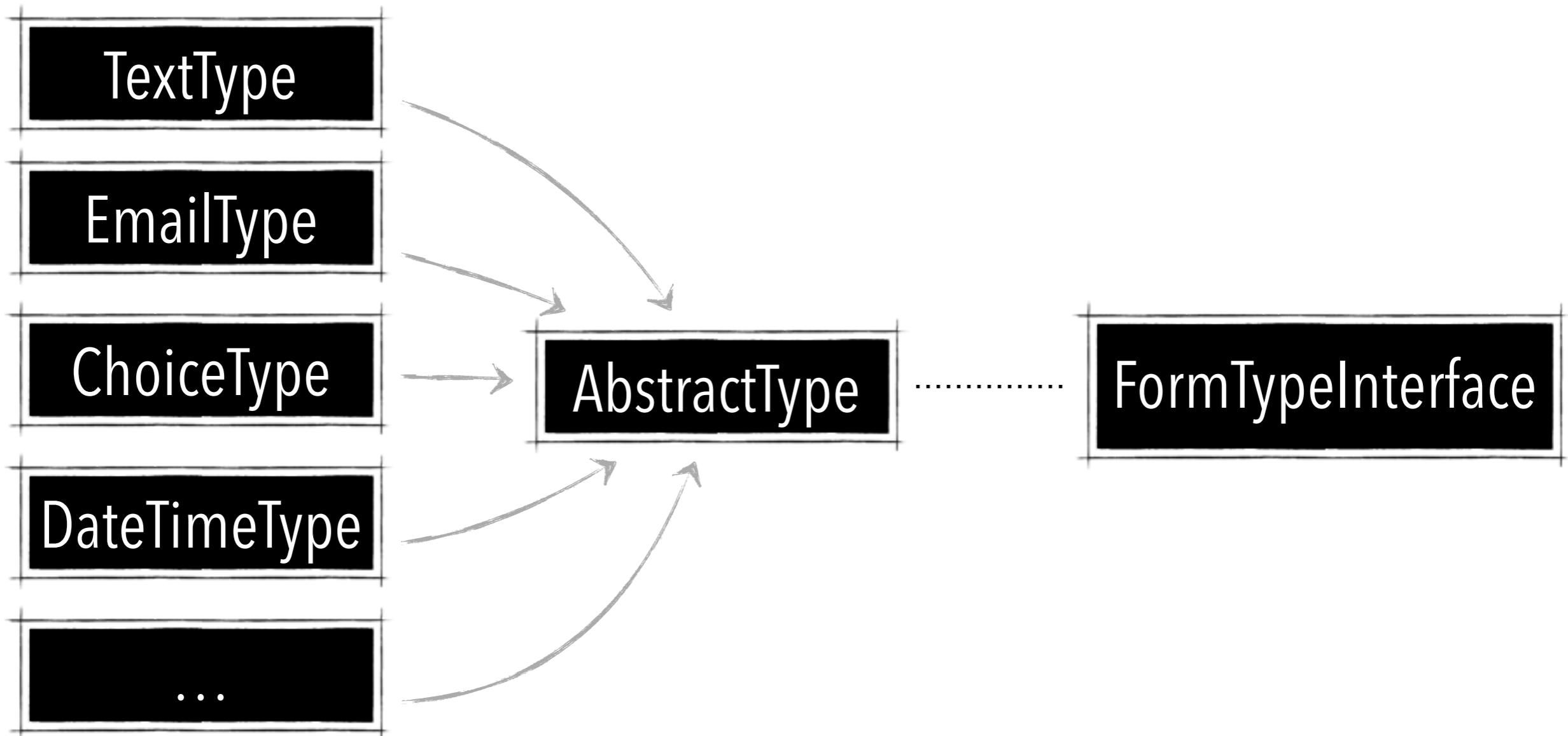
```
$ php app/console container:debug | grep form.type
```

```
form.type.birthday  
form.type.button  
form.type.checkbox  
form.type.choice  
form.type.collection  
form.type.country  
form.type.currency  
form.type.date  
form.type.datetime  
form.type.email  
form.type.entity  
form.type.file  
form.type.form  
form.type.hidden  
form.type.integer  
form.type.language  
form.type.locale  
form.type.money  
form.type.number  
form.type.password  
form.type.percent  
form.type.radio  
form.type.repeated  
form.type.reset  
form.type.search  
form.type.submit  
form.type.text  
form.type.textarea  
form.type.time  
form.type.timezone  
form.type.url
```

```
container Symfony\Component\Form\Extension\Core\Type\BirthdayType  
container Symfony\Component\Form\Extension\Core\Type\ButtonType  
container Symfony\Component\Form\Extension\Core\Type\CheckboxType  
container Symfony\Component\Form\Extension\Core\Type\ChoiceType  
container Symfony\Component\Form\Extension\Core\Type\CollectionType  
container Symfony\Component\Form\Extension\Core\Type\CountryType  
container Symfony\Component\Form\Extension\Core\Type\CurrencyType  
container Symfony\Component\Form\Extension\Core\Type\DateType  
container Symfony\Component\Form\Extension\Core\Type\DateTimeType  
container Symfony\Component\Form\Extension\Core\Type\EmailType  
container Symfony\Bridge\Doctrine\Form\Type\EntityType  
container Symfony\Component\Form\Extension\Core\Type\FileType  
container Symfony\Component\Form\Extension\Core\Type\FormType  
container Symfony\Component\Form\Extension\Core\Type\HiddenType  
container Symfony\Component\Form\Extension\Core\Type\IntegerType  
container Symfony\Component\Form\Extension\Core\Type\LanguageType  
container Symfony\Component\Form\Extension\Core\Type\LocaleType  
container Symfony\Component\Form\Extension\Core\Type\MoneyType  
container Symfony\Component\Form\Extension\Core\Type\NumberType  
container Symfony\Component\Form\Extension\Core\Type>PasswordType  
container Symfony\Component\Form\Extension\Core\Type\PercentType  
container Symfony\Component\Form\Extension\Core\Type\RadioType  
container Symfony\Component\Form\Extension\Core\Type\RepeatedType  
container Symfony\Component\Form\Extension\Core\Type\ResetType  
container Symfony\Component\Form\Extension\Core\Type\SearchType  
container Symfony\Component\Form\Extension\Core\Type\SubmitType  
container Symfony\Component\Form\Extension\Core\Type\TextType  
container Symfony\Component\Form\Extension\Core\Type\TextareaType  
container Symfony\Component\Form\Extension\Core\Type\TimeType  
container Symfony\Component\Form\Extension\Core\Type\TimezoneType  
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

```
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

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

ou

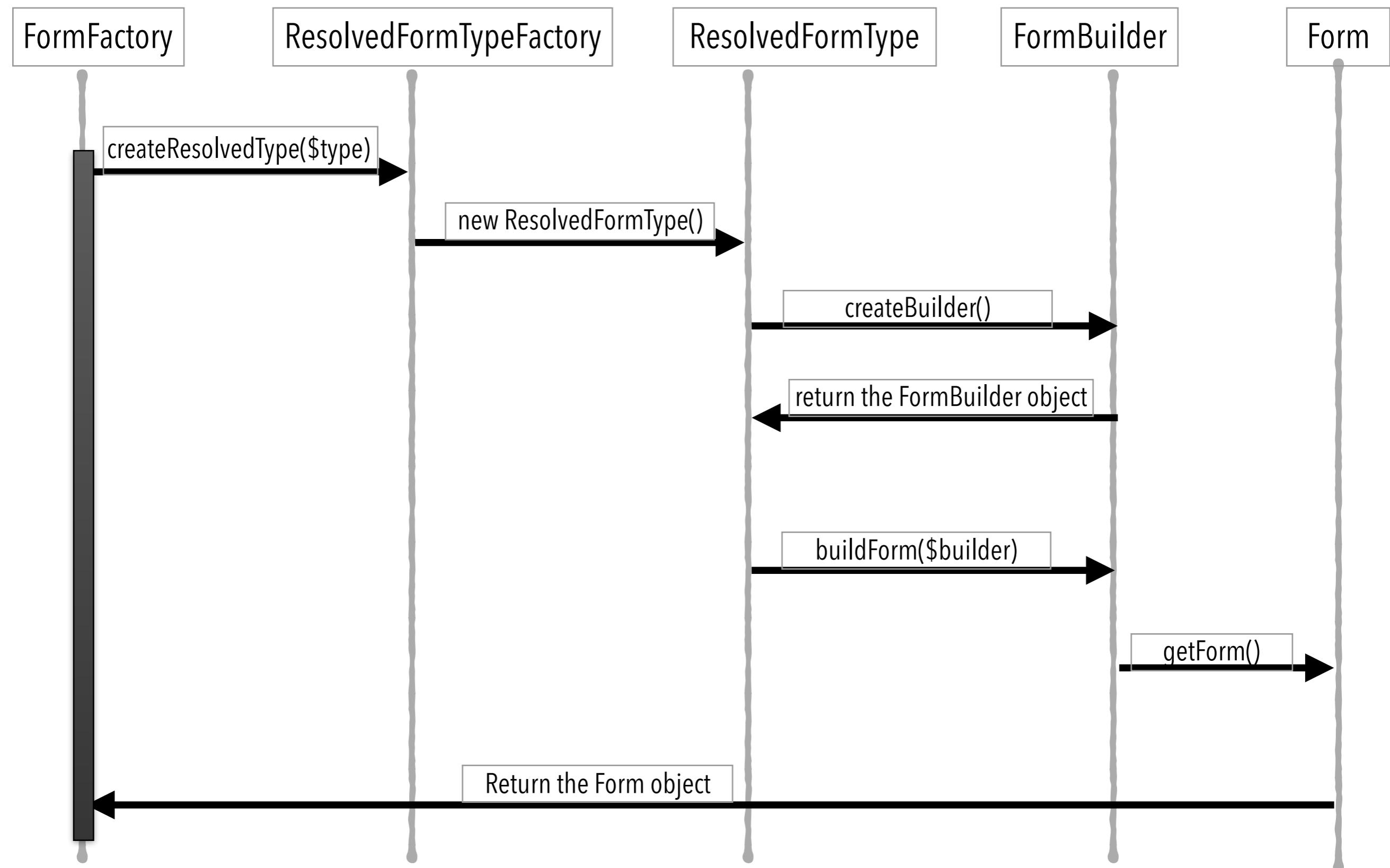
```
$form = $this->createForm('article', $article);
```

ou

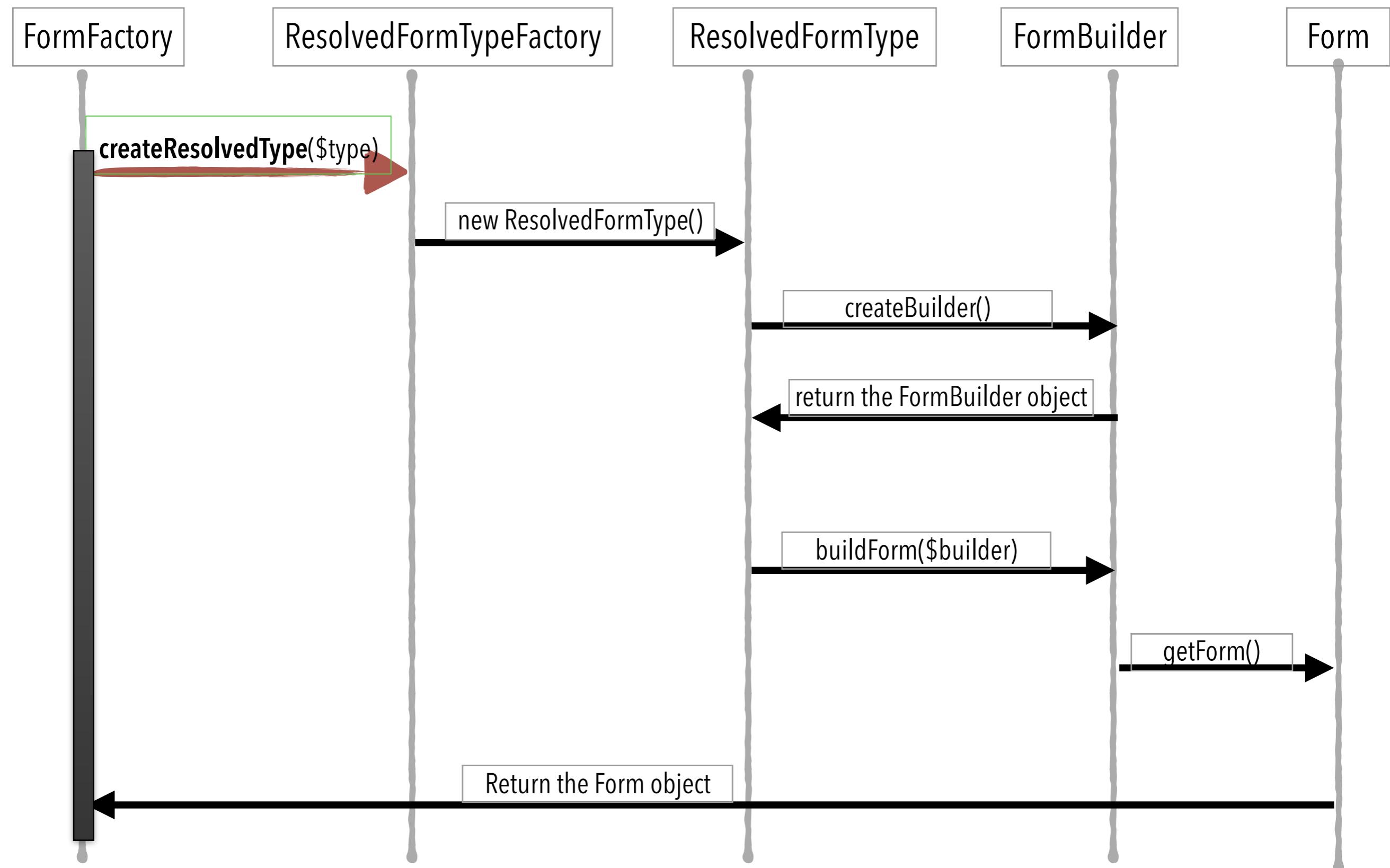
```
$this->container->get('form.factory')->create($type, $data, $options);
```

Symfony\Component\Form\FormFactory

Vue d'ensemble



Vue d'ensemble

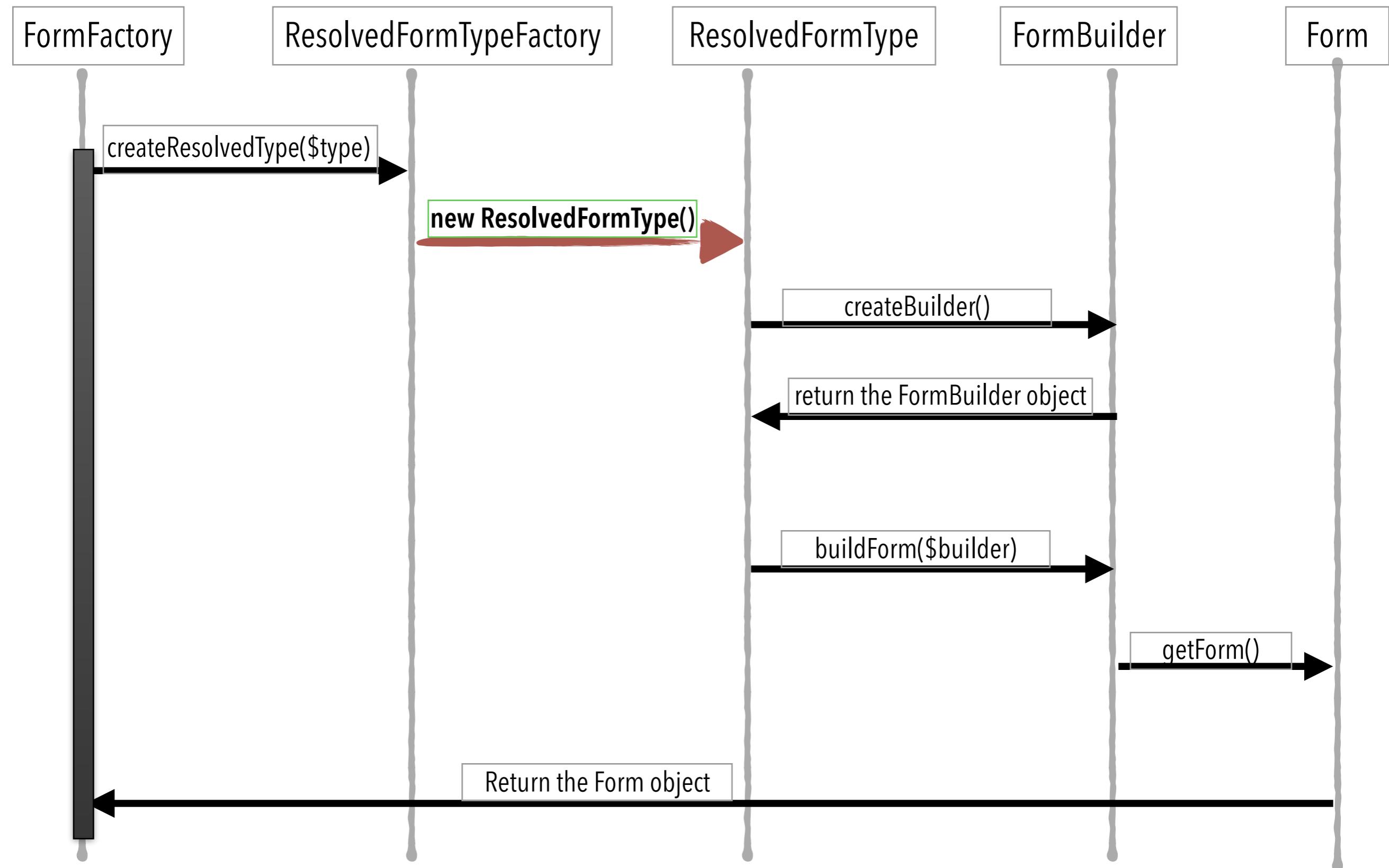


Résolution dynamique d'un type

```
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

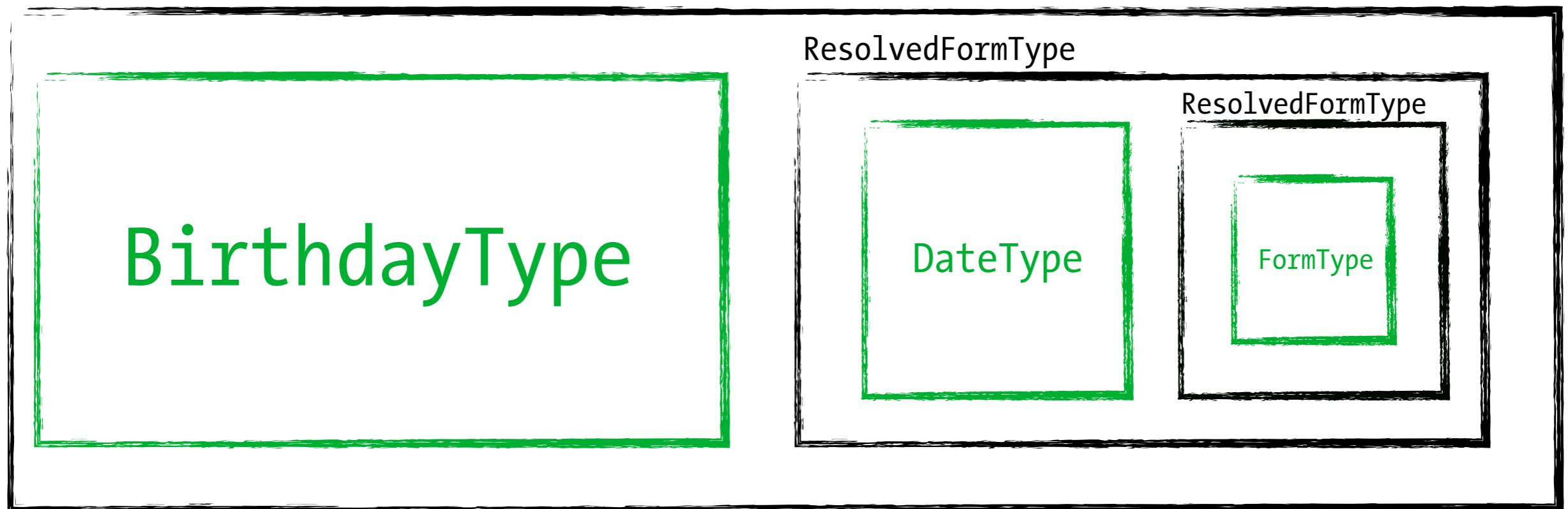


Composition d'un type résolu

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

Composition d'un type résolu

ResolvedFormType



Collecter les infos de chaque type

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

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

Débogage des formulaires

The screenshot shows the Symfony Profiler interface with the 'FORMS' tab selected. The left sidebar lists various sections: CONFIG, REQUEST, EXCEPTION, EVENTS, LOGS, TIMELINE, ROUTING, FORMS (with a '1' badge), and SECURITY. The main content area displays a profile for a GET request to `http://www.symfony-training.local/app_dev.php/article/new` at `Sun, 06 Apr 2014 17:33:37 +0200`. The 'article' form is selected in the 'Forms' tree. The 'Default Data' section shows the model format as 'same as normalized format', the normalized format as 'Object(Live\ArticleBundle\Entity\Article)', and the view format as 'same as normalized format'. The 'Submitted Data' section notes that the form was not submitted. The 'Passed Options' section is empty. The 'Option' column has a tooltip: 'This form was not submitted.'

article [article]

Default Data -

Model Format	<i>same as normalized format</i>
Normalized Format	Object(Live\ArticleBundle\Entity\Article)
View Format	<i>same as normalized format</i>

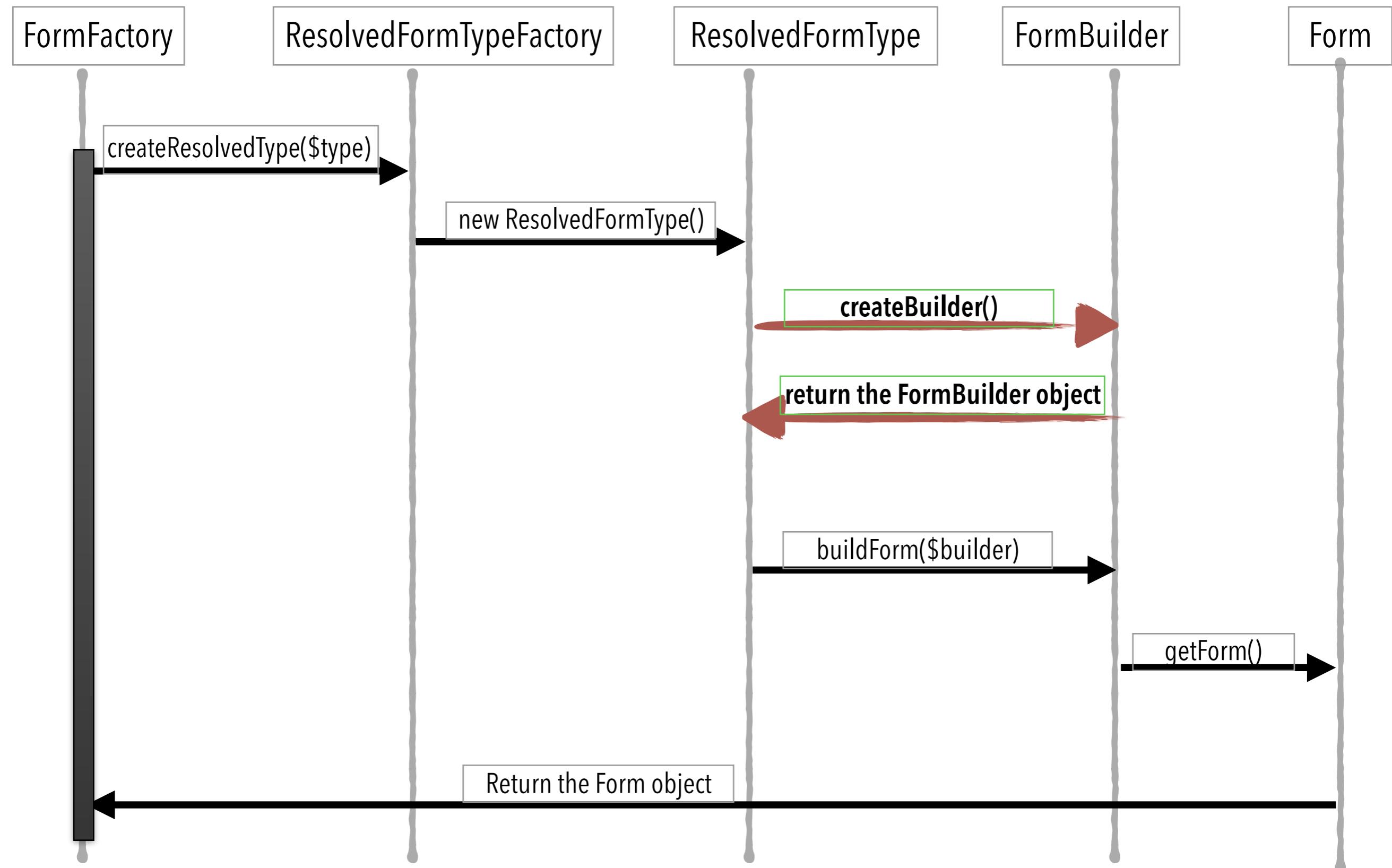
Submitted Data -

This form was not submitted.

Passed Options -

Option	Passed Value	Resolved Value

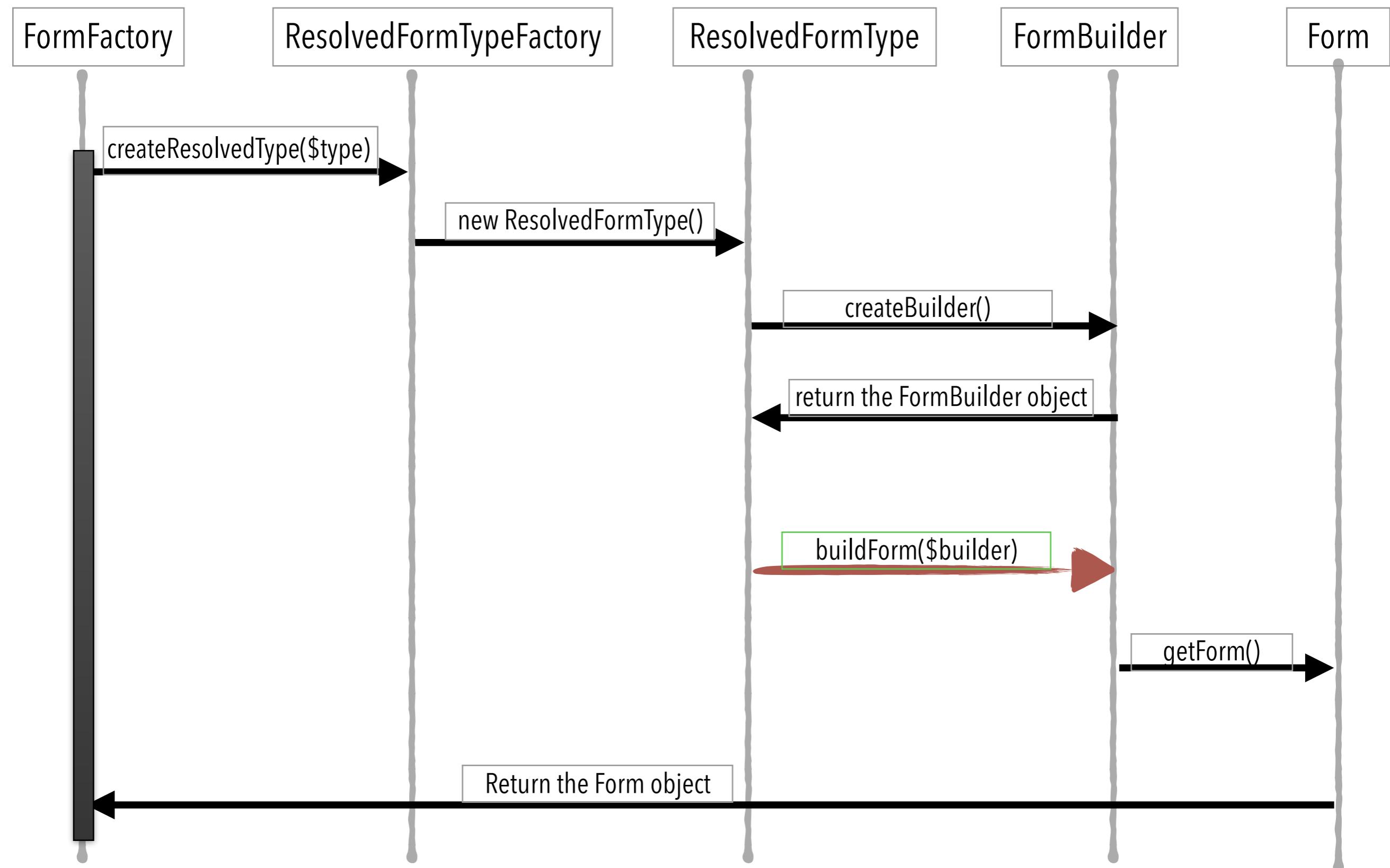
Vue d'ensemble



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

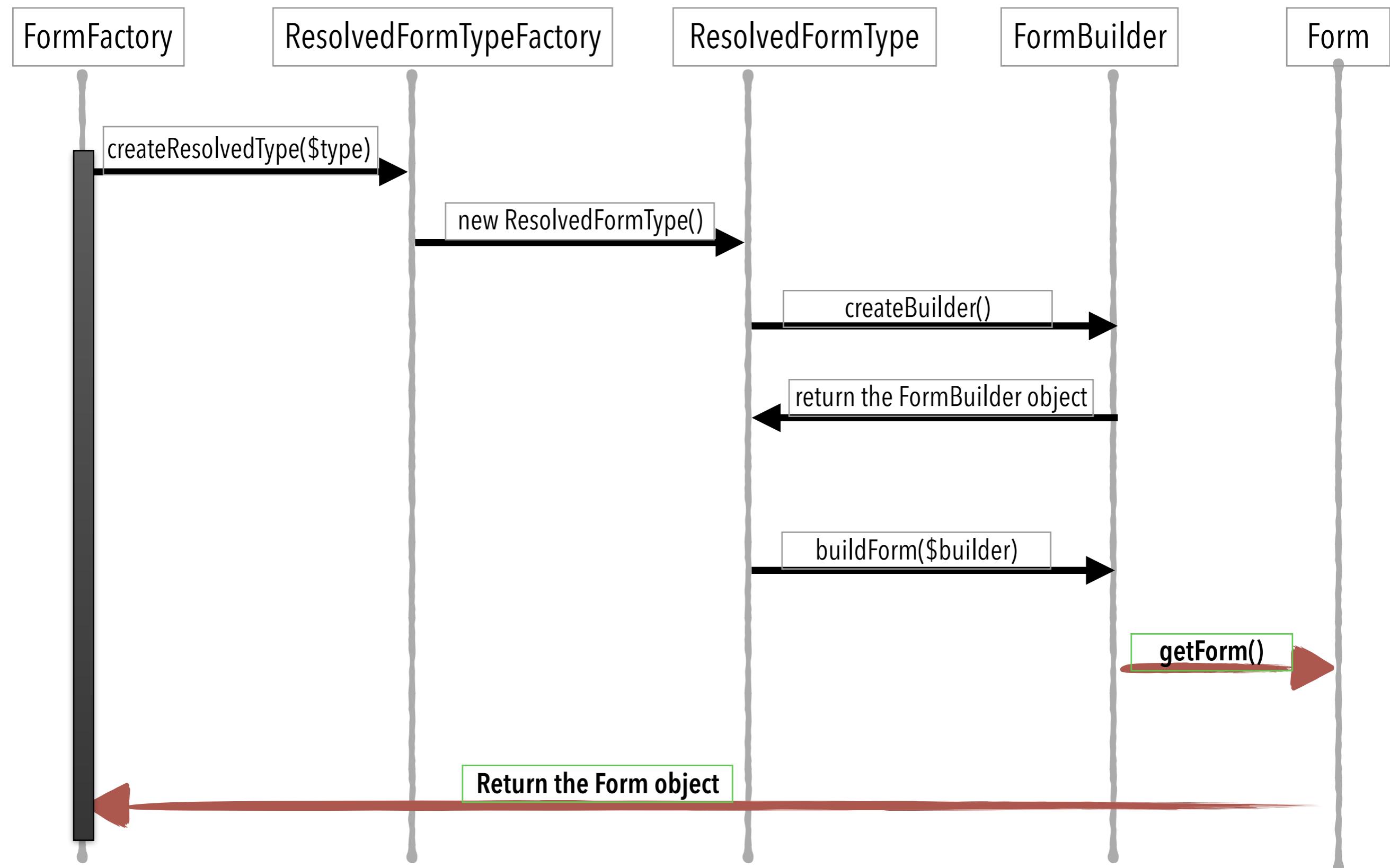
```
namespace Live\ArticleBundle\Form;

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()

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

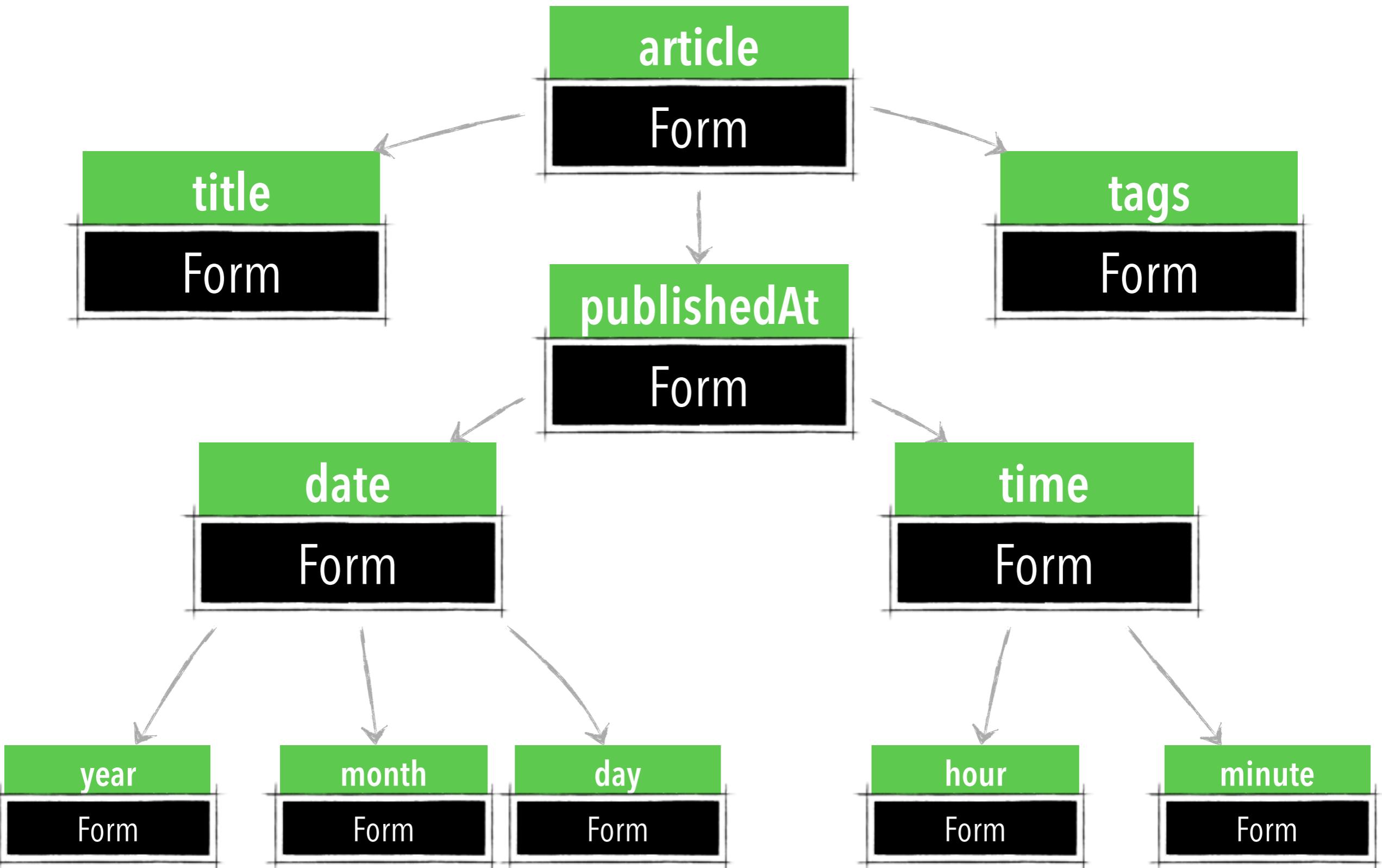
// Initialisation des valeurs par défaut
$form->setData($article);
```

Création du formulaire

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

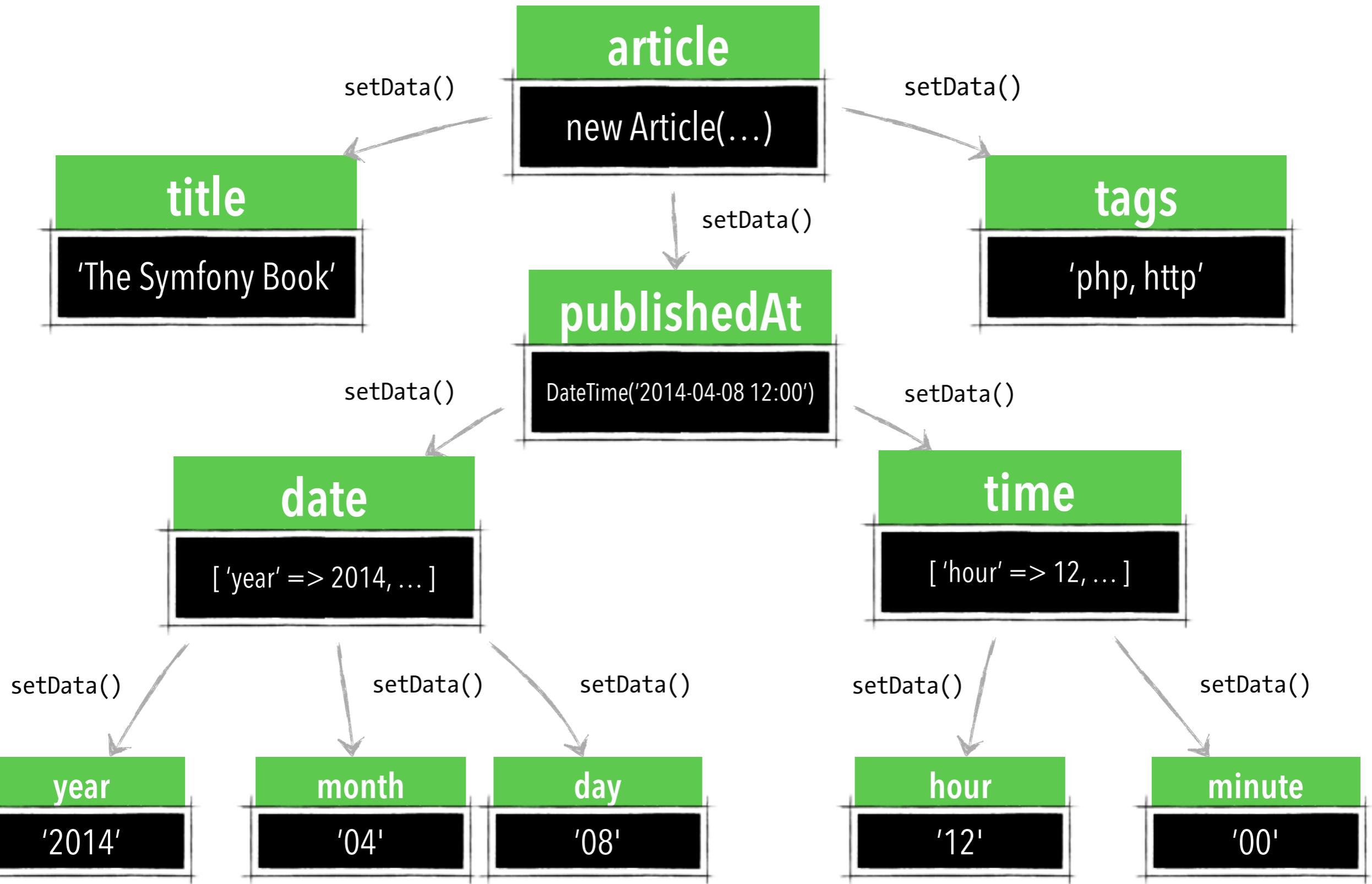
// Initialisation des valeurs par défaut
$form->setData($article);
```

Et voilà ! Le formulaire

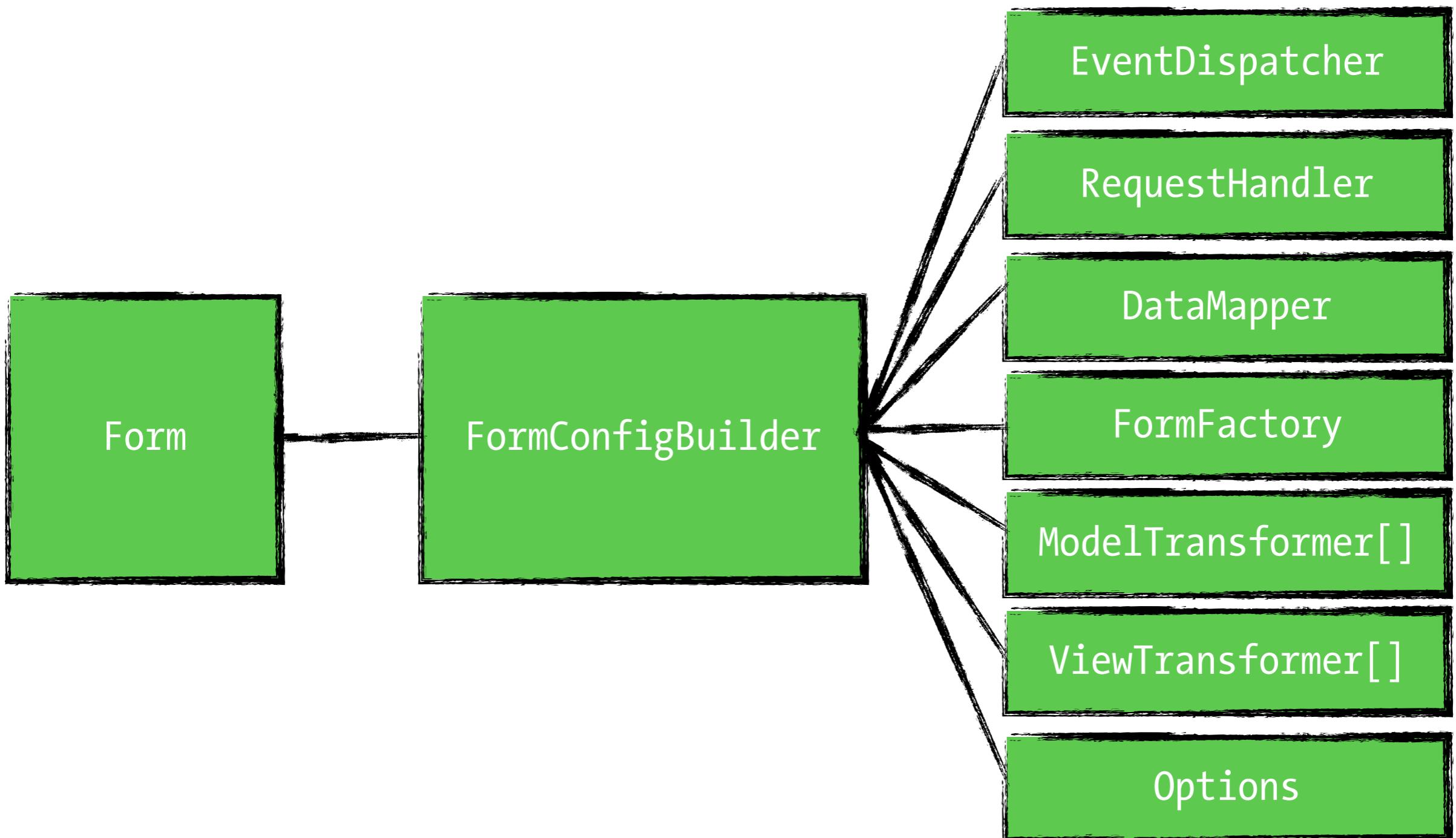


Initialisation des valeurs par défaut

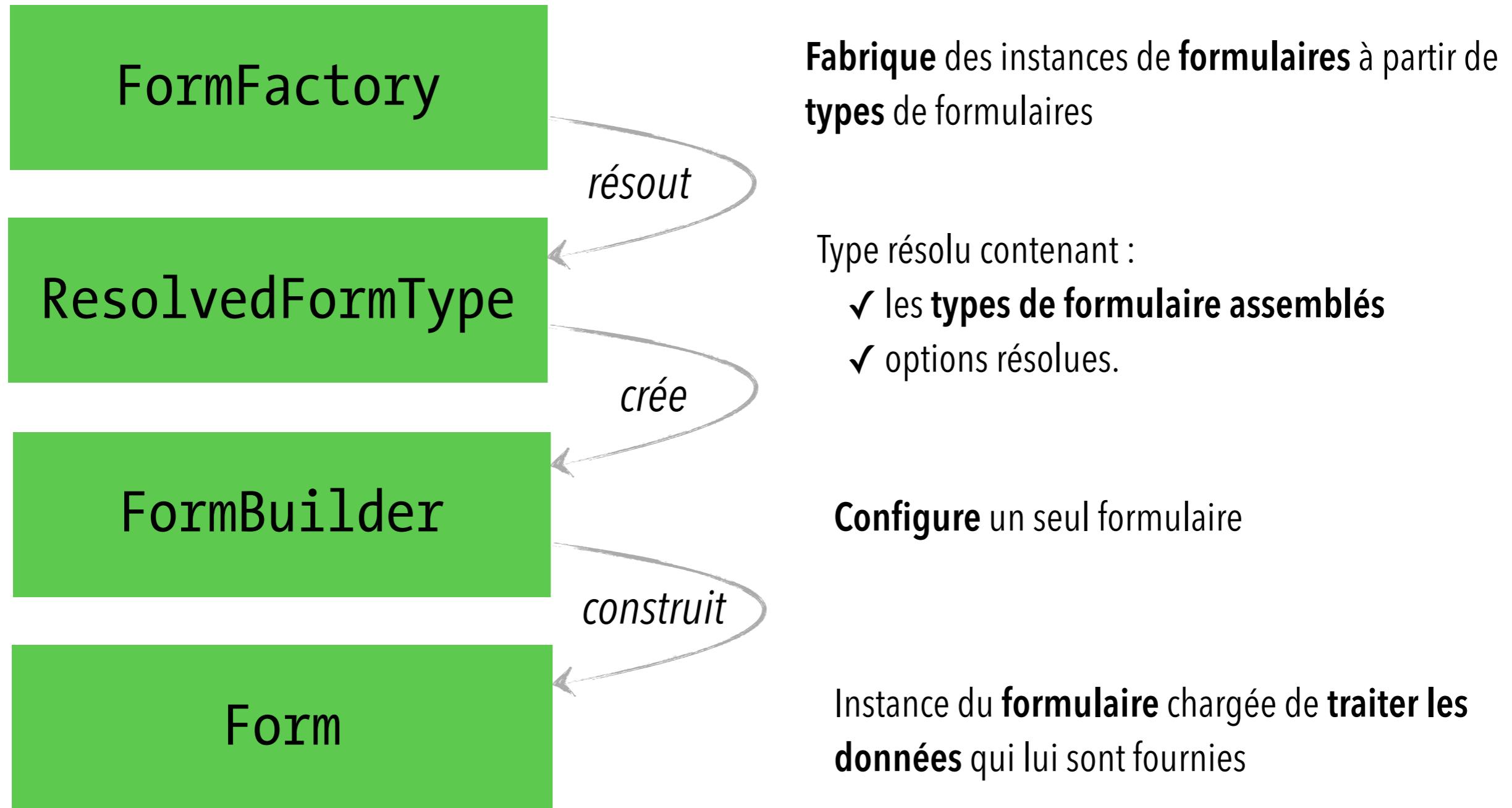
```
$form->setData($article);
```



Composition d'un formulaire



Récapitulons





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

Normalisation des données

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.

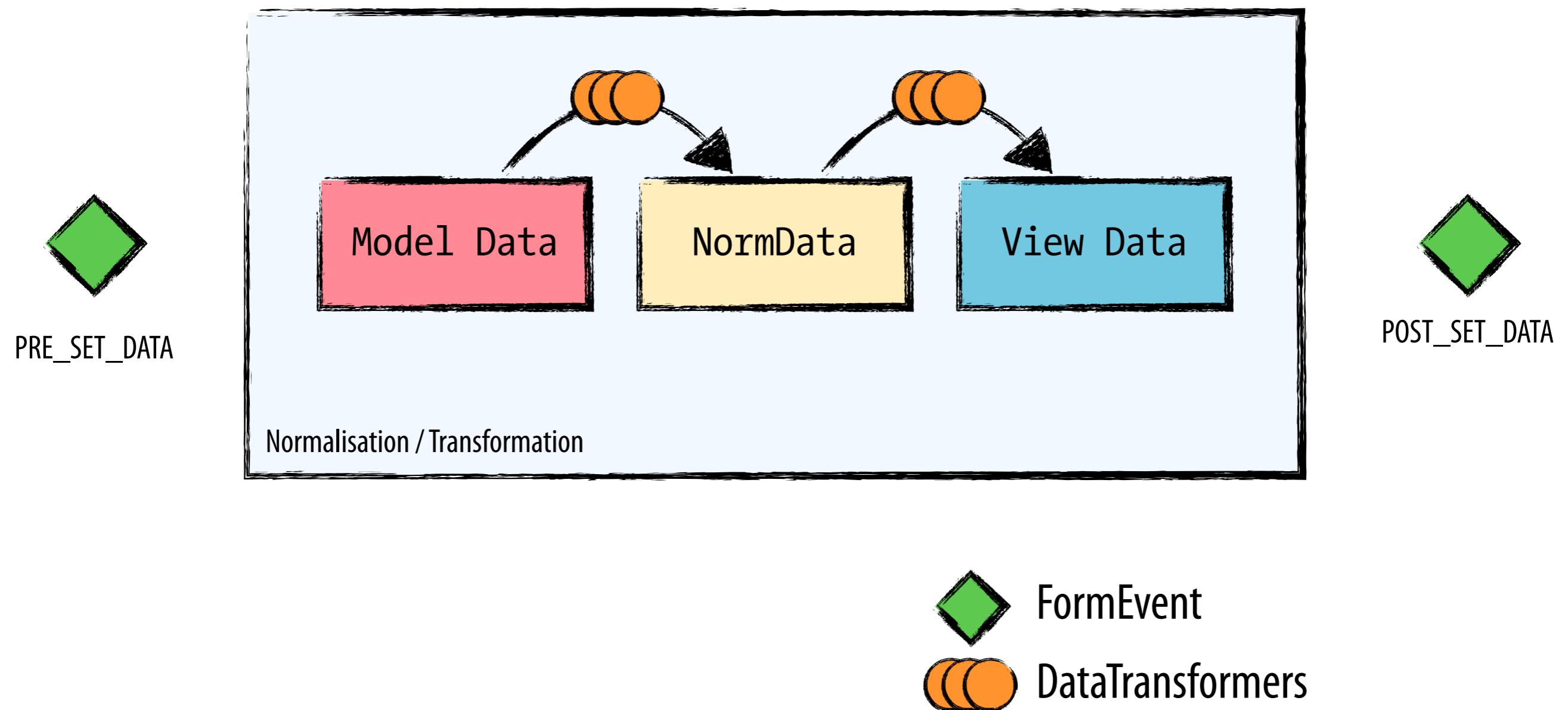
Ne pas changer le contenu de la donnée mais **seulement sa représentation**

Changer la représentation

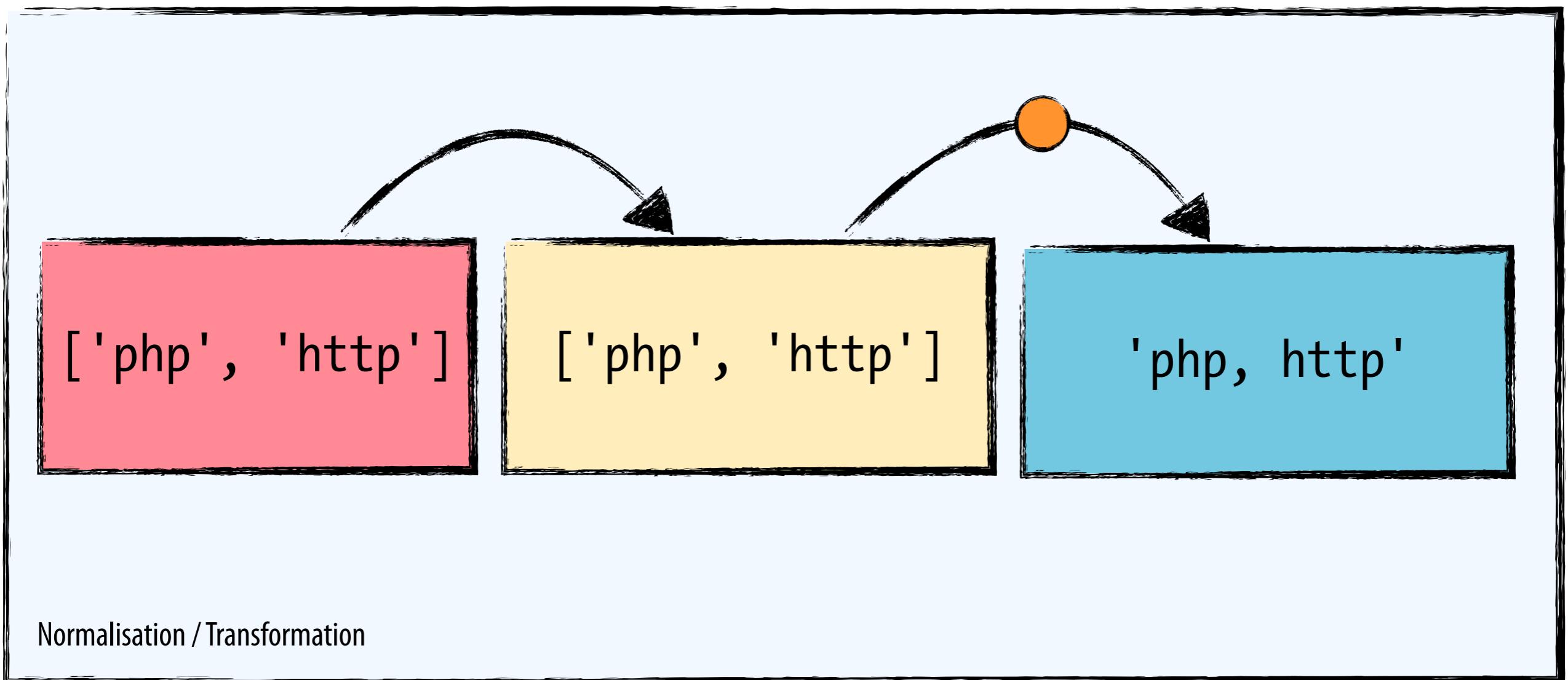
'2014-04-08' => '2015-09-10'

'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

```
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 ?

```
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 !

```
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'));
```

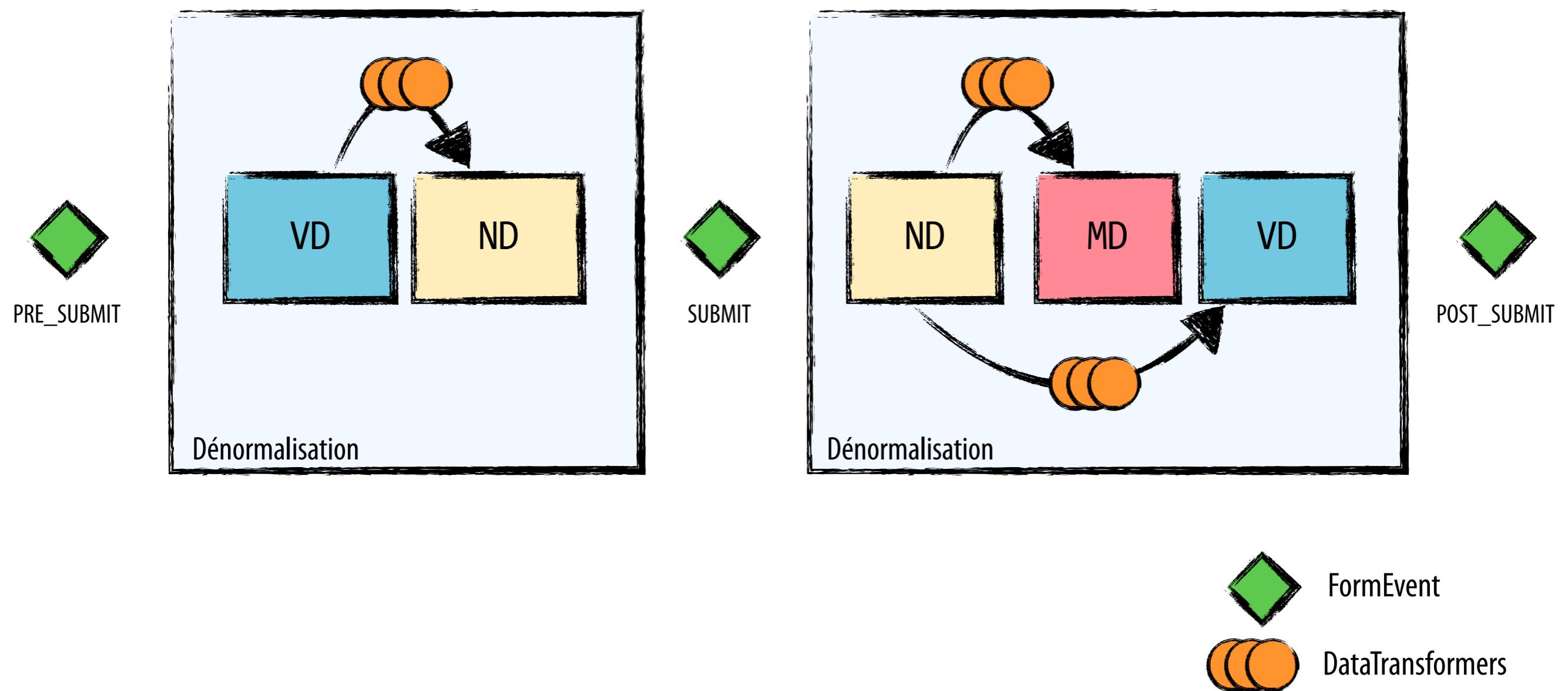
ou

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

HttpFoundationRequestHandler

- Vérifie 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

```
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

PRE_SET_DATA	FormEvent(\$form, \$modelData)
POST_SET_DATA	FormEvent(\$form, \$modelData)
PRE_SUBMIT	FormEvent(\$form, \$submittedData)
SUBMIT	FormEvent(\$form, \$normData)
POST_SUBMIT	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 »

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

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

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

Enregister un « event subscriber »

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

A vous de jouer !



<https://www.flickr.com/photos/bigbrotherbackpacking>

job@sensiolabs.com



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

Fin

hugo.hamon@sensiolabs.com // sarah.khalil@sensiolabs.com // [@poledev](https://twitter.com/poledev)