

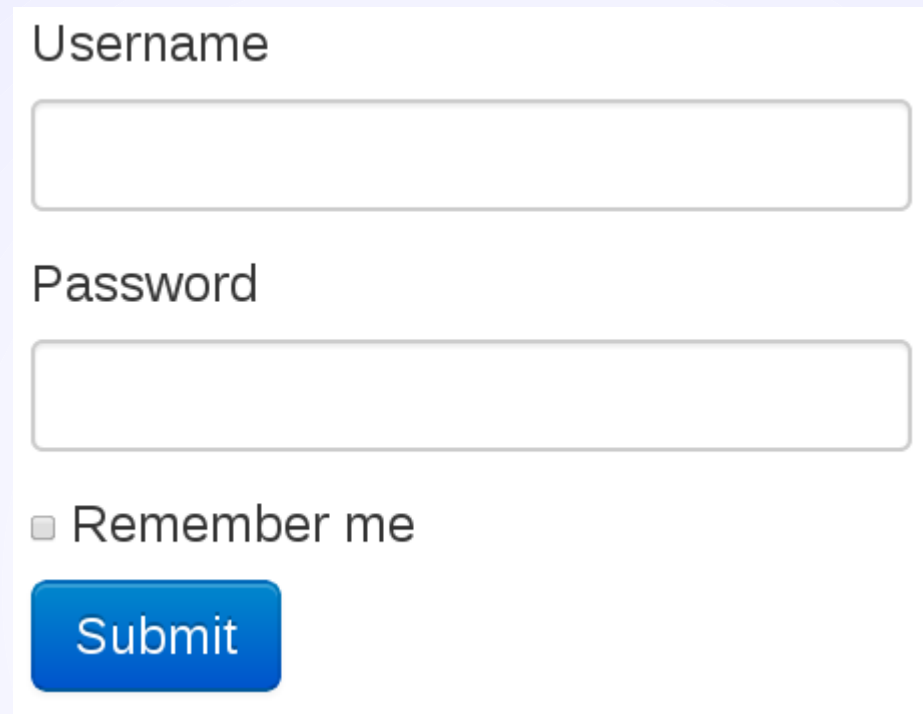
SensioLabs

Formulaires Symfony2

Cas pratiques et explications

Alexandre Salomé – sfPot mai 2013

- Pré-requis
 - Avoir lu la documentation des formulaires
- 4 cas pratiques
 - Un formulaire de login
 - Changer de mot de passe (ancien/nouveau)
 - Traductions avec Doctrine
 - Masquer les champs pour certains utilisateurs
- Explications à chaque cas



Username

Password

☐ Remember me

Submit

Cas n°1 – Formulaire de login

« Le formulaire de connexion n'utilise pas le composant Form, ce qui nous oblige à dupliquer le HTML. Il faut utiliser le composant Form »

Requis par la couche de sécurité

4/51

- POST /login-check

_username

_password

_csrf_token (*intention : authenticate*)

```
$this->get('form.csrf_provider')->generateCsrfToken('authenticate')
```

_remember_me

Login – Le formulaire

5/51

```
class LoginType extends AbstractType
{
    public function buildForm($builder, $options)
    {
        $builder
            ->add('_username', 'text')
            ->add('_password', 'password')
            ->add('_remember_me', 'checkbox')
            ->add('submit', 'submit'); // new!
    }

    public function setDefaultOptions($resolver)
    {
        $resolver->setDefaults(array(
            'csrf_field_name' => '_csrf_token',
            'intention'       => 'authenticate',
        ));
    }

    public function getName() { return 'login'; }
}
```

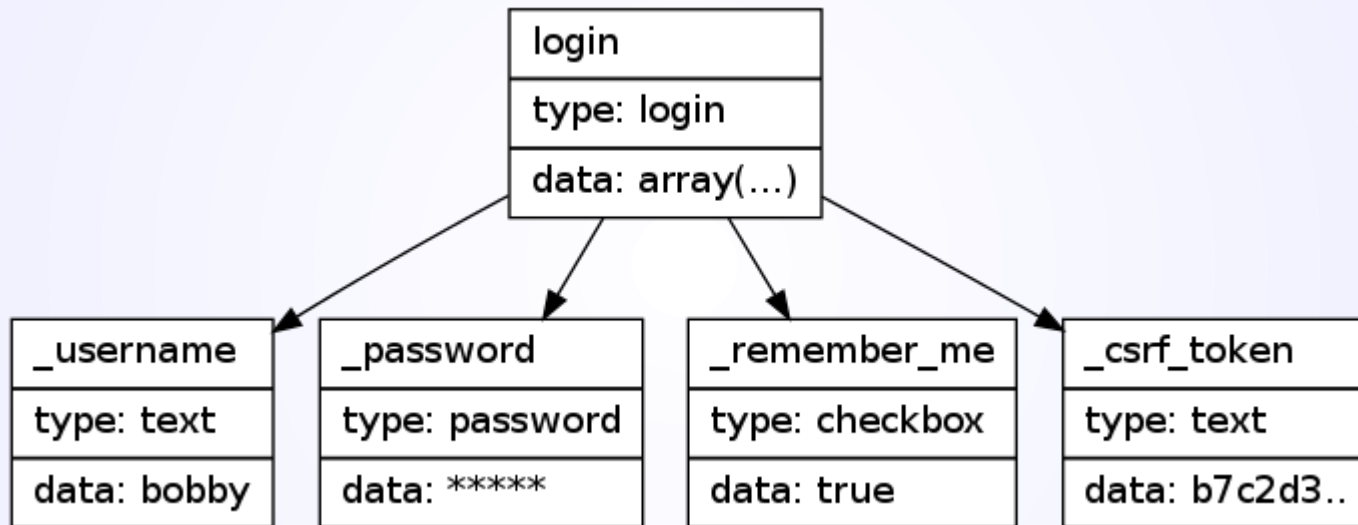
Le conteneur de services

6/51

```
<service id="form.type.login" class="Backlog\Form\Type\LoginType">  
  <tag name="form.type" alias="login" />  
</service>
```

Construction de formulaire

7/51



Formulaires dans les contrôleurs

8/51

```
$this->createForm($type, $data, $options)
```

```
// équivalent à
```

```
$this
```

```
    ->get('form.factory')
```

```
    ->create($type, $data, $options)
```

```
;
```


FormFactory

9/51

```
$this  
    ->get('form.factory')  
    ->create('login')  
;
```

```
login[_username]  
login[_password]  
login[_remember_me]  
login[_csrf_token]
```

```
$this  
    ->get('form.factory')  
    ->createNamed('foo', 'login')  
;
```

```
foo[_username]  
foo[_password]  
foo[_remember_me]  
foo[_csrf_token]
```

```
$this  
    ->get('form.factory')  
    ->createNamed('', 'login')  
;
```

```
_username  
_password_  
_remember_me  
_csrf_token
```

Login – Le contrôleur (1 / 2)

10/51

```
public function loginAction()  
{  
    $form = $this->get('form.factory')  
        ->createNamed('', 'login', array(  
            'action' => $this->generateUrl('session_loginCheck')  
        ));  
  
    if ($error = $this->getErrorMessage()) {  
        $form->addError(new FormError($error));  
    }  
  
    return $this->render('...', array(  
        'form' => $form->createView()  
    ));  
}
```

Login – Le contrôleur (2/2)

11/51

```
protected function getErrorMessage()  
{  
    $request = $this->getRequest();  
    $attrs    = $request->attributes;  
    $session  = $request->getSession();  
  
    if ($attrs->has(SecurityContext::AUTHENTICATION_ERROR)) {  
        $error = $attrs->get(SecurityContext::AUTHENTICATION_ERROR);  
    } else {  
        $error = $session->get(SecurityContext::AUTHENTICATION_ERROR);  
        $session->remove(SecurityContext::AUTHENTICATION_ERROR);  
    }  
  
    return $error instanceof \Exception ?  
        $error->getMessage()  
        : $error  
    ;  
}
```

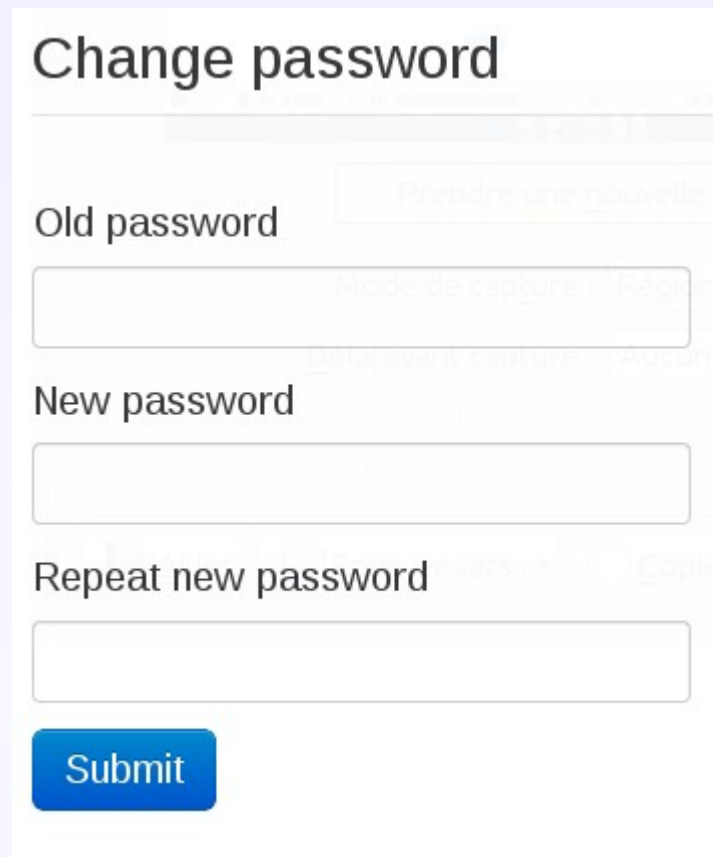
Login – Le template

12/51

```
{{ form(form) }}
```

Cas n°1 – Conclusion

- Construction d'un formulaire
- Paramétrage du CSRF (nom de champ + intention)
- Flexibilité grâce à la FormFactory
- Réutilisation des templates



A screenshot of a web form titled "Change password". The form is set against a light yellow background. It contains three text input fields: "Old password", "New password", and "Repeat new password". Each field is a simple rectangular box with a thin border. Below the "Repeat new password" field is a blue rectangular button with the word "Submit" in white text. The form is centered on the page.

Cas n°2 – Changement de MDP

« Je veux que l'utilisateur saisisse son ancien mot de passe pour en mettre un nouveau »

Cycle de vie du formulaire

15/51

	Construction FormBuilder	Utilisation Form	Soumission \$form->bind
Listeners	Modifiable	Lecture seulement	Lecture seulement
Attributs & Options	Modifiable	Lecture seulement	Lecture seulement
(Data View) Transformers	Modifiable	Lecture seulement	Lecture seulement
Enfants / Parents	Modifiable	Modifiable	Lecture seulement

ProfileType

16/51

```
class ProfileType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array
$options)
    {
        $builder
            ->add('fullname', 'text')
            ->add('initials', 'text')
            ->add('change_password', 'change_password', array(
                'virtual' => true
            ))
            ->add('submit', 'submit')
        ;
    }

    public function getName()
    {
        return 'profile';
    }
}
```


ChangePasswordType

17/51

```
class ChangePasswordType extends AbstractType
{
    // ...

    public function buildForm(FormBuilderInterface $builder, array
        $options)
    {
        $encoderFactory = $this->encoderFactory;

        $builder
            ->add('old_password', 'password', array(
                'mapped' => false
            ))
            ->add('new_password', 'repeated', array(
                'mapped' => false,
                'type' => 'password'
            ))
    }
}
```

change_password

18/51

- Data = user
- Injection du service d'encodage
- À la soumission du formulaire
 - Vérifie le mot de passe
 - Ajoute un message d'erreur si le MDP est incorrect
 - Enregistre le nouveau mot de passe

Le formulaire

19/51

```
class ChangePasswordType extends AbstractType
{
    // ...

    public function buildForm(FormBuilderInterface $builder, array
        $options)
    {
        // ...

        $builder->addEventListener(FormEvents::POST_SUBMIT,
            function (FormEvent $event) use ($encoderFactory) {
                ...
            });
    }
}
```

Le formulaire

20/51

```
function (FormEvent $event) use ($encoderFactory) {
    $form      = $event->getForm();
    $user      = $form->getData();
    $encoder   = $encoderFactory->getEncoder($user);

    $oldPassword = $form->get('old_password')->getData();
    $newPassword = $form->get('new_password')->getData();

    if (!$oldPassword || !$newPassword) {
        return;
    }

    if (!$user->isPasswordValid($oldPassword, $encoder)) {
        $form->addError(new FormError('Bad credentials'));

        return;
    }

    $user->setPassword($newPassword, $encoder);
}
```

Déclaration du form type

21/51

```
<service id="form.type.change_password" class="...">  
  <argument type="service" id="security.encoder_factory" />  
  <tag name="form.type" alias="change_password" />  
</service>
```

Cas n°2 – Conclusion

- Le cycle de vie d'un formulaire
- Le rôle des FormType
- Les listeners pour interagir après la construction
- virtual = true
 - Partage la donnée avec le sous-formulaire
- mapped = false
 - Permet de « hooker » dans un FormType

Code UPC

fr_FR **+** en_US

Title

Baseline

Cas n°3 – Traductions & Doctrine

*« Je veux gérer mes traductions
en Javascript de manière homogène »*

Le contrat

- Contrôleur intact
- Modèle de données explicite (oneToMany)
- Mise en commun au niveau des formulaires

Modèle Doctrine (1 / 2)

25/51

```
Acme\Entity\Product:
  type: entity
  id: ~
  fields:
    upc: {type: string, length: 64, nullable: true }
    price: {type: price, nullable: true }
  oneToMany:
    translations:
      targetEntity: ProductTranslation
      mappedBy: product
      indexBy: culture
      cascade: [ persist, remove ]
```

Modèle Doctrine (2/2)

26/51

```
Acme\Entity\ProductTranslation:
  type: entity
  id: ~
  fields:
    culture: {type: string, length: 8 }
    title: {type: string, length: 255, nullable: true }
    baseline: {type: text, nullable: true }
  manyToOne:
    product:
      targetEntity: Product
      inversedBy: translations
      joinColumn:
        name: product_id
        referencedColumnName: id
```

Le contrat – le contrôleur

27/51

```
public function editAction(Request $request, $id)
{
    $product = ...;

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

    if ($request->isMethod('POST') &&
        $form->bind($request)->isValid()
    ) {
        // save and flush
    }

    return $this->render(...);
}
```

Le contrat – l'API commune

28/51

```
$product->getUpc();  
$product->setUpc($upc);
```

```
$product->getTranslations();  
$trans = $product->getTranslation('fr_FR');
```

```
$trans->getTitle();  
$trans->setTitle('Product title');
```

```
$trans->getBaseline();  
$trans->setBaseline('Baseline of the product');
```

```
$product->removeTranslation($trans);  
$product->addTranslation(new ProductTranslation(...));
```

Le formulaire

29/51

```
class TranslationsType extends AbstractType
{
    public function setDefaultOptions(OptionsResolverInterface $resolver)
    {
        $resolver->setDefaults(array(
            'allow_add'      => true,
            'allow_delete'   => true,
            'prototype'      => true,
            'by_reference'   => false
        ));
    }

    public function getParent()
    {
        return 'collection';
    }

    public function getName()
    {
        return 'translations';
    }
}
```

Le formulaire

30/51

```
class TranslationsType extends AbstractType
{
    public function __construct($defaultCulture, array $availableCultures =
array())
    {
        $this->availableCultures = $availableCultures;
        $this->defaultCulture    = $defaultCulture;
    }

    public function buildView(FormView $view, FormInterface $form, array
$options)
    {
        $cultures = $this->availableCultures;
        $existing = array_keys($form->all());

        $view->vars['missing_cultures'] = array_diff($cultures,
$existing);
        $view->vars['default_culture'] = $this->defaultCulture;
    }
}
```

Templating

31/51

Nom :

Alexandre

Prénom :

Salomé

E-mail :

alexandre@...

Cet e-mail est déjà utilisé

Il est interdit d'utiliser le domaine « @... »

Templating

32/51

Nom: LABEL	Alexandre WIDGET
Prénom: LABEL	Salomé WIDGET
E-mail: LABEL	alexandre@... WIDGET
	ERRORS Cet e-mail est déjà utilisé Il est interdit d'utiliser le domaine « @... »

Templating

33/51

Nom	Alexandre
Prénom	Salomé
E-mail	alexandre@...
	Cet e-mail est déjà utilisé Il est interdit d'utiliser le domaine « @... »

Nom :

Alexandre
ROW

Prénom :

Salomé
ROW

E-mail :

alexandre@...

ROW

Cet e-mail est déjà utilisé

Il est interdit d'utiliser le domaine « @... »

form_div_layout.html.twig

35/51

Ce fichier comporte les blocs utilisés pour le rendu des formulaires. Les plus fréquents sont les suivants :

```
{% block ..._widget %}  
{% block ..._label %}  
{% block ..._errors %}  
{% block ..._row %}
```

Au moment du rendu, le moteur cherche un bloc dans le template correspondant au type ou au parent le plus proche. Par exemple pour le type « birthday » :

```
{% block birthday_row %}  
{% block      date_row %}  
{% block      form_row %}
```

La vue

36/51

```
{% block translations_row %}  
{% spaceless %}
```

...

```
{% endspaceless %}  
{% endblock %}
```



Code UPC

fr_FR

+ en_US

Title

Titre de mon produit

Baseline

Baseline de mon produit

```
{% block translations_row %}
{% spaceless %}

    <div class="translations-container" data-id="{{ id }}" {% if prototype is defined
%}data-prototype="{{ form_widget(prototype)|e }}" {% endif %}>
        <ul class="nav nav-tabs nav-translations">
            {% for key, subForm in form.children %}
                <li{{ key == default_culture ? ' class="active"' : '' }}>
                    <a data-toggle="tab" href="#{{ id }}-{{ key }}">{{ key }}</a>
                </li>
            {% endfor %}
            {% if prototype is defined %}
                {% for culture in missing_cultures %}
                    <li><a data-toggle="tab" href="#{{ id }}-{{ culture }}"
data-translation-create="{{ culture }}"><i class="icon-plus"></i> {{ culture }}</a></li>
                {% endfor %}
            {% endif %}
        </ul>
        <div class="form-translations">
            {% for key, subForm in form.children %}
                <div class="tab-pane{{ loop.first ? ' active' : '' }}" id="{{ id }}-{{ key }}"
                    {{ form_widget(subForm) }}
                </div>
            {% endfor %}
        </div>
    </div>

    <hr />
{% endspaceless %}
{% endblock %}
```

Le Javascript

38/51

```
$(document).on('click', '.nav-translations a[data-translation-create]',  
function (e) {  
    e.preventDefault();  
    var $link      = $(e.currentTarget);  
    var $container  = $($link.parents(".translations-container")[0]);  
    var $translations = $container.find('.form-translations');  
    var culture     = $link.attr('data-translation-create');  
    var prototype   = $container.attr('data-prototype');  
    var id          = $container.attr('data-id');  
  
    prototype = prototype.replace(/__name__/g, culture) ;  
    $link.find('i.icon-plus').remove();  
    $translations.find(".tab-pane").removeClass('active');  
  
    var newCulture = $translations.append(  
        '<div class="tab-pane active" id="' + id + '-' + culture + '">  
        + prototype  
        + '</div>  
    );  
  
    $link.removeAttr('data-translation-create');  
});
```

Déclaration du form type

39/51

```
<service id="form.type.translations" class="...">  
  <tag name="form.type" alias="translations" />  
</service>
```

Implémentation

40/51

```
class ProductType extends AbstractType
{
    public function buildForm(...)
    {
        $builder
            ->add('upc', 'text')
            ->add('active', 'checkbox')
            ->add('translations', 'translations', array(
                'type' => 'product_translation'
            ))
    }
}
```


Conclusion

- Mise en commun des templates de formulaires
- Étendre un type de formulaire
- Relation entre formulaire et modèle

Code UPC

fr_FR [+ en_US](#)

Title

Slug

Cas n°4 – Champs cachés

« Je veux masquer certains champs pour certains utilisateurs »

Extension de type

- Réutilisation horizontale de comportements
- Exemples
 - CSRF
 - `FormTypeCsrfExtension`
 - Validation
 - `FormTypeValidatorExtension`

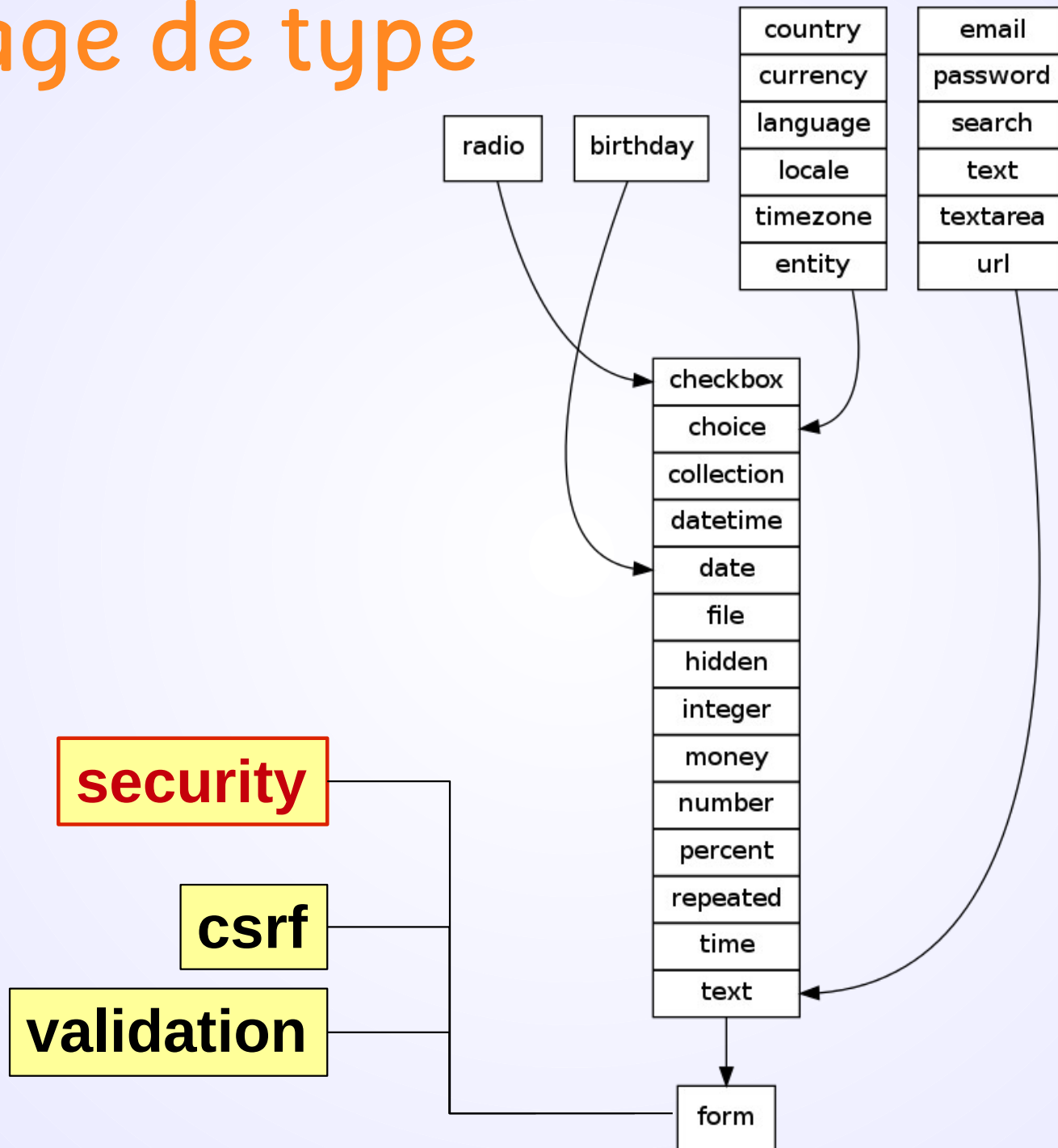
FormType vs FormTypeExtension

44/51

FormType	FormTypeExtension
buildForm	
buildView	
finishView	
setDefaultOptions	
getParent	
getName	
	getExtendedType

Héritage de type

45/51



SecurityTypeExtension (1 / 3)

46/51

```
class SecurityTypeExtension extends AbstractTypeExtension
{
    public function __construct(SecurityContextInterface $context)
    {
        $this->context = $context;
    }

    /**
     * {@inheritdoc}
     */
    public function getExtendedType()
    {
        return 'form';
    }
}
```

SecurityTypeExtension (2/3)

47/51

```
// ...

public function buildForm(FormBuilderInterface $builder, array $options)
{
    $g = $options['is_granted'];
    if (null === $g || $this->context->isGranted($g)) {
        return;
    }

    $builder->addEventListener(FormEvents::PRE_SET_DATA,
    function (FormEvent $event) {
        $form = $event->getForm();
        if ($form->isRoot()) // ...

        $form->getParent()->remove($form->getName());
    });
}

public function setDefaultOptions(OptionsResolverInterface $resolver)
{
    $resolver->setDefaults(array('is_granted' => null));
}
```

SecurityTypeExtension (3/3)

48/51

```
<service id="form.type_extension.security" class="...">  
  <argument type="service" id="security.context" />  
  <tag name="form.type_extension" alias="form" />  
</service>
```


Utilisation

49/51

```
class ProductType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array
$options)
    {
        $builder
            ->add('upc', 'text', array('label' => 'Code UPC'))
            ->add('translations', 'translations', array(
                'type' => 'product_translation',
                'is_granted' => 'ROLE_MODERATOR'
            ))
        ;
    }
}
```

Cas n°4 – Conclusion

- Implémentation simple et utilisation rapide
- Utiliser les options pour configurer
 - `is_granted` / `is_granted_subject`
 - `data_as_subject`

Fin

Questions ?