



# Formulaires multiples

# Mise en place

---

- Créez une nouvelle entité appelée Catégorie qui possèdera les champs suivants :
  - Nom : string
  - Couleur : string
- Reliez la classe Categorie avec la classe Produit par une relation ManyToMany
- Créez une migration et synchronisez la base de données.
- Créez la classe formulaire correspondant à la catégorie.

# Modification du formulaire produit

---

- Modifiez la classe formulaire `ProduitType` de la façon suivante
- Ajoutez la possibilité d'inclure dans le formulaire produit le formulaire catégorie :


```
->add("categories", CollectionType::class, [  
    'entry_type' => CategorieType::class,  
    'entry_options' => ['label' => false],  
    'by_reference' => false,  
])
```

# Controleur

---


- Modifiez la fonction du controleur permettant de créer un produit pour ajouter plusieurs catégories au produit vide permettant de créer le formulaire.
- Modifiez le template de la façon suivante:
- Il faut faire une boucle pour afficher toutes les instances de catégories déjà présentes dans le produit.

```
{{ form_start(form) }}
    {{ form_row(form.nom) }}
    {{ form_row(form.description) }}
    ..... {# ici les autres champs du formulaire produit #}
    <h3>Catégories</h3>
    <ul class="categories">
        {% for categorie in form.categories %}
            <li>{{ form_row(categorie.nom) }}<BR>
                {{ form_row(categorie.couleur) }}
            </li>
        {% endfor %}
    </ul>
{{ form_end(form) }}
```

- 
- 
- Affichez le formulaire, il affiche déjà des sous-formulaires contenant les catégories que vous avez créé.
  - Nous avons obtenu un formulaire permettant de créer un produit avec des formulaires imbriqués contenant les informations de catégories.
  - Il serait intéressant de pouvoir ajouter des catégories à la volée...

- 
- Ajoutez une option dans la classe du formulaire :

```
->add("categories", CollectionType::class, [  
    'entry_type' => CategorieType::class,  
    'entry_options' => ['label' => true],  
    'allow_add' => true,  
    'by_reference' => false,  
])
```

- 
- 
- L'option `allow_add` va permettre d'ajouter un prototype dans le code HTML.
  - Ce prototype est en fait le morceau de code HTML correspondant au morceau de formulaire permettant d'ajouter une nouvelle catégorie.
  - Nous utiliserons ensuite le Javascript pour dynamiquement ajouter ce nouveau morceau de formulaire lorsqu'on clique sur un bouton.



- 
- Dans la balise `<ul class="categories">` ajoutez :
  - `data-prototype="{{ form_widget(form.categories.vars.prototype)|e('html_attr') }}"`
  - Ce qui donne :
  - `<ul class="tags" data-prototype="{{ form_widget(form.categories.vars.prototype)|e('html_attr') }}">`

- 
- Ajoutez ensuite juste après la balise <ul> un bouton :
  - `<button type="button" class="add_item_link" data-collection-holder-class="categories">Ajouter une catégorie</button>`
  - Il faut maintenant ajouter la fonction Javascript qui :
    - Récupère le contenu du prototype
    - Compte le nombre de sous-formulaire existant déjà
    - Place un index dans certains champs du prototype
    - Ajoute le prototype à la liste des sous-formulaires

```
<script type="text/javascript">
jQuery(document).ready(function() {
    // Récupérer l'entité UL qui contient la collection de catégories
    var $tagsCollectionHolder = $('ul.categories');
    //Compter le nombres de input que nous avons
    //on utilisera ce nombre comme nouvel index lors de l'insertion du
nouveau formulaire
    $tagsCollectionHolder.data('index',
$tagsCollectionHolder.find('input').length);

    $('body').on('click', '.add_item_link', function(e) {
        var $collectionHolderClass =
$(e.currentTarget).data('collectionHolderClass');
        //utiliser la fonction ajouter une nouvelle categorie (code ci-dessous)
        addFormToCollection($collectionHolderClass);
    })
})
```

```
function addFormToCollection($collectionHolderClass) {  
    //On récupère le ul qui contient la collection de catégories (version  
    généraliste)  
    var $collectionHolder = $('.' + $collectionHolderClass);  
  
    //On récupère le prototype de formulaire HTML  
    var prototype = $collectionHolder.data('prototype');  
  
    // On récupère le nouvel index  
    var index = $collectionHolder.data('index');  
  
    var newForm = prototype;  
    //La partie ci-dessous n'est nécessaire que si vous n'avez pas  
    // utilisé l'option 'label' => false dans la classe php du formulaire  
    //Remplace __name__label__ dans le formulaire pour indiquer l'index du  
    formulaire actuel  
    newForm = newForm.replace(/__name__label__/g, index);  
  
    //Remplace __name__ dans le prototype par l'index du formulaire actuel  
    newForm = newForm.replace(/__name__/g, index);  
}
```

```
// incrémente l'index pour le prochain ajout de formulaire
$collectionHolder.data('index', index + 1);

//Affiche le formulaire dans la page au sein d'une balise <li>
var $newFormLi = $('<li></li>').append(newForm);
// Ajoute le formulaire HTML à la liste
$collectionHolder.append($newFormLi)
}
});
</script>
```