**SonataAdminBundle :**

1. **Documentation :**

<https://symfony.com/doc/3.x/bundles/SonataAdminBundle/index.html>

1. **Installation :**

**composer require sonata-project/admin-bundle**

Après installer SonataAdminBundle on doit installer l’un de 4 bundles auxiliere pour l’orm dans notre cas on va installer le bundle de l’orm doctrine :

**composer require sonata-project/doctrine-orm-admin-bundle**

1. **Une fois les Bundles installés, il faut les activer dans AppKernel :**

**new** Sonata\CoreBundle\SonataCoreBundle(),

**new** Sonata\BlockBundle\SonataBlockBundle(),

**new** Knp\Bundle\MenuBundle\KnpMenuBundle(),

**new** Sonata\DoctrineORMAdminBundle\SonataDoctrineORMAdminBundle(),

**new** Sonata\AdminBundle\SonataAdminBundle(),

1. **Configurer le bundle :**

*# config/config.yml*

**sonata\_block:**

**blocks:**

*# enable the SonataAdminBundle block*

**sonata.admin.block.admin\_list:**

**contexts:** [admin]

1. **Activer le translator:**

*# config/config.yml*

translator: { fallbacks: ["%locale%"] }

1. **Definir les routes :**nnotation

**admin\_area:**

**resource:** '@SonataAdminBundle/Resources/config/routing/sonata\_admin.xml'

**prefix:** /admin

**\_sonata\_admin:**

**resource:** .

**type:** sonata\_admin

**prefix:** /admin

1. **préparer l’environement :**

$ bin/console cache:clear

bin/console cache:clear

bin/console assets:install

**sous windows :**

ouvrir le lien <http://localhost:8000/admin>

**sous linux :**

exécuter la commande suivantes :

**bin/console server:run**

ensuite ouvrir le lien  <http://localhost:8000/admin>

1. **Créer les classes Admin :**

On doit créer un répertoire nommé Admin Dans src/AppBundle/Admin

ensuite créer la classe : CategoryAdmin

<?php

**namespace** AppBundle\Admin;

**use** Sonata\AdminBundle\Admin\AbstractAdmin;

**class** CategoryAdmin **extends** AbstractAdmin

{

/\*these lines configure which fields are displayed on the edit and create actions.

The FormMapper behaves similar to the FormBuilder of the Symfony Form component;\*/

/\* c’est a dire ici declarer nos attribut de l’entité afin de créer les champ de texte comme formbuilder\*/

**protected function** configureFormFields(FormMapper $formMapper)

{

$formMapper->add('name', TextType::**class**);

}

/\*this method configures the filters, used to filter and sort the list of models \*/

/\*ici pour le filtre et le trie \*/

**protected function** configureDatagridFilters(DatagridMapper $datagridMapper)

{

$datagridMapper->add('name');

}

/\* Here you specify which fields are shown when all models are listed (the addIdentifier()method means that this field will link to the show/edit page of this particular model).\*/

/\*ici specifier l’attribut afficher dans /list \*/

**protected function** configureListFields(ListMapper $listMapper)

{

$listMapper->addIdentifier('name'); /\*ici va métre un lien pour le edit par le nom\*/

$listMapper→add('name'); /\*ici add ne créer pas de lien pour le edit\*/

}

}

toutes nos classe Admin doit étre appele comme des services tager par sonata.admin

**services:**

*# ...*

**admin.category:**

**class:** AppBundle\Admin\CategoryAdmin

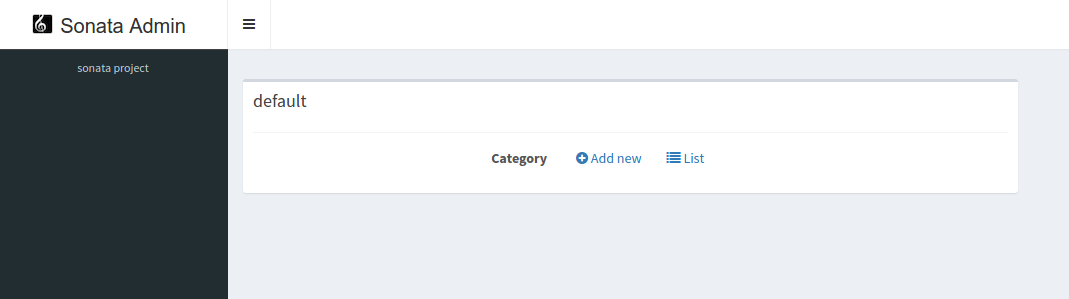
**arguments:** [~, App\Entity\Category, ~]

**tags:**

- { **name:** sonata.admin, **manager\_type:** orm, **label:** Category }

**public:** true

voir maintenet le navigateur :



Pour ajouter des champs référer une entité :

**protected function** configureFormFields(FormMapper $formMapper)

{

$formMapper

->add('title', TextType::**class**)

->add('body', TextareaType::**class**)

->add('category', EntityType::**class**, [

'class' => Category::**class**,

'choice\_label' => 'name',

]);

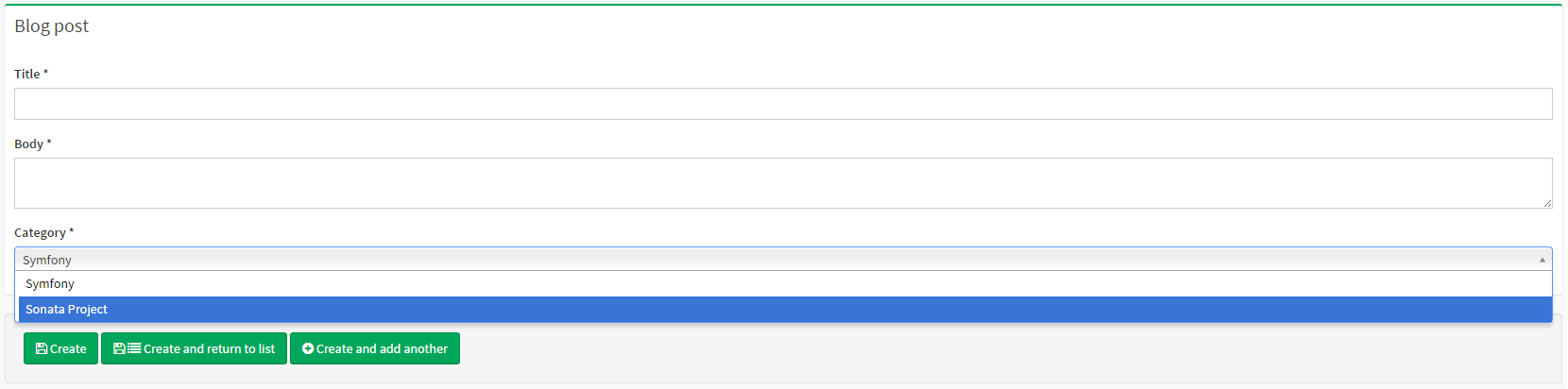
}

juste commele formbuilder ajouter ->add('category', EntityType::**class**, [

'class' => Category::**class**,

'choice\_label' => 'name',

]);

pour faciliter le développement on peut utilise ModelType au place EntityType

$formMapper

->add('title', TextType::**class**)

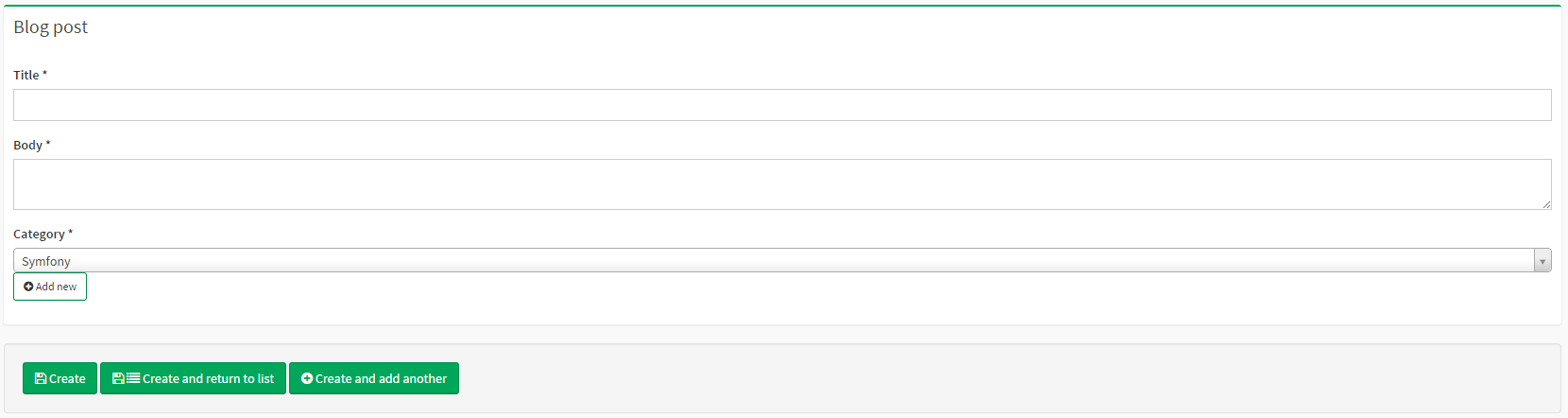
->add('body', TextareaType::**class**)

->add('category', ModelType::**class**, [

'class' => Category::**class**,

'property' => 'name',

]);



1. **Utilisation des groupes :**

Par exemple, les champs title et body peuvent appartenir au groupe Content et au champ category à un groupe de métadonnées. Pour ce faire, utilisez la méthode with ():

**protected function configureFormFields(FormMapper $formMapper)**

{

$formMapper

->with('Content', ['class' => 'col-md-9'])

->add('title', TextType::**class**)

->add('body', TextareaType::**class**)

->end()

->with('Meta data', ['class' => 'col-md-3'])

->add('category', ModelType::**class**, [

'class' => Category::**class**,

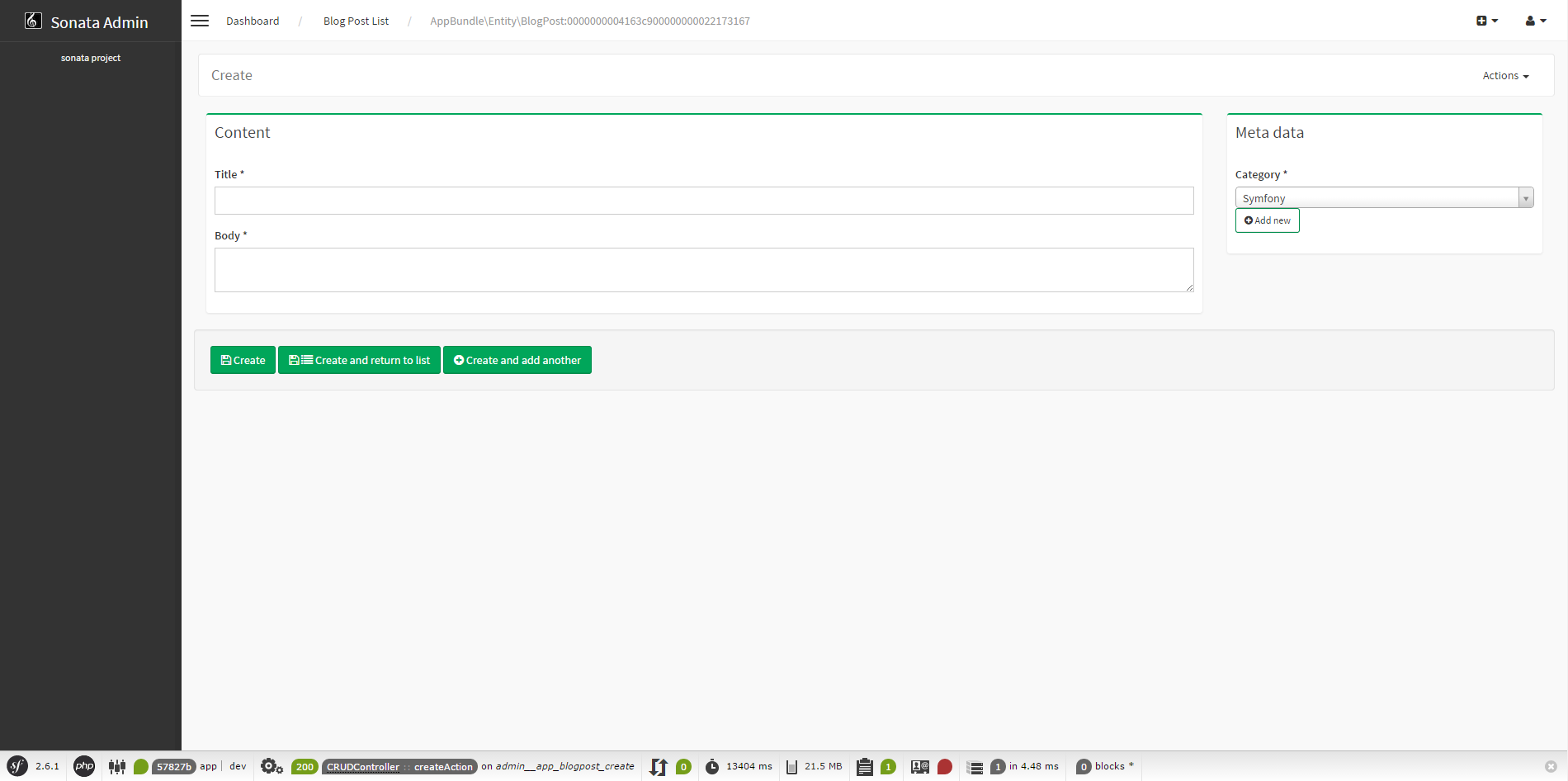
'property' => 'name',

])

->end()

;

}



1. **Utilisations des tabs :**

**protected function configureFormFields(FormMapper $formMapper)**

{

$formMapper

→tab('Post') // permet de créer une tab

->with('Content', ['class' => 'col-md-9'])

->add('title', TextType::**class**)

->add('body', TextareaType::**class**)

→end()

→end() // end du tab

->tab('Publish Options')// permet de créer une tab

->with('Meta data', ['class' => 'col-md-3'])

->add('category', ModelType::**class**, [

'class' => Category::**class**,

'property' => 'name',

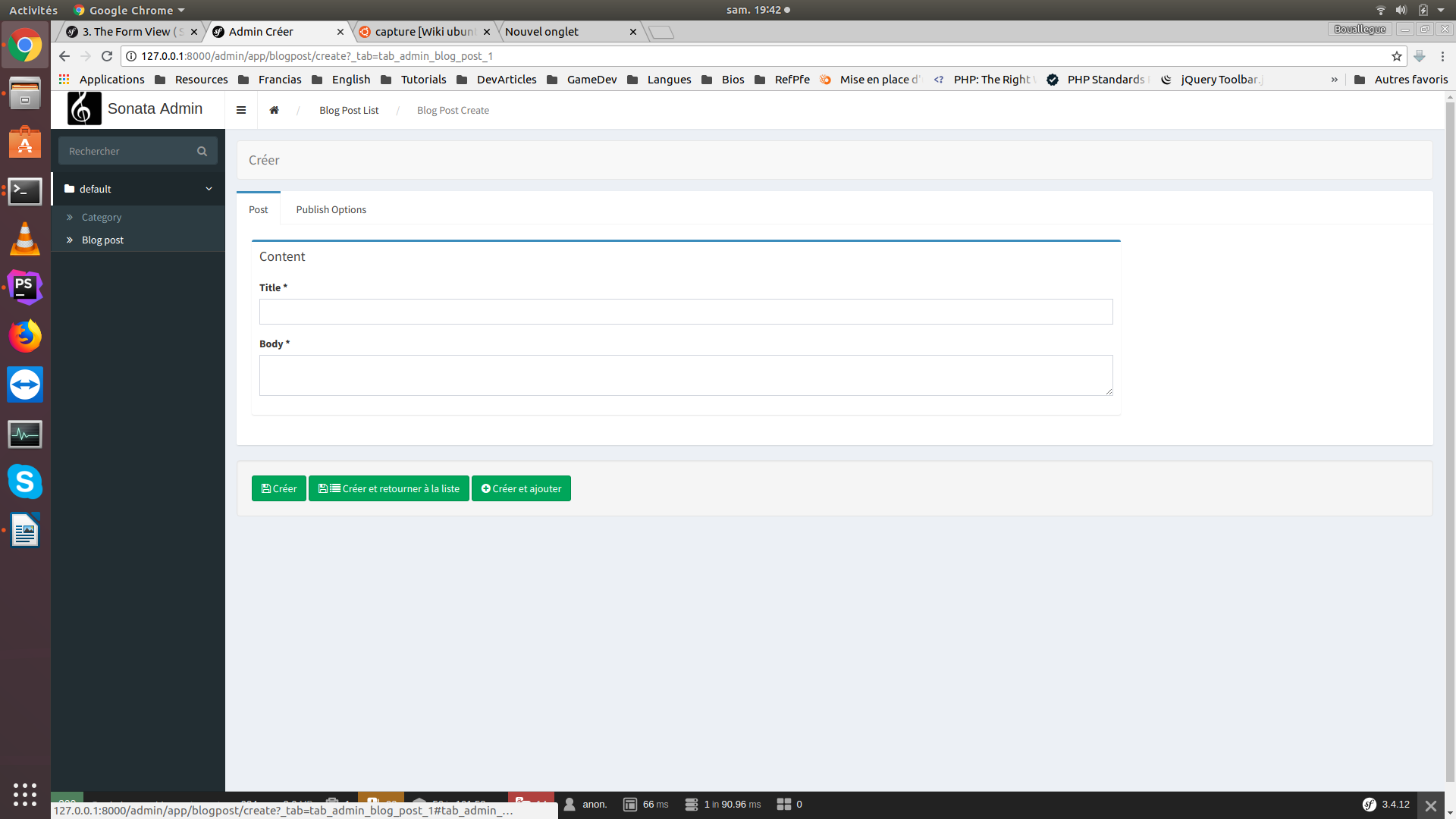
])

->end()

->end()// fin du tab

;

}

****

vous pouvez constater qu’on a deux tabs post et publish option.

1. **Pour changer la représentation du breadcrumb et de cerate view:**

**public function toString($object)**

{

**return** $object **instanceof** BlogPost

? $object->getTitle()

: 'Blog Post'; // shown in the breadcrumb on the create view

vous pouvez constater que apres l’ajout il n’y a pas de AppBundle:entity../0154545qsd5 il va etre chnager par le titre de BlogPost.

1. **Pour afficher d’autre model(entité) dans le listview(datagrid) :**

**protected function configureListFields(ListMapper $listMapper)**

{

$listMapper

->addIdentifier('title')

->add('body')

->add('draft')

->add('category.name'); // category.name permet d’appele l’entité category et le champ name

}

1. **Pour filtrer avec une autre entité (dans notre cas category):**

**protected function configureDatagridFilters(DatagridMapper $datagridMapper)**

{

$datagridMapper->add('title')

->add('body')

/\*ici le block pour ajouter une autre entité\*/

->add('category', **null**, [], EntityType::**class**, [

'class' => Category::**class**,

'choice\_label' => 'name',

]);

}