PerfectNeeds LocaleBundle

Very simple bundle that allows you to translate your entities.

Reference Repo.

https://github.com/vm5/EntityTranslationsBundle

Installation:

- composer require vm5/entity-translations-bundle
- Register bundle in AppKernel.php: `new VM5\EntityTranslationsBundle\VM5EntityTranslationsBundle`
- Copy LocaleBundle inside your ptoject in `src/PN/Bundle`
- Register bundle in AppKernel.php: `new PN\Bundle\LocaleBundle\LocaleBundle()`
- Add messages file into `app/Resources/translations` for each language called messages.{LOCALE}.php (ex. messages.ar.php)
- You must add this code in config.yml:

```
parameters:
    locale: en
    app.locales: en|ar|

doctrine:
   orm:
        # search for the "ResolveTargetEntityListener" cla
ss for an article about this
```

```
        resolve_target_entities:
            VM5\EntityTranslationsBundle\Model\Language:
```

Change the Locale in API methods

```
$this->get('vm5_entity_translations.translator')->setLocal
e('ar'); //translate entities
$this->get('translator')->setLocale('ar'); // translates m
essages
```

Routing example

```
cms:
    resource: "@CMSBundle/Controller/FrontEnd/"
    type:    annotation
    prefix: /{_locale}
    defaults:  {_locale: '%locale%' }
    requirements:
        _locale: '%app.locales%'
```

Example entities:

```
<?php

namespace PN\Bundle\CMSBundle\Entity;

```

```php
use Doctrine\ORM\Mapping as ORM;
use VM5\EntityTranslationsBundle\Model\Translatable;
use PN\Bundle\LocaleBundle\Model\LocaleTrait;

/**
 * Blogger
 *
 * @ORM\HasLifecycleCallbacks
 * @ORM\Table(name="blogger")
 * @ORM\Entity(repositoryClass="PN\Bundle\CMSBundle\Reposi
tory\BloggerRepository")
 */
class Blogger implements Translatable {

    use LocaleTrait;

    /**
     * @var int
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="title", type="string", length=255
)
     */
    protected $title;

    /**
     * @ORM\OneToMany(targetEntity="PN\Bundle\CMSBundle\En
tity\Translation\BloggerTranslation", mappedBy="translatab
le", cascade={"ALL"}, orphanRemoval=true)
     */
    protected $translations;

    /**
     * Now we tell doctrine that before we persist or upda
te we call the updatedTimestamps() function.
     *
     * @ORM\PrePersist
     * @ORM\PreUpdate
     */
    public function updatedTimestamps() {
        $this->setModified(new \DateTime(date('Y-m-d H:i:s
')));

        if ($this->getCreated() == null) {
            $this->setCreated(new \DateTime(date('Y-m-d H:
i:s')));
        }
    }

    /**
     * Constructor
     */
    public function __construct() {
        $this->translations = new \Doctrine\Common\Collect
ions\ArrayCollection();
    }

    /**
     * Get id
     *
     * @return int
     */
    public function getId() {
        return $this->id;
    }

    /**
     * Set title
     *
     * @param string $title
     *
     * @return Blogger
     */
    public function setTitle($title) {
        $this->title = $title;

        return $this;
    }

    /**
     * Get title
     *
     * @return string
     */
    public function getTitle() {
        return !$this->currentTranslation ? $this->title :
 $this->currentTranslation->getTitle();
    }

}
```

BloggerTranslation.php in `CMSBundle/Entity/Translation`

```
<?php

namespace PN\Bundle\CMSBundle\Entity\Translation;

use Doctrine\ORM\Mapping as ORM;
use VM5\EntityTranslationsBundle\Model\EditableTranslation
;
use PN\Bundle\LocaleBundle\Model\TranslationEntity;

/**
```

```
 * @ORM\Entity
 * @ORM\Table(name="blogger_translations")
 */
class BloggerTranslation extends TranslationEntity impleme
nts EditableTranslation {

    /**
     * @var string
     *
     * @ORM\Column(name="title", type="string", length=255
)
     */
    protected $title;

    /**
     * @var
     * @ORM\Id
     * @ORM\ManyToOne(targetEntity="PN\Bundle\CMSBundle\En
tity\Blogger", inversedBy="translations")
     */
    protected $translatable;

    /**
     * @var Language
     * @ORM\Id
     * @ORM\ManyToOne(targetEntity="PN\Bundle\LocaleBundle
\Entity\Language")
     */
    protected $language;

    /**
     * Set title
     *
     * @param string $title
     *
     * @return Blogger
     */
    public function setTitle($title) {
        $this->title = $title;

        return $this;
    }

    /**
     * Get title
     *
     * @return string
     */
    public function getTitle() {
        return $this->title;
    }

}
```

Then you can translate them on yourself

```
$blogger = new Blogger();

// Arabic
$arabicTranslation = new BloggerTranslation();
$arabicTranslation->setLanguage($arabicLanguage);
$arabicTranslation->setTitle('Title on arabic');
$blogger->addTranslation($arabicTranslation);

// French
$frenchTranslation = new BloggerTranslation();
$frenchTranslation->setLanguage($frenchLanguage);
$frenchTranslation->setTitle('Title on french');
$blogger->addTranslation($frenchTranslation);

$em->persist($blogger);
$em->flush();
```

Using form to easily translate entities.

Create BloggerTranslationType class

in `CMSBundle/Form/Translation/BloggerTranslationType.php`

Add all translatable columns such as BloggerTranslation.php (entity)

```php
<?php

namespace PN\Bundle\CMSBundle\Form\Translation;

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

class BloggerTranslationType  extends AbstractType {

    /**
     * {@inheritdoc}
     */
    public function buildForm(FormBuilderInterface $builde
r, array $options) {
        $builder->add('title')
        ;
    }

    /**
     * {@inheritdoc}
     */
    public function configureOptions(OptionsResolver $reso
lver) {
        $resolver->setDefaults(array(
            'data_class' => \PN\Bundle\CMSBundle\Entity\Tr
anslation\BloggerTranslation::class
```

```
        ));
    }

    /**
     * {@inheritdoc}
     */
    public function getBlockPrefix() {
        return 'pn_bundle_cmsbundle_blogger';
    }
}
```

add translations field in BloggerType.php

```
<?php

namespace PN\Bundle\CMSBundle\Form;

use Doctrine\ORM\EntityRepository;
use Symfony\Bridge\Doctrine\Form\Type\EntityType;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;
use VM5\EntityTranslationsBundle\Form\Type\TranslationsTyp
e;
use PN\Bundle\CMSBundle\Form\Translation\BloggerTranslatio
nType;
use PN\Bundle\CMSBundle\Entity\BloggerTag;
use PN\Bundle\SeoBundle\Form\SeoType;
```

```

class BloggerType extends AbstractType {

    /**
     * {@inheritdoc}
     */
    public function buildForm(FormBuilderInterface $builde
r, array $options) {
        $builder->add('title')
                ->add('publish')
                ->add('seo', SeoType::class)
                ->add('post', PostType::class)
                ->add('translations', TranslationsType::cl
ass, [
                    'entry_type' => BloggerTranslationType
::class,
                    'entry_language_options' => [
                        'en' => [
                            'required' => true,
                        ]
                    ],
                ])
        ;
    }

    /**
     * {@inheritdoc}
     */
    public function configureOptions(OptionsResolver $reso
lver) {
        $resolver->setDefaults(array(
            'data_class' => 'PN\Bundle\CMSBundle\Entity\Bl
ogger'
        ));
    }

    /**
     * {@inheritdoc}
     */
    public function getBlockPrefix() {
        return 'pn_bundle_cmsbundle_blogger';
    }

}
```

in your main form.

It’s important to include required in entry_language_options for specific locales, because validation is triggered only when language is not empty or it’s required.

Language is assumed as not empty when at least one of the fields are filled in.