NucleosProfileBundle

Dec 29, 2021

Contents

This is an extension for the basic user profile brought with the NucleosUserBundle.

The NucleosProfileBundle builds on top of this to make it quick and easy to open up a registration form and let the user to change the profile.

The following documents are available:

CHAPTER 1

Installation

1.1 Prerequisites

You need to configure the NucleosUserBundle first, check NucleosUserBundle documentation.

1.1.1 Translations

If you wish to use default texts provided in this bundle, you have to make sure you have translator enabled in your config.

```
# config/packages/framework.yaml
framework:
    translator: ~
```

For more information about translations, check Symfony documentation.

1.2 Installation

1. Download NucleosProfileBundle using composer
2. Enable the Bundle
3. Configure the NucleosProfileBundle
4. Configure your application's security.yaml
5. Import NucleosProfileBundle routing

1.2.1 Step 1: Download NucleosProfileBundle using composer

Require the bundle with composer:

```
$ composer require nucleos/profile-bundle
```

1.2.2 Step 2: Enable the bundle

Enable the bundle in the kernel:

```
// config/bundles.php
return [
    // ...
    Nucleos\ProfileBundle\NucleosProfileBundle::class => ['all' => true],
    // ...
]
```

1.2.3 Step 3: Configure the NucleosProfileBundle

Add the following configuration to your `config/packages/nucleos_profile.yaml`.

```
# config/packages/nucleos_profile.yaml
nucleos_profile:
    registration:
        confirmation:
            from_email:  "%mailer_user%"
```

1.2.4 Step 4: Configure your application's security.yaml

In addition to the NucleosUserBundle configuration, you have to define some new access rules.

Below is a minimal example of the configuration necessary to use the NucleosUserBundle in your application:

```
# config/packages/security.yaml
security:
    // ...

    access_control:
        - { path: ^/profile, role: IS_AUTHENTICATED_REMEMBERED }
```

1.2.5 Step 5: Import NucleosProfileBundle routing files

Now that you have activated and configured the bundle, all that is left to do is import the NucleosProfileBundle routing files.

By importing the routing files you will have ready made pages for things such as logging in, creating users, etc.

```
# config/routes/nucleos_profile.yaml
nucleos_profile_profile:
    resource: "@NucleosProfileBundle/Resources/config/routing/profile.php"
    prefix: /profile

nucleos_profile_change_registration:
    resource: "@NucleosProfileBundle/Resources/config/routing/registration.php"
    prefix: /register
```

CHAPTER 2

Hooking into the Controllers

The controllers packaged with the NucleosProfileBundle provide a lot of functionality that is sufficient for general use cases. But, you might find that you need to extend that functionality and add some logic that suits the specific needs of your application.

For this purpose, the controllers are dispatching events in many places in their logic. All events can be found in the constants of the `Nucleos\ProfileBundle\NucleosProfileEvents` class.

All controllers follow the same convention: they dispatch a `SUCCESS` event when the form is valid before saving the user, and a `COMPLETED` event when it is done. Thus, all `SUCCESS` events allow you to set a response if you don't want the default redirection. And all `COMPLETED` events give you access to the response before it is returned.

Controllers with a form also dispatch an `INITIALIZE` event after the entity is fetched, but before the form is created.

For instance, this listener will change the redirection after the password resetting to go to the homepage.

```php
// src/App/EventListener/ProfileEditingListener.php
namespace App\EventListener;

use Nucleos\ProfileBundle\Event\FormEvent;
use Nucleos\ProfileBundle\NucleosProfileEvents;
use Symfony\Component\EventDispatcher\EventSubscriberInterface;
use Symfony\Component\HttpFoundation\RedirectResponse;
use Symfony\Component\Routing\Generator\UrlGeneratorInterface;

/**
 * Listener responsible to change the redirection at the end of the profile editing
 */
class ProfileEditingListener implements EventSubscriberInterface
{
    private $router;

    public function __construct(UrlGeneratorInterface $router)
    {
        $this->router = $router;
    }
```

(continues on next page)

(continued from previous page)

```
    public static function getSubscribedEvents(): array
    {
        return [
            NucleosProfileEvents::PROFILE_EDIT_SUCCESS => 'onProfileEditingSuccess',
        ];
    }

    public function onProfileEditingSuccess(FormEvent $event): void
    {
        $url = $this->router->generate('homepage');

        $event->setResponse(new RedirectResponse($url));
    }
}
```

You can then register this listener:

```
# config/services.yaml
services:
    app.password_resetting:
        class: App\EventListener\ProfileEditingListener
        arguments: ['@router']
        tags:
            - { name: kernel.event_subscriber }
```

CHAPTER 3

Sending E-Mails

The NucleosProfileBundle has built-in support for sending emails in two different instances.

3.1 Registration Confirmation

The first is when a new user registers and the bundle is configured to require email confirmation before the user registration is complete. The email that is sent to the new user contains a link that, when visited, will verify the registration and enable the user account.

Requiring email confirmation for a new account is turned off by default. To enable it, update your configuration as follows:

```yaml
# config/packages/nucleos_profile.yaml
nucleos_profile:
    registration:
        confirmation:
            enabled: true
```

3.2 Default Mailer Implementations

The bundle comes with three mailer implementations. They are listed below by service id:

- `nucleos_profile.mailer.simple` is the default implementation, and uses symfony mailer to send emails.
- `nucleos_profile.mailer.noop` is a mailer implementation which performs no operation, so no emails are sent.

3.3 Configuring the Sender Email Address

The NucleosProfileBundle default mailer allows you to configure the sender email address of the emails sent out by the bundle.

To configure the sender email address for registration emails sent out by the bundle, update your `nucleos_profile` config as follows:

```yaml
# config/packages/nucleos_profile.yaml
nucleos_profile:
    # ...
    registration:
        confirmation:
            from_email:   resetting@example.com
```

3.4 Using A Custom Mailer

The default mailer service used by NucleosProfileBundle relies on the symfony mailer library to send mail. If you would like to use a different library to send emails or change the content of the email you may do so by defining your own service.

First you must create a new class which implements `Nucleos\UserBundle\Mailer\MailerInterface` which is listed below:

```php
namespace Nucleos\UserBundle\Mailer;

use Nucleos\UserBundle\Model\UserInterface;

interface MailerInterface
{

    /**
     * Send an email to a user to confirm the password reset
     *
     * @param UserInterface $user
     */
    function sendConfirmationEmailMessage(UserInterface $user): void;
}
```

After you have implemented your custom mailer class and defined it as a service, you must update your bundle configuration so that NucleosProfileBundle will use it. Set the `mailer` configuration parameter under the `service` section. An example is listed below.

```yaml
# config/packages/nucleos_profile.yaml
nucleos_profile:
    # ...
    service:
        mailer: app.custom_nucleos_profile_mailer
```

CHAPTER 4

Customize Profile

If the user is allowed to change some date, you can modify the profile form. It is possible to add, remove or modify form fields provided by NucleosProfileBundle.

4.1 How to customize a Form

If you want to modify the Profile form in your project there are a few steps that you should take. For example if you would like to add checkbox for accepting terms and conditions you will have to follow these steps:

1. Extend `Nucleos\UserBundle\Model\User\User`

```php
namespace App\Model;

use Nucleos\UserBundle\Model\User as BaseUser;

class User extends BaseUser
{
    protected ?bool $termsAccepted = false;

    public function gettermsAccepted(): ?bool
    {
        return $this->termsAccepted;
    }

    public function setTermsAccepted(?bool $termsAccepted): void
    {
        $this->termsAccepted = $termsAccepted;
    }
}
```

2. Use Symfony Form Extensions to add fields. You can use builder to remove or modify fields as well.

```php
namespace App\Form\Type;

use Symfony\Component\Form\AbstractTypeExtension;
use Symfony\Component\Form\Extension\Core\Type\CheckboxType;
use Symfony\Component\Form\FormBuilderInterface;

class ProfileFormType extends AbstractTypeExtension
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder->add('termsAccepted', CheckboxType::class);
    }

    public static function getExtendedTypes(): iterable
    {
        return ['Nucleos\ProfileBundle\Form\Type\ProfileFormType'];
    }
}
```

CHAPTER 5

Customize Registration

Your business needs might require changes in registration form. It is possible to add, remove or modify form fields provided by NucleosProfileBundle.

5.1 How to customize a Form

If you want to modify the Registration form in your project there are a few steps that you should take. For example if you would like to add checkbox for accepting terms and conditions you will have to follow these steps:

1. Extend `Nucleos\UserBundle\Model\User\User`

```
namespace App\Model;

use Nucleos\UserBundle\Model\User as BaseUser;

class User extends BaseUser
{
    protected ?bool $termsAccepted = false;

    public function gettermsAccepted(): ?bool
    {
        return $this->termsAccepted;
    }

    public function setTermsAccepted(?bool $termsAccepted): void
    {
        $this->termsAccepted = $termsAccepted;
    }
}
```

2. Use Symfony Form Extensions to add fields. You can use builder to remove or modify fields as well.

```php
namespace App\Form\Type;

use Symfony\Component\Form\AbstractTypeExtension;
use Symfony\Component\Form\Extension\Core\Type\CheckboxType;
use Symfony\Component\Form\FormBuilderInterface;

class RegistrationFormType extends AbstractTypeExtension
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder->add('termsAccepted', CheckboxType::class);
    }

    public static function getExtendedTypes(): iterable
    {
        return ['Nucleos\ProfileBundle\Form\Type\RegistrationFormType'];
    }
}
```

CHAPTER 6

Configuration reference

All available configuration options are listed below with their default values.

```yaml
nucleos_profile:
    registration:
        confirmation:
            from_email:          ~ # Required
            enabled:             false # change to true for required email confirmation
    service:
        mailer:                  nucleos_profile.mailer.simple
```