Contenido

[Configure the parameters of database 2](#_Toc474709865)

[Create the database 3](#_Toc474709866)

[Create the entity 3](#_Toc474709867)

[Update the database. 3](#_Toc474709868)

[Creating relations 3](#_Toc474709869)

[Security 5](#_Toc474709870)

[Create the entity User 5](#_Toc474709871)

[THE ENTITY ROLL 14](#_Toc474709872)

[Getters and setters. Update database 18](#_Toc474709873)

[Config/security.yml 18](#_Toc474709874)

[The controller security 20](#_Toc474709875)

[The view of security 21](#_Toc474709876)

[Template permission 24](#_Toc474709877)

[Controller permission 24](#_Toc474709878)

[Generating a CRUD Controller Based on a Doctrine Entity. This case the user 26](#_Toc474709879)

[User formtype 31](#_Toc474709880)

[User new twig 34](#_Toc474709881)

[User Edit 35](#_Toc474709882)

[User index 38](#_Toc474709883)

[Validation 41](#_Toc474709884)

[The form after validation 42](#_Toc474709885)

[The form twig new 46](#_Toc474709886)

[The form edit twig 47](#_Toc474709887)

[The form index 49](#_Toc474709888)

[Controller that required ROLE\_ADMIN 52](#_Toc474709889)

## Configure the parameters of database

First the file that you need to configure is parameters.yml that is located in the config/folder/parameters.yml

## Create the database

After of create of configured the parameters open cmd and write

php bin/console doctrine:database:create

to create the database

## Create the entity

Entity is like a table that can interact with the database. Use this command to create

php bin/console doctrine:generate:entity

## Update the database.

Once you have created the entity write this command on cmd

php bin/console doctrine:schema:update --force

## Creating relations

One product can have one category but one category could be of many products

// src/AppBundle/Entity/Product.php

// ...

class Product

{

// ...

/\*\*

\* @ORM\ManyToOne(targetEntity="Category", inversedBy="products")

\* @ORM\JoinColumn(name="category\_id", referencedColumnName="id")

\*/

private $category;

}

And the category

// src/AppBundle/Entity/Category.php

// ...

use Doctrine\Common\Collections\ArrayCollection;

class Category

{

// ...

/\*\*

\* @ORM\OneToMany(targetEntity="Product", mappedBy="category")

\*/

private $products;

public function \_\_construct()

{

$this->products = new ArrayCollection();

}

}

Now that you've added new properties to both the Product and Category classes, tell Doctrine to generate the missing getter and setter methods for you:

php bin/console doctrine:generate:entities AppBundle

finally put this command

php bin/console doctrine:schema:update –force

# Security

## Create the entity User

<?php

namespace AppBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

use Symfony\Component\Security\Core\User\UserInterface;

use Symfony\Component\Validator\Constraints as Assert;

use Symfony\Bridge\Doctrine\Validator\Constraints\UniqueEntity;

/\*\*

\* @ORM\Table(name="app\_users")

\* @UniqueEntity(

\* fields={"email"},

\* message="Ya existe un usuario con este correo."

\* )

\* @ORM\Entity(repositoryClass="AppBundle\Repository\UserRepository")

\*/

class User implements UserInterface, \Serializable

{

/\*\*

\* @ORM\Column(type="integer")

\* @ORM\Id

\* @ORM\GeneratedValue(strategy="AUTO")

\*/

private $id;

/\*\*

\* @ORM\Column(type="string", length=25, unique=true)

\*/

private $username;

/\*\*

\* @ORM\Column(type="string", length=64)

\* @Assert\NotBlank()

\*/

private $password;

/\*\*

\* @ORM\Column(type="string", length=60, unique=true)

\* @Assert\NotBlank()

\* @Assert\Email(

\* message = "El email '{{ value }}' No es un email valido.",

\* checkMX = true

\* )

\*/

private $email;

/\*\*

\* @ORM\Column(name="is\_active", type="boolean")

\*/

private $isActive;

/\*\*

\* @ORM\ManyToOne(targetEntity="Roll", inversedBy="users")

\* @ORM\JoinColumn(name="idroll", referencedColumnName="id")

\*/

private $roll;

public function \_\_construct()

{

$this->isActive = true;

// may not be needed, see section on salt below

// $this->salt = md5(uniqid(null, true));

}

public function getUsername()

{

return $this->email;

}

public function getSalt()

{

// you \*may\* need a real salt depending on your encoder

// see section on salt below

return null;

}

public function getPassword()

{

return $this->password;

}

public function getRoles()

{

return array($this->roll->getDescription());

}

public function eraseCredentials()

{

}

/\*\* @see \Serializable::serialize() \*/

public function serialize()

{

return serialize(array(

$this->id,

$this->username,

$this->password,

// see section on salt below

// $this->salt,

));

}

/\*\* @see \Serializable::unserialize() \*/

public function unserialize($serialized)

{

list (

$this->id,

$this->username,

$this->password,

// see section on salt below

// $this->salt

) = unserialize($serialized);

}

/\*\*

\* Get id

\*

\* @return integer

\*/

public function getId()

{

return $this->id;

}

/\*\*

\* Set username

\*

\* @param string $username

\*

\* @return User

\*/

public function setUsername($username)

{

$this->username = $username;

return $this;

}

/\*\*

\* Set password

\*

\* @param string $password

\*

\* @return User

\*/

public function setPassword($password)

{

$this->password = $password;

return $this;

}

/\*\*

\* Set email

\*

\* @param string $email

\*

\* @return User

\*/

public function setEmail($email)

{

$this->email = $email;

return $this;

}

/\*\*

\* Get email

\*

\* @return string

\*/

public function getEmail()

{

return $this->email;

}

/\*\*

\* Set isActive

\*

\* @param boolean $isActive

\*

\* @return User

\*/

public function setIsActive($isActive)

{

$this->isActive = $isActive;

return $this;

}

/\*\*

\* Get isActive

\*

\* @return boolean

\*/

public function getIsActive()

{

return $this->isActive;

}

/\*\*

\* Set roll

\*

\* @param \AppBundle\Entity\Roll $roll

\*

\* @return User

\*/

public function setRoll(\AppBundle\Entity\Roll $roll = null)

{

$this->roll = $roll;

return $this;

}

/\*\*

\* Get roll

\*

\* @return \AppBundle\Entity\Roll

\*/

public function getRoll()

{

return $this->roll;

}

}

## THE ENTITY ROLL

<?php

namespace AppBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/\*\*

\* Roll

\*

\* @ORM\Table(name="roll")

\* @ORM\Entity(repositoryClass="AppBundle\Repository\RollRepository")

\*/

class Roll

{

/\*\*

\* @ORM\OneToMany(targetEntity="User", mappedBy="roll")

\*/

private $users;

public function \_\_construct()

{

$this->users = new ArrayCollection();

}

/\*\*

\* @var int

\*

\* @ORM\Column(name="id", type="integer")

\* @ORM\Id

\* @ORM\GeneratedValue(strategy="AUTO")

\*/

private $id;

/\*\*

\* @var string

\*

\* @ORM\Column(name="description", type="string", length=60, unique=true)

\*/

private $description;

/\*\*

\* Get id

\*

\* @return int

\*/

public function getId()

{

return $this->id;

}

/\*\*

\* Set description

\*

\* @param string $description

\*

\* @return Roll

\*/

public function setDescription($description)

{

$this->description = $description;

return $this;

}

/\*\*

\* Get description

\*

\* @return string

\*/

public function getDescription()

{

return $this->description;

}

/\*\*

\* Add user

\*

\* @param \AppBundle\Entity\User $user

\*

\* @return Roll

\*/

public function addUser(\AppBundle\Entity\User $user)

{

$this->users[] = $user;

return $this;

}

/\*\*

\* Remove user

\*

\* @param \AppBundle\Entity\User $user

\*/

public function removeUser(\AppBundle\Entity\User $user)

{

$this->users->removeElement($user);

}

/\*\*

\* Get users

\*

\* @return \Doctrine\Common\Collections\Collection

\*/

public function getUsers()

{

return $this->users;

}

}

## Getters and setters. Update database

php bin/console doctrine:generate:entities AppBundle/Entity/User

php bin/console doctrine:generate:entities AppBundle/Entity/Roll

php bin/console doctrine:schema:update --force

## Config/security.yml

# To get started with security, check out the documentation:

# http://symfony.com/doc/current/security.html

security:

encoders:

AppBundle\Entity\User:

algorithm: bcrypt

# http://symfony.com/doc/current/security.html#b-configuring-how-users-are-loaded

providers:

our\_db\_provider:

entity:

class: AppBundle:User

property: email

# if you're using multiple entity managers

# manager\_name: customer

firewalls:

main:

anonymous: ~

form\_login:

login\_path: login

check\_path: login

provider: our\_db\_provider

logout:

path: /logout

target: /

# disables authentication for assets and the profiler, adapt it according to your needs

dev:

pattern: ^/(\_(profiler|wdt)|css|images|js)/

security: false

main:

anonymous: ~

# activate different ways to authenticate

# http\_basic: ~

# http://symfony.com/doc/current/security.html#a-configuring-how-your-users-will-authenticate

# form\_login: ~

# http://symfony.com/doc/current/cookbook/security/form\_login\_setup.html

## The controller security

<?php

namespace AppBundle\Controller;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;

use Symfony\Component\HttpFoundation\Response;

use Symfony\Component\HttpFoundation\Request;

class SecurityController extends Controller

{

/\*\*

\* @Route("/login", name="login")

\*/

public function loginAction(Request $request)

{

$authenticationUtils = $this->get('security.authentication\_utils');

// get the login error if there is one

$error = $authenticationUtils->getLastAuthenticationError();

// last username entered by the user

$lastUsername = $authenticationUtils->getLastUsername();

return $this->render('security/login.html.twig', array(

'last\_username' => $lastUsername,

'error' => $error,

));

}

}

## The view of security

{% extends 'base.html.twig' %}

{% block body %}

<div style="margin-top:10%"></div>

<div class="container">

<div class="row" id="formulario">

<div class="col-md-3"></div>

<div class="col-md-6">

{% if error %}

<div class="alert alert-danger fade in">

{{ error.messageKey|trans(error.messageData, 'security') }}

</div>

{% endif %}

<form class="form-horizontal" action="{{ path('login') }}" method="post">

<div class="form-group">

<label for="username">Correo electronico:</label>

<input class="form-control" placeholder="Correo electronico" type="text" id="username" name="\_username" value="{{ last\_username }}" />

</div>

<div class="form-group">

<label for="password">Clave:</label>

<input class="form-control" placeholder="Clave" type="password" id="password" name="\_password" />

</div>

<input type="hidden" name="\_target\_path" value="usuario\_index" />

<div class="form-group">

<button class="btn btn-primary" type="submit">Ingresar</button>

</div>

</form>

</div>

<div class="col-md-3"></div>

</div>

</div>

<div style="margin-top:10%"></div>

{% endblock %}

{% block javascripts %}

{{ parent() }}

<script>

$("#formulario li").attr("class","alert alert-danger")

</script>

{% endblock %}

## Template permission

if the user is admin

{% if is\_granted('ROLE\_ADMIN') %}

…..

{% endif %}

If the user is aunthenticated

{% if is\_granted('IS\_AUTHENTICATED\_FULLY') %}

<li><a href="{{ path('logout') }}">Cerrar sesion</a></li>

{% endif %}

If the user is no aunthenticated

{% if not is\_granted('IS\_AUTHENTICATED\_FULLY') %}

<li><a href="{{ path('login') }}">Iniciar sesion</a></li>

{% endif %}

## Controller permission

<?php

namespace AppBundle\Controller;

use AppBundle\Entity\User;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\Method;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;

use Symfony\Component\HttpFoundation\Request;

use Symfony\Component\HttpFoundation\Session\Session;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\Security;

/\*\*

\* User controller.

\* @Security("has\_role('ROLE\_ADMIN')")

\* @Route("usuario")

\*/

class UserController extends Controller

{

## Generating a CRUD Controller Based on a Doctrine Entity. This case the user

By default, the command is run in the interactive mode and asks questions to determine the entity name, the route prefix or whether or not to generate write actions:

$ php app/console generate:doctrine:crud

<?php

namespace AppBundle\Controller;

use AppBundle\Entity\User;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\Method;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;

use Symfony\Component\HttpFoundation\Request;

use Symfony\Component\HttpFoundation\Session\Session;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\Security;

/\*\*

\* User controller.

\* @Security("has\_role('ROLE\_ADMIN')")

\* @Route("usuario")

\*/

class UserController extends Controller

{

/\*\*

\* Lists all user entities.

\*

\* @Route("/", name="usuario\_index")

\* @Method("GET")

\*/

public function indexAction()

{

$em = $this->getDoctrine()->getManager();

$users = $em->getRepository('AppBundle:User')->findAll();

return $this->render('user/index.html.twig', array(

'users' => $users,

));

}

/\*\*

\* Creates a new user entity.

\*

\* @Route("/nuevo", name="usuario\_new")

\* @Method({"GET", "POST"})

\*/

public function newAction(Request $request)

{

$user = new User();

$form = $this->createForm('AppBundle\Form\UserType', $user);

$form->handleRequest($request);

$usuario = "a";

if ($form->isSubmitted() && $form->isValid()) {

$user->setUsername($user->getEmail());

$encoder = $this->container->get('security.password\_encoder');

$encoded = $encoder->encodePassword($user, $user->getPassword());

$user->setPassword($encoded);

$em = $this->getDoctrine()->getManager();

$em->persist($user);

$em->flush($user);

$this->addFlash(

'mensaje',

'Usuario creado correctamente'

);

return $this->redirectToRoute('usuario\_index');

}

return $this->render('user/new.html.twig', array(

'user' => $user,

'form' => $form->createView(),

'usuario'=>$usuario,

));

}

/\*\*

\* Displays a form to edit an existing user entity.

\*

\* @Route("/{id}/editar", name="usuario\_edit")

\* @Method({"GET", "POST"})

\*/

public function editAction(Request $request, User $user)

{

$deleteForm = $this->createDeleteForm($user);

$editForm = $this->createForm('AppBundle\Form\UserType', $user);

$editForm->handleRequest($request);

if ($editForm->isSubmitted() && $editForm->isValid()) {

$encoder = $this->container->get('security.password\_encoder');

$encoded = $encoder->encodePassword($user, $user->getPassword());

$user->setPassword($encoded);

$this->getDoctrine()->getManager()->flush();

$this->addFlash(

'mensaje',

'Usuario actualizado'

);

return $this->redirectToRoute('usuario\_edit', array('id' => $user->getId()));

}

return $this->render('user/edit.html.twig', array(

'user' => $user,

'edit\_form' => $editForm->createView(),

'delete\_form' => $deleteForm->createView(),

));

}

/\*\*

\* Deletes a user entity.

\*

\* @Route("/{id}", name="usuario\_delete")

\* @Method("DELETE")

\*/

public function deleteAction(Request $request, User $user)

{

$form = $this->createDeleteForm($user);

$form->handleRequest($request);

if ($form->isSubmitted() && $form->isValid()) {

$em = $this->getDoctrine()->getManager();

$em->remove($user);

$em->flush($user);

}

return $this->redirectToRoute('usuario\_index');

}

/\*\*

\* Creates a form to delete a user entity.

\*

\* @param User $user The user entity

\*

\* @return \Symfony\Component\Form\Form The form

\*/

private function createDeleteForm(User $user)

{

return $this->createFormBuilder()

->setAction($this->generateUrl('usuario\_delete', array('id' => $user->getId())))

->setMethod('DELETE')

->getForm()

;

}

}

## User formtype

<?php

namespace AppBundle\Form;

use Symfony\Component\Form\AbstractType;

use Symfony\Component\Form\FormBuilderInterface;

use Symfony\Component\OptionsResolver\OptionsResolver;

use Symfony\Component\Form\Extension\Core\Type\EmailType;

use Symfony\Component\Form\Extension\Core\Type\PasswordType;

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

use Symfony\Bridge\Doctrine\Form\Type\EntityType;

class UserType extends AbstractType

{

/\*\*

\* {@inheritdoc}

\*/

public function buildForm(FormBuilderInterface $builder, array $options)

{

$builder

->add('email', EmailType::class, array('attr' => array('class' => 'form-control'),'label'=>'E-mail'))

->add('password', PasswordType::class, array('attr' => array('class' => 'form-control'),'label'=>'Clave'))

->add('isActive',CheckboxType::class,array('label'=>'Activo'))

->add('roll', EntityType::class, array(

'attr' => array('class' => 'form-control'),

'label'=>'Rol del usuario',

// query choices from this entity

'class' => 'AppBundle:Roll',

'placeholder' => 'Seleccione el rol de usuario...',

// use the User.username property as the visible option string

'choice\_label' => 'description',

));

}

/\*\*

\* {@inheritdoc}

\*/

public function configureOptions(OptionsResolver $resolver)

{

$resolver->setDefaults(array(

'data\_class' => 'AppBundle\Entity\User'

));

}

/\*\*

\* {@inheritdoc}

\*/

public function getBlockPrefix()

{

return 'appbundle\_user';

}

}

## User new twig

{% extends 'base.html.twig' %}

{% block body %}

<h1>Crear usuario</h1>

<div class="container">

<div class="row" id="formulario">

<div class="col-md-3"></div>

<div class="col-md-6">

{{ form\_start(form) }}

{{ form\_widget(form) }}

<div class="form-group">

<input class="btn btn-primary" type="submit" value="Guardar cambios" />

</div>

{{ form\_end(form) }}

<div class="form-group">

<a class="btn btn-warning" href="{{ path('usuario\_index') }}">Regresar atras</a>

</div>

</div>

<div class="col-md-3"></div>

</div>

</div>

<div style="margin-bottom:10%;"></div>

{% endblock %}

{% block javascripts %}

{{ parent() }}

<script>

$("#formulario li").attr("class","alert alert-danger")

</script>

{% endblock %}

## User Edit

{% extends 'base.html.twig' %}

{% block body %}

<h1>Editar usuario</h1>

<div class="container">

<div class="row" id="formulario">

<div class="col-md-3"></div>

<div class="col-md-6">

{% for flashMessage in app.session.flashbag.get('mensaje') %}

<div class="alert alert-success">

{{ flashMessage }}

</div>

{% endfor %}

{{ form\_start(edit\_form) }}

{{ form\_widget(edit\_form) }}

<div class="form-group" style="margin-top:20px">

<input class="btn btn-warning col-md-12" type="submit" value="Guardar cambios" />

</div>

{{ form\_end(edit\_form) }}

<div class="form-group">

<a class="col-md-12 btn btn-info" href="{{ path('usuario\_index') }}">Regresar atras</a>

</div>

<div class="form-group">

{{ form\_start(delete\_form, {'attr': {'onclick': 'return confirm("Esta seguro de borrar el registro?")'}} ) }}

<input class="col-md-12 btn btn-danger" type="submit" value="Borrar">

{{ form\_end(delete\_form) }}

</div>

</div>

<div class="col-md-3"></div>

</div>

</div>

<div style="margin-bottom:13%"></div>

{% endblock %}

{% block javascripts %}

{{ parent() }}

<script>

$("#formulario li").attr("class","alert alert-danger");

$("#appbundle\_user\_password").attr("required","as");

</script>

{% endblock %}

## User index

{% extends 'base.html.twig' %}

{% block body %}

<div class="container">

<div style="margin-bottom:30px"></div>

<div class="row">

<div class="col-md-12">

<a class="btn btn-success" href="{{ path('usuario\_new') }}">Crear un nuevo usuario</a>

</div>

</div>

<div class="row">

<div class="col-md-12">

<h2> Bienvenido {{ app.user.username }}</h2>

</div>

</div>

{% for flashMessage in app.session.flashbag.get('mensaje') %}

<div class="alert alert-success">

{{ flashMessage }}

</div>

{% endfor %}

<h1>Lista de usuarios</h1>

<div class="table-responsive">

<table class="table table-hover table-bordered">

<thead>

<tr>

<th>Id</th>

<th>Email</th>

<th>Activo</th>

<th>Acciones</th>

</tr>

</thead>

<tbody>

{% for user in users %}

<tr>

<td>{{ user.id }}</td>

<td>{{ user.email }}</td>

<td>{% if user.isActive %}Si{% else %}No{% endif %}</td>

<td>

<div class="row">

<div class="col-md-12">

<a class="btn btn-warning" href="{{ path('usuario\_edit', { 'id': user.id }) }}">Editar</a>

</div>

</div>

</td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

</div>

<div style="margin-bottom:20%"></div>

{% endblock %}

# Validation

Example

namespace AppBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

use Symfony\Component\Validator\Constraints as Assert;

use Symfony\Bridge\Doctrine\Validator\Constraints\UniqueEntity;

/\*\*

\* Customer

\*

\* @ORM\Table(name="customer")

\* @ORM\Entity(repositoryClass="AppBundle\Repository\CustomerRepository")

\* @UniqueEntity("cedula",message="Esta cedula ya existe")

\* @UniqueEntity("email",message="Este email ya existe")

\* @UniqueEntity("numeroRegistro",message="Este numero de registro ya existe")

\*/

class Customer

{

/\*\*

\* @var string

\*

\* @ORM\Column(name="email", type="string", length=100, unique=true)

\* @Assert\NotBlank(message = "El email es requerido")

\* @Assert\Email(

\* message = "El email '{{ value }}' No es un email valido.",

\* checkMX = true

\* )

\*/

private $email;

con la fecha

/\*\*

\* @var \DateTime

\*

\* @ORM\Column(name="fecha\_curso", type="datetimetz")

\* @Assert\Date()

\* @Assert\NotBlank(message = "La fecha del curso es requerida")

\*/

private $fechaCurso;

## The form after validation

<?php

namespace AppBundle\Form;

use Symfony\Component\Form\AbstractType;

use Symfony\Component\Form\FormBuilderInterface;

use Symfony\Component\OptionsResolver\OptionsResolver;

use Symfony\Component\Form\Extension\Core\Type\EmailType;

use Symfony\Component\Form\Extension\Core\Type\TextType;

use Symfony\Component\Form\Extension\Core\Type\ChoiceType;

use Symfony\Bridge\Doctrine\Form\Type\EntityType;

use Symfony\Component\Form\Extension\Core\Type\DateType;

class CustomerType extends AbstractType

{

/\*\*

\* {@inheritdoc}

\*/

public function buildForm(FormBuilderInterface $builder, array $options)

{

$builder

->add('nombre', TextType::class, array('attr' => array('class' => 'form-control'),'label'=>'Nombre'))

->add('apellido', TextType::class, array('attr' => array('class' => 'form-control'),'label'=>'Apellido'))

->add('cedula', TextType::class, array('attr' => array('class' => 'form-control'),'label'=>'Cedula'))

->add('celular', TextType::class, array('attr' => array('class' => 'form-control'),'label'=>'Celular'))

->add('email', EmailType::class, array('attr' => array('class' => 'form-control'),'label'=>'E-mail'))

->add('numeroRegistro', TextType::class, array('attr' => array('class' => 'form-control'),'label'=>'NumeroRegistro'))

->add('descargarCertificado', ChoiceType::class, array(

'attr' => array('class' => 'form-control'),

'choices' => array(

'El cliente puede descargar el certificado' => '',

'Si' => 1,

'No' => 0,

),

))

->add('fecha\_curso', DateType::class, array(

'widget' => 'single\_text',

// do not render as type="date", to avoid HTML5 date pickers

'html5' => false,

// add a class that can be selected in JavaScript

'attr' => ['class' => 'form-control datepicker'],

))

->add('nivel\_certificado', EntityType::class, array(

'attr' => array('class' => 'form-control'),

'label'=>'Nivel del certificado',

// query choices from this entity

'class' => 'AppBundle:Nivel\_certificado',

'placeholder' => 'Seleccione el nivel de certificado...',

// use the User.username property as the visible option string

'choice\_label' => 'descripcion',

));

;

}

/\*\*

\* {@inheritdoc}

\*/

public function configureOptions(OptionsResolver $resolver)

{

$resolver->setDefaults(array(

'data\_class' => 'AppBundle\Entity\Customer'

));

}

/\*\*

\* {@inheritdoc}

\*/

public function getBlockPrefix()

{

return 'appbundle\_customer';

}

}

## The form twig new

{% extends 'base.html.twig' %}

{% block body %}

<h1>Customer creation</h1>

<div class="container">

<div class="row" id="formulario">

<div class="col-md-3"></div>

<div class="col-md-6">

{{ form\_start(form) }}

{{ form\_widget(form) }}

<div class="form-group">

<input class="btn btn-primary" type="submit" value="Guardar cambios" />

</div>

{{ form\_end(form) }}

<div class="form-group">

<a class="btn btn-warning" href="{{ path('cliente\_index') }}">Regresar atras</a>

</div>

</div>

<div class="col-md-3"></div>

</div>

</div>

<div style="margin-bottom:10%;"></div>

{% endblock %}

{% block javascripts %}

{{ parent() }}

<script>

$("#formulario li").attr("class","alert alert-danger")

</script>

{% endblock %}

## The form edit twig

{% extends 'base.html.twig' %}

{% block body %}

<h1>Customer edit</h1>

<div class="container">

<div class="row" id="formulario">

<div class="col-md-3"></div>

<div class="col-md-6">

{% for flashMessage in app.session.flashbag.get('mensaje') %}

<div class="alert alert-success">

{{ flashMessage }}

</div>

{% endfor %}

{{ form\_start(edit\_form) }}

{{ form\_widget(edit\_form) }}

<div class="form-group" style="margin-top:20px">

<input class="btn btn-warning col-md-12" type="submit" value="Guardar cambios" />

</div>

{{ form\_end(edit\_form) }}

<div class="form-group">

<a class="col-md-12 btn btn-info" href="{{ path('cliente\_index') }}">Regresar atras</a>

</div>

<div class="form-group">

{{ form\_start(delete\_form, {'attr': {'onclick': 'return confirm("Esta seguro de borrar el registro?")'}} ) }}

<input class="col-md-12 btn btn-danger" type="submit" value="Borrar">

{{ form\_end(delete\_form) }}

</div>

</div>

<div class="col-md-3"></div>

</div>

</div>

<div style="margin-bottom:13%"></div>

{% endblock %}

{% block javascripts %}

{{ parent() }}

<script>

$("#formulario li").attr("class","alert alert-danger");

</script>

{% endblock %}

## The form index

{% extends 'base.html.twig' %}

{% block body %}

<div class="container">

<div style="margin-bottom:30px"></div>

<div class="row">

<div class="col-md-12">

<a class="btn btn-success" href="{{ path('cliente\_new') }}">Crear un nuevo cliente</a>

</div>

</div>

{% for flashMessage in app.session.flashbag.get('mensaje') %}

<div class="alert alert-success">

{{ flashMessage }}

</div>

{% endfor %}

<h1>Lista de clientes</h1>

<div class="table-responsive">

<table class="table table-hover table-bordered">

<thead>

<tr>

<th>Nombre</th>

<th>Apellido</th>

<th>Cedula</th>

<th>Celular</th>

<th>Email</th>

<th>Numeroregistro</th>

<th>Fechacurso</th>

<th>Descargarcertificado</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

{% for customer in customers %}

<tr>

<td>{{ customer.nombre }}</td>

<td>{{ customer.apellido }}</td>

<td>{{ customer.cedula }}</td>

<td>{{ customer.celular }}</td>

<td>{{ customer.email }}</td>

<td>{{ customer.numeroRegistro }}</td>

<td>{{ customer.fechaCurso|date("m/d/Y") }}</td>

<td>{% if customer.descargarCertificado %}Si{% else %}No{% endif %}</td>

<td>

<a class='btn btn-warning' href="{{ path('cliente\_edit', { 'id': customer.id }) }}">Editar</a>

</td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

</div>

<div style="margin-bottom:20%"></div>

{% endblock %}

## Controller that required ROLE\_ADMIN

<?php

namespace AppBundle\Controller;

use AppBundle\Entity\Customer;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\Method;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;

use Symfony\Component\HttpFoundation\Request;

use Symfony\Component\HttpFoundation\Session\Session;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\Security;

/\*\*

\* Customer controller.

\* @Security("has\_role('ROLE\_ADMIN')")

\* @Route("cliente")

\*/

class CustomerController extends Controller

{

/\*\*

\* Lists all customer entities.

\*

\* @Route("/", name="cliente\_index")

\* @Method("GET")

\*/

public function indexAction()

{

$em = $this->getDoctrine()->getManager();

$customers = $em->getRepository('AppBundle:Customer')->findAll();

return $this->render('customer/index.html.twig', array(

'customers' => $customers,

));

}

/\*\*

\* Creates a new customer entity.

\*

\* @Route("/new", name="cliente\_new")

\* @Method({"GET", "POST"})

\*/

public function newAction(Request $request)

{

$customer = new Customer();

$form = $this->createForm('AppBundle\Form\CustomerType', $customer);

$form->handleRequest($request);

if ($form->isSubmitted() && $form->isValid()) {

$em = $this->getDoctrine()->getManager();

$em->persist($customer);

$em->flush($customer);

$this->addFlash(

'mensaje',

'Cliente creado correctamente'

);

return $this->redirectToRoute('cliente\_index');

}

return $this->render('customer/new.html.twig', array(

'customer' => $customer,

'form' => $form->createView(),

));

}

/\*\*

\* Displays a form to edit an existing customer entity.

\*

\* @Route("/{id}/edit", name="cliente\_edit")

\* @Method({"GET", "POST"})

\*/

public function editAction(Request $request, Customer $customer)

{

$deleteForm = $this->createDeleteForm($customer);

$editForm = $this->createForm('AppBundle\Form\CustomerType', $customer);

$editForm->handleRequest($request);

if ($editForm->isSubmitted() && $editForm->isValid()) {

$this->getDoctrine()->getManager()->flush();

$this->addFlash(

'mensaje',

'Cliente actualizado correctamente'

);

return $this->redirectToRoute('cliente\_edit', array('id' => $customer->getId()));

}

return $this->render('customer/edit.html.twig', array(

'customer' => $customer,

'edit\_form' => $editForm->createView(),

'delete\_form' => $deleteForm->createView(),

));

}

/\*\*

\* Deletes a customer entity.

\*

\* @Route("/{id}", name="cliente\_delete")

\* @Method("DELETE")

\*/

public function deleteAction(Request $request, Customer $customer)

{

$form = $this->createDeleteForm($customer);

$form->handleRequest($request);

if ($form->isSubmitted() && $form->isValid()) {

$em = $this->getDoctrine()->getManager();

$em->remove($customer);

$em->flush($customer);

}

return $this->redirectToRoute('cliente\_index');

}

/\*\*

\* Creates a form to delete a customer entity.

\*

\* @param Customer $customer The customer entity

\*

\* @return \Symfony\Component\Form\Form The form

\*/

private function createDeleteForm(Customer $customer)

{

return $this->createFormBuilder()

->setAction($this->generateUrl('cliente\_delete', array('id' => $customer->getId())))

->setMethod('DELETE')

->getForm()

;

}

}