**HỌC SYMFONY 3.4**

[**\*. Source tham khảo** 1](#_Toc53030424)

[**\*. Install** 2](#_Toc53030425)

[**\*. Tạo trang đầu tiên** 2](#_Toc53030426)

[**\*. Component** 3](#_Toc53030427)

[**\* Service Container** 4](#_Toc53030428)

[**\*. Bundles và cấu hình** 4](#_Toc53030429)

[**\*. Symfony Reference** 4](#_Toc53030430)

[**\*. Routing (định tuyến)** 4](#_Toc53030431)

[**\*. Controller (Điều khiển)** 5](#_Toc53030432)

[**\*. Template** 6](#_Toc53030433)

[**\*. Configuration** 7](#_Toc53030434)

[**(1) config.yml** 7](#_Toc53030435)

[**(2) parameters.yml** 7](#_Toc53030436)

[**(3) routing.yml** 7](#_Toc53030437)

[**(5) services.yml** 7](#_Toc53030438)

[**\* Security** 7](#_Toc53030439)

[**(1) Initial security.yml Setup (Authentication)** 8](#_Toc53030440)

[***(a) Provider*** 8](#_Toc53030441)

[***(b) access\_control:*** 9](#_Toc53030442)

[**\*. Thêm Authentication** 9](#_Toc53030443)

[**(1) registration** 10](#_Toc53030444)

[**(2) Login** 10](#_Toc53030445)

[**\*. Doctrine ORM** 11](#_Toc53030446)

[**\*. Forms** 20](#_Toc53030447)

[**\* Validation** 21](#_Toc53030448)

[**\*. Frontend** 21](#_Toc53030449)

[**\* Ajax Control** 21](#_Toc53030450)

[**\* Tranlation** 21](#_Toc53030451)

[**\*. Deploy** 22](#_Toc53030452)

# **\*. Source tham khảo**

(1). E:\0000\007.SOURCE\symfony\symfony3-Auth-master\symfony3-Auth-master

<https://github.com/giansalex/symfony3-Auth>

Source 3.2 đăng ký, đăng nhập, đăng xuất

(2). E:\0000\007.SOURCE\symfony\symfony-demo-master\symfony-demo-master

<https://github.com/edlef/symfony-demo>

Source 3.4 xem thêm sửa, xóa có mối quan hệ, nhưng không chứa đăng nhập

# **\*. Install**

- Cài bằng composer ( lưu ý: cài bằng symfony bị thiếu phải cài tiếp các bundles)

|  |
| --- |
| composer create-project symfony/framework-standard-edition my\_project\_name "3.4.\*" |

- Cấu trúc Symfony sau khi cài

|  |
| --- |
| + app/ : Chứa những thứ như cấu hình và mẫu. Về cơ bản, bất kỳ thứ gì không phải là mã PHP đều có ở đây.  + src/ : Mã PHP của bạn nằm ở đây.  99% thời gian, bạn sẽ làm việc trong src / (tệp PHP) hoặc ứng dụng / (mọi thứ khác). Khi bạn tiếp tục đọc, bạn sẽ tìm hiểu những gì có thể được thực hiện bên trong mỗi điều này.  + bin/ : Tệp bin / bảng điều khiển nổi tiếng nằm ở đây (và các tệp thực thi khác, ít quan trọng hơn).  + tests/ : Các bài kiểm tra tự động (ví dụ: Bài kiểm tra đơn vị) cho ứng dụng của bạn có tại đây.  + var/ : Đây là nơi lưu trữ các tệp được tạo tự động, chẳng hạn như tệp bộ đệm (var / cache /), nhật ký (var / logs /) và phiên (var / session /).  + vendor/ : Thư viện của bên thứ ba (tức là "nhà cung cấp") có ở đây! Chúng được tải xuống thông qua trình quản lý gói Composer.  + web / : Đây là gốc tài liệu cho dự án của bạn: đặt mọi tệp có thể truy cập công khai vào đây (ví dụ: CSS, JS và hình ảnh). |

- Chạy ứng dụng

|  |
| --- |
| cd my\_project\_name/  php bin/console server:run |

# **\*. Tạo trang đầu tiên**

Để tạo trang phải có Route và Controller

**\* Kết xuất ra view (Twig)**

|  |
| --- |
| <?php  // src/AppBundle/Controller/LuckyController.php  namespace AppBundle\Controller;  use Symfony\Component\HttpFoundation\Response;  use Symfony\Component\Routing\Annotation\Route;  use Symfony\Bundle\FrameworkBundle\Controller\Controller;  class LuckyController extends Controller  {  /\*\*  \* @Route("/lucky/number")  \*/  public function numberAction()  {  $number = random\_int(0, 100);  return $this->render('lucky/number.html.twig', [  'number' => $number,  ]);  }  } |
| {# app/Resources/views/lucky/number.html.twig #}  <h1>Your lucky number is {{ number }}</h1> |

# **\*. Component**

<https://www.tutorialspoint.com/symfony/symfony_components.htm>

- nằm thư mục symfony/component

- mỗi component cung cấp một tính năng cụ thể, vd: Filesystem, Finder, Console, ClassLoader…

Symfony về cơ bản là một tập hợp các component và bundles

- Component: Trong thư mục \vendor\symfony\symfony\src\Symfony\Component.

+ là tập hợp các lớp cung cấp một chức năng lõi duy nhất (không can thiệp code vào).

+ Symfony có khoảng hơn 30 component, vd: cache, debug, asset..

- Bundles: Tront thư mục \vendor\symfony\symfony\src\Symfony\Bundle

+ Bundles tương tự như plugin nhưng dễ tạo và dễ sử dụng. VD: TwigBundle, FrameworkBundle..

\* Quy trình xử lý request

Bước 1 – Truy cập URL

Bước 2 – Đến máy chủ cái webserver, vd apache.

Bước 3 – Chuyển yêu cầu đến PHP bên dưới, sau đó sẽ gửi nó đến web Symfony.

Bước 4 - HttpKernel là component cốt lõi của khung công tác web Symfony. HttpKernel giải quyết controller của yêu cầu đã cho bằng component Routing và chuyển tiếp yêu cầu đến controller đích.

Bước 5 - Tất cả logic nghiệp vụ diễn ra trong controller mục tiêu.

Bước 6 - Controller sẽ tương tác với Model, tương tác với Datasource thông qua Doctrine ORM.

Bước 7 - Khi controller hoàn tất quá trình, nó sẽ tự tạo phản hồi hoặc thông qua View Engine và gửi nó trở lại máy chủ web.

Bước 8 - Cuối cùng, phản hồi sẽ được máy chủ web gửi đến trình duyệt được yêu cầu.

# **\* Service Container**

<https://www.tutorialspoint.com/symfony/symfony_service_container.htm>

- là vùng chứa các objects.

- **DependencyInjection** component

+ số lượng đối tượng có xu hướng tăng khi ứng dụng phát triển. -> dependency giữa các objects cũng tăng.-> cần sử dụng **DependencyInjection** component để quản lýdependency giữa các objects

+ trong DependencyInjection có class ContainerBuilder để chứa các đối tượng

+ Thêm object vào service container để cần sử dụng thì gọi ra.

+ Có thể cài đặt sẵn các đối tượng trong service container bằng file services.yml bằng namespace của các class tạo object

# **\*. Bundles và cấu hình**

<https://symfony.com/doc/3.4/bundles.html>

- Ứng dụng Symfony của bạn được cài đặt sẵn với một tập hợp các gói, như FrameworkBundle và TwigBundle. Các gói tương tự như ý tưởng về một plugin, nhưng có một điểm khác biệt quan trọng: tất cả các chức năng trong ứng dụng Symfony đều đến từ một gói.

- Các gói được đăng ký trong tệp **app/AppKernel.php.**

- Mỗi gói cung cấp các công cụ (đôi khi gọi là dịch vụ).

- Bạn có thể kiểm soát cách các gói của mình hoạt động thông qua tệp app/config/config.yml. Tệp đó - và các chi tiết khác như môi trường & tham số.

- Bất cứ khi nào tạo một bundle hoặc sử dụng bundles có sẵn, luôn chắn là đã bật nó trong registerBundles(). Sử dụng lệnh: generate:bundle

- Cấu trúc Thư mục Bundles là: controller,denpendencyInjection, resources/config, resources/views, resources/public, test…

# **\*. Symfony Reference**

Tham khảo Symfony

# **\*. Routing (định tuyến)**

<https://symfony.com/doc/3.4/routing.html>

Chức năng:

+ Định tuyến giúp tạo url seo ánh xạ đến controller trong ứng dụng

+ Tạo url trong tempates và controllers

+ Tải tài nguyên từ bundles và chỗ khác.

+ Gỡ lỗi định tuyến.

1. Định tuyến có thể được cấu hình trong YAML, XML, PHP or annotations

- Annotations: trong ghi chú trong controller trên method

- Yaml: trong file # app/config/routing.yml

- XML: trong file <!-- app/config/routing.xml -->

- PHP: trong file // app/config/routing.php

# **\*. Controller (Điều khiển)**

<https://symfony.com/doc/3.4/controller.html>

Controller là một hàm PHP mà bạn tạo để đọc thông tin từ đối tượng Symfony’s Request và tạo và trả về một đối tượng Response. Phản hồi có thể là một trang HTML, JSON, XML, tải xuống tệp, chuyển hướng, lỗi 404 hoặc bất kỳ thứ gì khác mà bạn có thể mơ thấy.

Lớp controller trong symphony:

+ Symfony\Bundle\FrameworkBundle\Controller\Controller

+ Symfony\Bundle\FrameworkBundle\Controller\AbstractController

Các method trong controller

(1) Sinh url: $url = $this->generateUrl('blog\_show', ['slug' => 'slug-value']);

(2) Chuyển hướng:

|  |
| --- |
| public function indexAction()  {  // redirects to the "homepage" route  return $this->redirectToRoute('homepage');  // does a permanent - 301 redirect  return $this->redirectToRoute('homepage', [], 301);  // redirects to a route with parameters  return $this->redirectToRoute('blog\_show', ['slug' => 'my-page']);  // redirects to a route and maintains the original query string parameters  return $this->redirectToRoute('blog\_show', $request->query->all());  // redirects externally  return $this->redirect('http://symfony.com/doc');  } |

+ Phương thức redirect () không kiểm tra đích của nó theo bất kỳ cách nào. Nếu bạn chuyển hướng đến một số URL do người dùng cuối cung cấp, ứng dụng của bạn có thể mở lỗ hổng bảo mật chuyển hướng chưa được kiểm chứng.

+ Phương thức redirectToRoute () chỉ đơn giản là một phím tắt tạo đối tượng Response chuyên chuyển hướng người dùng

(3) Kết xuất mẫu return $this->render('lucky/number.html.twig', ['number' => $number]);

(4) Tìm nạp Services as Controller Arguments

(5) Truy cập vùng chứa

|  |
| --- |
| $templating = $this->get('templating');  $router = $this->get('router');  $mailer = $this->get('mailer');  // you can also fetch parameters  $someParameter = $this->getParameter('some\_parameter'); |

(6) Quản lý lỗi và 404 page

(7) Đối tượng Request như là một Tham số controller

(8) Quản lý Session

(9) Flash Message

(10) Đối tượng Request và Response

(11) Sử lý Json

# **\*. Template**

<https://symfony.com/doc/3.4/templating.html>

Là một tệp văn bản dựa trên (HMTL, XML, CSV, PHP…)

Symfony đóng gói một ngôn ngữ tạo template được gọi là Twig.

Cú pháp:

{{...}}: Chứa biến hoặc biểu thức

{% ... %}: Dùng thực hiện câu lệnh, vòng lặp

{# ... #}: Để bình luận, không hiển thị

Filters: bộ lộc để hiển thị. VD: {{ title|upper }} chuyển chữ hoa trước khi hiển thị

VD:

|  |
| --- |
| {% for i in 1..10 %}  <div class="{{ cycle(['even', 'odd'], i) }}">  <!-- some HTML here -->  </div>  {% endfor %} |

- Các ưu điểm của Twig

+ Twig Template Caching

+ Template Inheritance and Layouts

- Tên template và chỗ lưu. Theo mặc định, các mẫu có thể tồn tại ở hai vị trí khác nhau:

+ app/Resources/views/

+ vendor/path/to/CoolBundle/Resources/views/

- Tham chiếu template trong Bundle

- Các công cụ có sẵn trong Twig

(1) Including other Templates:

(2) Linking to Pages

(3) Linking to Assets

- Including Stylesheets and JavaScripts in Twig

- Tham chiếu Request, User, Session

# **\*. Configuration**

## **(1) config.yml**

Mỗi ứng dụng Symfony bao gồm tập hợp các bundles để thêm các tool (services).

Mỗi Bundles có thể được tùy chỉnh trong file # app/config/config.yml

## **(2) parameters.yml**

## **(3) routing.yml**

## **(5) services.yml**

Ứng dụng của bạn có đầy đủ các đối tượng hữu ích: “Mailer” object có thể giúp bạn gửi email trong khi một đối tượng khác có thể giúp bạn lưu mọi thứ vào cơ sở dữ liệu. Hầu hết mọi thứ mà ứng dụng của bạn “làm” đều được thực hiện bởi một trong những đối tượng này. Và mỗi khi bạn cài đặt một bundle mới, bạn sẽ có quyền truy cập vào nhiều hơn nữa!

Trong Symfony, các đối tượng hữu ích này được gọi là services và mỗi services nằm bên trong một đối tượng rất đặc biệt được gọi là service container.

- Tìm nạp, sử dụng Services

- Tạo cấu hình services trong container

- Tự độn load services trong file services.yml

# **\* Security**

[1) Khởi tạo thiết lập security.yml (Authentication)](https://symfony.com/doc/3.4/security.html#initial-security-yml-setup-authentication)

[A) Cấu hình làm thế nào User Authenticate](https://symfony.com/doc/3.4/security.html#a-configuring-how-your-users-will-authenticate)

[B) Cấu hình Users Loaded](https://symfony.com/doc/3.4/security.html#b-configuring-how-users-are-loaded)

[C) Mã hóa User Password](https://symfony.com/doc/3.4/security.html#c-encoding-the-user-s-password)

[D) Cấu hình hoàn thành!](https://symfony.com/doc/3.4/security.html#d-configuration-done)

[2) Từ chối truy cập, Roles và Authorization](https://symfony.com/doc/3.4/security.html#denying-access-roles-and-other-authorization)

[Roles](https://symfony.com/doc/3.4/security.html#roles)

[Thêm code to Deny Access](https://symfony.com/doc/3.4/security.html#add-code-to-deny-access)

[Kiểm tra User is Logged (IS\_AUTHENTICATED\_FULLY)](https://symfony.com/doc/3.4/security.html#checking-to-see-if-a-user-is-logged-in-is-authenticated-fully)

[Access Control Lists (ACLs): Bảo mật riêng từng Database Objects](https://symfony.com/doc/3.4/security.html#access-control-lists-acls-securing-individual-database-objects)

[3) Truy xuất User Object](https://symfony.com/doc/3.4/security.html#retrieving-the-user-object)

[Luôn kiểm User is Logged In](https://symfony.com/doc/3.4/security.html#always-check-if-the-user-is-logged-in)

[Truy xuất User in a Template](https://symfony.com/doc/3.4/security.html#retrieving-the-user-in-a-template)

[Đăng xuất](https://symfony.com/doc/3.4/security.html#logging-out)

[Phân cấp Roles](https://symfony.com/doc/3.4/security.html#hierarchical-roles)

[Final Words](https://symfony.com/doc/3.4/security.html#final-words)

[Learn More](https://symfony.com/doc/3.4/security.html#learn-more)

Xác thực: [Authentication (Identifying/Logging in the User)](https://symfony.com/doc/3.4/security.html#authentication-identifying-logging-in-the-user)

Ủy quyền; [Authorization (Denying Access)](https://symfony.com/doc/3.4/security.html#authorization-denying-access)

Khác; [Other Security Related Topics](https://symfony.com/doc/3.4/security.html#other-security-related-topics)

## **(1) Initial security.yml Setup (Authentication)**

app/config/security.yml

- Các từ khóa trong security.yml: <https://symfony.com/doc/3.4/reference/configuration/security.html>

***A) Cấu hình làm thế nào User Authenticate***

### ***(a) Provider***

Symfony tải tài khoản để bảo mật từ nhiều nơi, DB, OAuth server…., có thể tải provider kết hợp từ nhiều nơi, hoặc chỉ tải từ một nơi.

***- Các bước load entity trong security.yml***  
<https://symfony.com/doc/3.4/security/entity_provider.html>   
<https://symfony.com/doc/3.4/security/custom_provider.html>   
<https://symfony.com/doc/3.4/security/multiple_user_providers.html>

B1. Tạo Entity // src/AppBundle/Entity/User.php  
B2. Chạy lệnh cập nhật DB: php bin/console doctrine:schema:update –force  
B3. Cấu hình security.yml sử dụng entity

+ encoders: mật khẩu trong DB được mã hóa bằng bcrypt  
+ providers: tên là our\_db\_provider truy vân từ entity User, theo thuộc tính username  
+ Tên our\_db\_provider đặt sao cũng được, nhưng phải khớp với tên provider trong firewall Hoặc nếu không cung cấp provider trong firewall thì provider đầu tiên sẽ được sử dụng.

|  |
| --- |
| # app/config/security.yml  security:  encoders:  AppBundle\Entity\User:  algorithm: bcrypt  # ...  providers:  our\_db\_provider:  entity:  class: AppBundle:User  property: username  # if you're using multiple entity managers  # manager\_name: customer  firewalls:  main:  pattern: ^/  http\_basic: ~  provider: our\_db\_provider  # ... |

### ***(b) access\_control:***

Xác định sự bảo vệ URL, sử dụng khi người dùng truy cập backend, => hiện trang login

|  |
| --- |
| access\_control:  # require ROLE\_ADMIN for /admin\*  - { path: '^/admin', roles: ROLE\_ADMIN } |

+ Các loại ROLE: ROLE\_USER, ROLE\_ADMIN, ROLE\_SUPER\_ADMIN, ROLE\_REMOTE, ROLE\_OPERATOR

# **\*. Thêm Authentication**

<https://symfony.com/doc/3.4/components/security.html>

<https://symfony.com/doc/3.4/security.html>

Lý thuyết chung

Các vấn đề cần chú ý:

+ Authenticate: User xác thực như thế nào (thông qua firewalls)  
+ Provider: User được loaded từ đâu  
+ Encoders: Mã hóa mật khẩu như thế nào  
+ Authorization: Ủy quyền User để access  
+ Role: Đặt tên vai trò cho user  
+ Access\_control: role có quyền truy cập url nào  
+ Get User, kiểm tra user đăng nhập.

## **(1) registration**

<https://symfony.com/doc/3.4/doctrine/registration_form.html> Form đăng ký => OK

B1. Tạo entity // src/AppBundle/Entity/User.php

B2. Tạo form cho entity // src/AppBundle/Form/**UserType.php**  
Thêm // src/AppBundle/Form/UserType.php để dùng lệnh trong controller register

$form = $this->createForm(UserType::class, $user);  
…………

return $this->render('registration/register.html.twig',['form' => $form->createView()]);

Lấy mẫu từ UserType để hiển thị ra view 'registration/register.html.twig'

B3. Tạo controller quản lý form submit src/AppBundle/Controller/RegistrationController.php  
B4. Cấu hình thuật toán mã hóa password trong # app/config/security.yml  
B5. Cấu hình đường dẫn đăng ký (nếu không dùng annotation routing) trong # app/config/routing.yml  
B6. Tạo mẫu view registration {# app/Resources/views/registration/register.html.twig #}

B7. Cập nhật bảng User cho DB: php bin/console doctrine:schema:update –force

Lưu ý tên class trong Entity phải có

|  |
| --- |
| /\*\*  \* **@ORM\Entity**  \* @ORM\Table(name="student")  \*/ |

B8. Dùng email thay thế cho username (nếu muốn) trong code // src/AppBundle/Entity/User.php  
B9. Thêm checkbox “Đồng ý quy định” trong // src/AppBundle/Form/UserType.php

**Hoàn thành**

## **(2) Login**

<https://symfony.com/doc/3.4/security/form_login_setup.html> Form đăng nhập

B1. Kích hoạt login dưới firewall trong file # app/config/security.yml. Lúc này, hệ thống bảo mật bắt đầu quá trình xác thực, nó sẽ chuyển hướng người dùng đến biểu mẫu login/.

B2. Tạo file controller quản lý login // src/AppBundle/Controller/SecurityController.php

B3. Định tuyến cho url login/ sử dụng annotation // src/AppBundle/Controller/SecurityController.php

B4. Thêm logic cho login // src/AppBundle/Controller/SecurityController.php  
B5. Tạo mẫu view login

<https://symfony.com/doc/3.4/security/form_login.html> Tùy chỉnh sau khi đăng nhập

<https://symfony.com/doc/3.4/security/remember_me.html> thêm Remember me

<https://symfony.com/doc/current/security.html#logging-out> cách logout

- Security component được chia thành nhiều thành phần phụ nhỏ hơn có thể được sử dụng riêng biệt:

+ symfony / security-core: Nó cung cấp tất cả các tính năng bảo mật phổ biến, từ xác thực đến ủy quyền và từ mã hóa mật khẩu đến tải người dùng.

+ symfony / security-http: Nó tích hợp thành phần phụ cốt lõi với giao thức HTTP để xử lý các yêu cầu và phản hồi HTTP.

+ symfony / security-csrf: Nó cung cấp khả năng bảo vệ chống lại các cuộc tấn công CSRF.

+ symfony / security -guard: Nó mang nhiều lớp xác thực lại với nhau, cho phép tạo ra các hệ thống xác thực phức tạp.

**\*. Doctrine ORM**

<https://www.doctrine-project.org/projects/doctrine-orm/en/2.7/reference/basic-mapping.html>   
<https://symfony.com/doc/3.4/reference/constraints.html>

- Giúp ánh xạ object đến DB.

Tạo project quản lý product

B1. Cấu hình DB trong app/config/parameters.yml  
B2. Tạo DB với lệnh: php bin/console doctrine:database:create  
B3. Tạo Entity Class  
B4. Tạo DB từ euntity: php bin/console doctrine:schema:update –force  
B5. Lưu đối tượng vào DB (Tạo)  
B6. Lấy đối tượng từ DB (Xem)  
B7. Cập nhật đối tượng vào DB (Sửa)  
B8. Xóa đối tượng.  
B9. Query với SQL  
B10. Query với Doctrine’s Query Builder

- Các câu lệnh cơ bản

+ persist(): thêm lệnh vào queue, // cho Doctrine biết bạn muốn lưu Sản phẩm

+ flush() : //executes the queries

+ addFlash() gửi thông báo message qua session

+ getDoctrine(): để tìm nạp EntityManager

+ $this->getDoctrine()-> getManager() lấy thực thể quản lý để thêm

+ $this->getDoctrine()->getRepository(): lấy thực thể quản lý để hiển thị, sửa, xóa

// có thể viết thêm chức năng cho Reposity bỏ trong thư mục Repository VD: findAllOrderedByTitle()

+ handleRequest(): đọc dữ liệu từ $request

|  |
| --- |
| <?php  namespace AppBundle\Entity;  use AppBundle\Entity\Traits\TimestampableTrait; // sử dụng Trait  use Doctrine\Common\Collections\ArrayCollection;  use Doctrine\ORM\Mapping as ORM; //sử dụng Doctrine  use Symfony\Component\Validator\Constraints as Assert;  /\*\*  \* @ORM\Table(name="article") // tên bảng  \* @ORM\Entity(repositoryClass="AppBundle\Repository\ArticleRepository") // reposity chỗ để viết thêm chức năng cho entity  \*/  class Article  {  use TimestampableTrait;  /\*\*  \* @var integer $id  \*  \* @ORM\Column(name="id", type="bigint") // tên, kiểu dự liệu  \* @ORM\Id  \* @ORM\GeneratedValue(strategy="IDENTITY") // định danh khóa chính  \*/  private $id;  /\*\*  \* @Assert\NotBlank() // ko trống  \* @var string $code //chuỗi  \*  \* @ORM\Column(name="code", type="string", length=255, nullable=false)  \*/  private $code;  /\*\*  \* @Assert\NotBlank()  \* @var string $title  \*  \* @ORM\Column(name="title", type="string", length=255, nullable=false)  \*/  private $title;  /\*\*  \* @Assert\NotBlank()  \* @var datetime $releaseDate // kiểu ngày  \*  \* @ORM\Column(name="release\_date", type="datetime", nullable=true)  \*/  private $releaseDate;  /\*\*  \* @var float $length  \*  \* @ORM\Column(name="length", type="integer", nullable=true)  \*/  private $length;  /\*\*  \* @Assert\NotBlank()  \* @var datetime $duration  \*  \* @ORM\Column(name="duration", type="integer", nullable=true)  \*/  private $duration;  /\*\*  \* @var text $description  \*  \* @ORM\Column(name="description", type="text", nullable=true)  \*/  private $description;  /\*\*  \* @Assert\NotBlank()  \* @var float $price  \*  \* @ORM\Column(name="price", type="float", nullable=true)  \*/  private $price;  /\*\*  \* @var float $dvd  \*  \* @ORM\Column(name="dvd", type="boolean", options={"default":false})  \*/  private $dvd;  /\*\*  \* @var float $blueray  \*  \* @ORM\Column(name="blueray", type="boolean", options={"default":false})  \*/  private $blueray;  /\*\*  \* @ORM\ManyToOne(targetEntity="ArticleType", inversedBy="articles")  \* @ORM\JoinColumn(name="type\_id", referencedColumnName="id")  \*/  private $type;  /\*\*  \* @ORM\ManyToOne(targetEntity="Author", inversedBy="articles")  \* @ORM\JoinColumn(name="author\_id", referencedColumnName="id")  \*/  private $author;  /\*\*  \* @ORM\ManyToMany(targetEntity="Actor", cascade={"persist"})  \* @ORM\JoinTable(name="articles\_actors")  \*/  private $actors;  public function \_\_construct()  {  $this->actors = new \Doctrine\Common\Collections\ArrayCollection();  }  /\*\*  \* @return int  \*/  public function getId()  {  return $this->id;  }  /\*\*  \* @return string  \*/  public function getCode()  {  return $this->code;  }  /\*\*  \* @param string $code  \*/  public function setCode($code)  {  $this->code = $code;  return $this;  }  /\*\*  \* @return string  \*/  public function getTitle()  {  return $this->title;  }  /\*\*  \* @param string $title  \*/  public function setTitle($title)  {  $this->title = $title;  return $this;  }  /\*\*  \* @return string  \*/  public function getAuthor()  {  return $this->author;  }  /\*\*  \* @param Author $author  \* @return $this  \*/  public function setAuthor(Author $author)  {  $this->author = $author;  return $this;  }  /\*\*  \* @return datetime  \*/  public function getReleaseDate()  {  return $this->releaseDate;  }  /\*\*  \* @param $releaseDate  \* @return $this  \*/  public function setReleaseDate($releaseDate)  {  $this->releaseDate = $releaseDate;  return $this;  }  /\*\*  \* @return float  \*/  public function getLength()  {  return $this->length;  }  /\*\*  \* @param float $length  \*/  public function setLength($length)  {  $this->length = $length;  return $this;  }  /\*\*  \* @return datetime  \*/  public function getDuration()  {  return $this->duration;  }  /\*\*  \* @param datetime $duration  \*/  public function setDuration($duration)  {  $this->duration = $duration;  return $this;  }  /\*\*  \* @return text  \*/  public function getDescription()  {  return $this->description;  }  /\*\*  \* @param text $description  \*/  public function setDescription($description)  {  $this->description = $description;  return $this;  }  /\*\*  \* @return float  \*/  public function getPrice()  {  return $this->price;  }  /\*\*  \* @param float $price  \*/  public function setPrice($price)  {  $this->price = $price;  return $this;  }  /\*\*  \* @return mixed  \*/  public function getType()  {  return $this->type;  }  /\*\*  \* @param ArticleType $type  \* @return $this  \*/  public function setType(ArticleType $type)  {  $this->type = $type;  return $this;  }  /\*\*  \* @return mixed  \*/  public function getActors()  {  return $this->actors;  }  public function addActor(Actor $actor)  {  $this->actors->add($actor);  }  public function removeTag(Actor $actor)  {  $this->actors->remove($actor);  }  /\*\*  \* @return mixed  \*/  public function isDvd()  {  return $this->dvd;  }  /\*\*  \* @param mixed $dvd  \*/  public function setDvd($dvd)  {  $this->dvd = $dvd;  }  /\*\*  \* @return mixed  \*/  public function isBlueray()  {  return $this->blueray;  }  /\*\*  \* @param mixed $blueray  \*/  public function setBlueray($blueray)  {  $this->blueray = $blueray;  }  } |

# **\*. Forms**

<https://symfony.com/doc/3.4/forms.html>

<https://symfony.com/doc/3.4/reference/forms/types/entity.html>

B1. Tạo form trong controller / hoặc sử dụng Form class  
B2. Render form  
B3. Quản lý form submit  
B4. Xác thực form (validation)  
B5. Xây dựng các field type cho input

***- hiển thị biểu mẫu hoàn chỉnh:***

+ form\_start (form): Hiển thị thẻ bắt đầu của biểu mẫu, bao gồm thuộc tính enctype chính xác khi sử dụng tải lên tệp.

+ form\_widget (form): Hiển thị tất cả các trường, bao gồm bản thân phần tử trường, nhãn và bất kỳ thông báo lỗi xác thực nào cho trường.

+ form\_end (form): Hiển thị thẻ kết thúc của biểu mẫu và bất kỳ trường nào chưa được hiển thị, trong trường hợp bạn tự kết xuất từng trường. Điều này hữu ích để hiển thị các trường ẩn và tận dụng Bảo vệ CSRF tự động.

***- Field Type Tùy chỉnh***

+ 'widget' => 'single\_text' : tạo 1 dòng lúc kiểu là DateType  
+ 'required' => false : không bắt buộc  
+ 'label' => 'Due Date',: nhãn của field

- Nên xây dựng một lớp để chứa logic xây dựng biểu mẫu: VD: UserType, TaskType…. Chỉ gọi ra để dùng thôi.

- Nên thêm chỉ định lớp tham chiếu cho form bằng configureOptions

- Không muốn mapped 1 field với class object thì có thể dùng

->add('agreeTerms', CheckboxType::class, [**'mapped' => false**])

- Lấy dữ liệu từ form $form->get('agreeTerms')->getData(); $form->get('agreeTerms')->setData();

**- Thêm class, id cho form, input**

1) thêm class cho input trong EntityType, VD: StudentType

|  |
| --- |
| $builder  ………  ->add('firstname', TextType::class, ['attr' => ['class' => 'myClass', 'id'=>'plgid']])  ……… |

2) thêm class cho form trong view

|  |
| --- |
| {{ form\_start(form, { 'attr' : { 'class': 'myClass' } }) }} |

# **\* Validation**

# **\*. Frontend**

- Symfony cung cấp thư viện JavaScript thuần - được gọi là Webpack Encore để làm việc với CSS và JavaScript

- Hoặc gắn css, js, image trong thư mục web

<link rel="stylesheet" href="{{ asset('css/phonglg.css') }}">

# **\* Ajax Control**

Lưu ý khi dùng JsonResponse để trả dữ liệu phải sử dụng nó nếu không sẽ báo lỗi.

use Symfony\Component\HttpFoundation\JsonResponse;

# **\* Tranlation**

<https://symfony.com/doc/3.4/translation.html>

1. Cách sử dụng cơ bản

B1. Tạo nguồn translate: VD tiếng pháp

|  |
| --- |
| # app/Resources/translations/messages.fr.yml  Symfony is great: J'aime Symfony PLG |

B2. Thiết lập sử dụng tiếng pháp fr

|  |
| --- |
| # app/config/config.yml  framework:  translator: { fallbacks: [fr] } |

B3. Sử dụng trong controller

|  |
| --- |
| return new Response('<html><body>HELLO page!'.$this->get('translator')->trans('Symfony is great').'</body></html>');  // $this->get('translator')->trans('Symfony is great') |

\* Lưu ý: từ khóa phải đúng, phân biệt chữ hoa chữ thường

2. Sử dụng keyword message

3. Sử dụng biến trong message

4. Sử dụng  [transChoice()](https://github.com/symfony/symfony/blob/3.4/src/Symfony/Component/Translation/Translator.php)

5. Sử dụng translate trong twig template

6. Có thể tùy chỉnh ngôn ngữ trong lúc dùng hàm trans() và transChoice()

# **\*. Deploy**

Bỏ debug tron gconfig\_dev.yml: toolbar: false

Cám ơn