

Création application : symfony new

LTS web my_project --version=lts --webapp

Version 6.2 web my_project --version="6.2.*" --webapp

LTS my_project --version=lts

La dernière version my_project

la demo symfony new demo --demo

LTS signifie Long Time Support soit version supportée sur le long terme.

Création inscription et connexion

symfony console

connexion (login) make:auth

Inscription (register) make:registration-form

make:user

Créer les formulaires pour l'inscription et la connexion des utilisateurs ainsi que le contrôleur user adapté

Création d'une route par configuration

Ajouter dans le fichier **config/routes.yaml**

nom_de_la_route:

path: /inscription ou / ou /machin/truc etc (url)

controller: App\Controller\nom_du_controllerController::nom_de_methode

Liste des routes du projet

liste toutes les routes php bin/console debug:router

liste une route php bin/console debug:router <nom de la route>

liste l'enchaînement des fonctions utilisées php bin/console router:match <nom de la route>

Symfony

server:start --no-tls lance le serveur en http

server:ca:install pour un serveur en https

server:start -d en background

server:stop stop le serveur (ctr c)

console about information version du projet

check:requirements valide l'environnement

symfony console

list donne la liste de toutes les commandes

liste make donne la liste des commandes du maker

make:controller --help donne la liste des options de la commande

doctrine:database:create Crée la db déclarée dans .env

symfony console (cont)

d:d:c Crée la db déclarée dans .env

make:user Crée la table user associée à l'authentification

make:entity Crée une table

make:entity --regenerate Génère les méthodes getter et setter manquantes

make:entity --overwrite Regénère toutes les méthodes getter et setter

make:migration Génère les fichiers de migration

doctrine:migrations:migrate Exécute les fichiers de migration

m:mig Génère les fichiers de migration

d:m:m Exécute les fichiers de migration

doctrine:migrations:diff Génère un fichier de migration en comparant le schéma interne et le schéma de la base

debug:autowiring Liste les classes autochargeable



By AGD (agarciautaitre)

cheatography.com/agarciautaitre/

Not published yet.

Last updated 7th March, 2023.

Page 1 of 3.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>

symfony console (cont)

debug:container	Liste les services disponibles pour l'application
dbal:run-sql	interroge la base de données
'SELECT * FROM product'	

Les fichiers de migrations sont nommés migrations\Versiondateheure.php et ils contiennent les ordres sql qui vont modifier la base

php bin/console

server:dum	??
p	
server:log	
list	liste les commandes de console
list make	liste les commandes du maker

Contrôleur

Créer un contrôleur	php bin/console make:controller nom_du_controleur
	symfony console make:controller nom_du_controleur

Les vues

Installation	composer require twig
Les vues = Twig sous symfony	

Exemples de déclaration de route

```
#[Route('/api/posts/{id}', methods: ['GET', 'HEAD'])]
#[Route('/contact', name: 'contact', condition: "context.getMethod() in ['GET', 'HEAD'] and request.headers.get('User-Agent') matches '/firefox/i'", // expressions can also include config parameters: // condition: "request.headers.get('User-Agent') matches '%app.allowed_browser-s%'" )]
#[Route('/posts/{id}', name: 'post_show', // expressions can retrieve route parameter values using the "params" variable condition: "params[id] < 1000" )]
// Controller (using an alias):
#[Route('/posts/{id}', name: 'post_show', condition: "service('route_checker').check(request)")]
use Symfony\Bundle\FrameworkBundle\Routing\Attribute\AsRoutingConditionService;
use Symfony\Component\HttpFoundation\Request;
#[AsRoutingConditionService(alias: 'route_checker')]
class RouteChecker
{
```

Exemples de déclaration de route (cont)

```
public function check(Request $request): bool
{
    // ...
}
```

php

-m liste les modules php installés

Composer require

https://packagist.org/packages/symfony/	Liste des packages dispo sur composer
symfony/orm-pack	Installe Doctrine
--dev symfony/maker-bundle	Installe le Maker
symfony/validator	Les contraintes de validation pour Forms et DB
symfony/http-foundation	installe session
symfony/form	Installe les formulaires

Usage des annotations pour les routes

Utiliser le package annotation	composer require annotations
--------------------------------	------------------------------



By AGD (agarciautaitre)

cheatography.com/agarciautaitre/

Not published yet.

Last updated 7th March, 2023.

Page 2 of 3.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>

Création route annotation

```
<?php
namespace App\Controller;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
class NomDuContrôleurController {
    /**
     * @Route(
     *     "/chemin_d_url/" ,
     *     {param1} ,
     *     name="nom_de_la_route",
     *     methods={ "GET", "POST", "HEAD", "PUT", etc},
     *     requirements={"param1":"regex1", "param2":"regex2"},
     *     defaults={"param1":"default1", "param2":"default2"},
     *     host="my.host",
     *     schemes={ "http", "https"},
     *     priority=1
     * )
     */
    public function nomTraitements($param1): Response {
        // ...

        return $this->render('nom_replacement/nom_du_template.html.twig', [
            'nomDuParametre' => $valeur_du_parametre,
            'param1' => $param1,
        ]);
    }
}
```

Option pour les routes

name="routeName"	Donner un nom à une Route
methods={"method1", "method2"}	retreindre l'accès au requête HTTP de type
Type de requête HTTP	GET,POST,-DEL,PUT,HEAD etc
host="nom ou IP du host"	retreindre l'accès à un host en particulier
requirements={"param1":"regex1", "param2":"regex2"}	appliquer des règles aux paramètres
defaults={"param1":"default1", "param2":"default2"}	Valeur par défauts des paramètres

Création d'un contrôleur en lien avec la base

```
//
src/Controller/ProductController.php
namespace App\Controller;
// ...
use App\Entity\Product;
use Doctrine\Persistence\ManagerRegistry;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
class ProductController extends AbstractController {
    #[Route('/product', name: 'create_product')]
    // ...
}
```

Création d'un contrôleur en lien avec la base (cont)

```
> public function createProduct(ManagerRegistry $doctrine): Response
{
    $entityManager = $doctrine->getManager();
    $product = new Product();
    $product->setName('Keyboard');
    $product->setPrice(1999);
    $product->setDescription('Ergonomic and stylish!');
    // tell Doctrine you want to (eventually) save the Product (no queries yet)
    $entityManager->persist($product);
    // actually executes the queries (i.e. the INSERT query)
    $entityManager->flush();
    return new Response('Saved new product with id '.$product->getId());
}
```

Tester le contrôleur avec : <http://localhost:8000/product>
 Vérifier la base de donnée : `php bin/console dbal:run-sql 'SELECT * FROM product'`
<https://symfony.com/doc/current/doctrine.html#migrations-creating-the-database-tables-schema>



By AGD (agarciadutaitre)

Not published yet.

Last updated 7th March, 2023.

Page 3 of 3.

Sponsored by [Readable.com](https://readable.com)

Measure your website readability!

<https://readable.com>