

# Symfony with Doctrine (ORM)

## Doctrine

The Doctrine Project is the home to several PHP libraries primarily focused on database storage and object mapping. The core projects are the [Object Relational Mapper \(ORM\)](#) and the [Database Abstraction Layer \(DBAL\)](#) it is built upon.

# Symfony Components

---

- **Console (scaffolding)**
- **Routing (URL rewriting)**
- **Doctrine (database Object Relational Mapping)**
- **Security (authentication)**
- **Form (form generation)**
- **Validator (input validation)**
- **Fixtures (dummy data generation)**
- **Profiler (debugging menu bar)**
- **Testing (generates tests)**

# Symfony Console

---

\$ php bin/console

Available commands:

cache	
cache:clear	Clears the cache
debug	
debug:autowiring	Lists classes/interfaces you can use for autowiring
debug:form	Displays form type information
debug:router	Displays current routes for an application
debug:twig	Shows a list of twig functions, filters, globals and tests
doctrine	
doctrine:database:create	Creates the configured database
doctrine:database:drop	Drops the configured database
doctrine:database:import	Import SQL file(s) directly to Database.
doctrine:migrations:migrate	[migrate] Execute a migration
make	
make:auth	Creates a Guard authenticator of different flavors
make:controller	Creates a new controller class
make:entity	Creates or updates a Doctrine entity class, and optionally an
make:fixtures	Creates a new class to load Doctrine fixtures
make:form	Creates a new form class
make:migration	Creates a new migration based on database changes
make:registration-form	Creates a new registration form system
make:reset-password	Create reset password controller, entity, and repositories
make:user	Creates a new security user class
make:validator	Creates a new validator and constraint class
router	
router:match	Helps debug routes by simulating a path info mat

# Doctrine Microposts entity 1

---

```
<?php

namespace App\Entity;

use App\Repository\MicropostRepository;
use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Table(name="microposts")
 * @ORM\Entity(repositoryClass=MicropostRepository::class)
 */
class Micropost
{
    /**
     * @ORM\Id
     * @ORM\GeneratedValue
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="text", nullable=true)
     */
    private $content;

    /**
     * @ORM\ManyToOne(targetEntity=User::class, inversedBy="microposts")
     * @ORM\JoinColumn(nullable=false)
     */
    private $user;

    /**
     * @ORM\Column(type="datetime")
     */
    private $created_at;

    /**
     * @ORM\Column(type="datetime", nullable=true)
     */
    private $updated_at;
}
```

# Doctrine Microposts entity 2

---

```
$ php bin/console make:migration
```

```
$ php bin/console doctrine:migrations:migrate
```

```
MariaDB [db_a12345]> show create table microposts;
```

```
CREATE TABLE `microposts` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `user_id` int(11) NOT NULL,  
  `content` longtext COLLATE utf8mb4_unicode_ci DEFAULT NULL,  
  `created_at` datetime NOT NULL,  
  `updated_at` datetime DEFAULT NULL,  
  PRIMARY KEY (`id`),  
  KEY `IDX_A51D467BA76ED395` (`user_id`),  
  CONSTRAINT `FK_A51D467BA76ED395` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`)  
  ) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci
```

# Doctrine Microposts Repository 1

---

```
<?php

namespace App\Repository;

use App\Entity\Micropost;
use App\Entity\User;
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
use Doctrine\Persistence\ManagerRegistry;
use Doctrine\ORM\EntityManagerInterface;

class MicropostRepository extends ServiceEntityRepository
{
    private $em;
    public function __construct(ManagerRegistry $registry, EntityManagerInterface $em)
    {
        parent::__construct($registry, Micropost::class);
        $this->em = $em;
    }

    public function get_posts()
    {
        $qb = $this->createQueryBuilder('m')
            ->innerJoin('m.user', 'u')
            ->addSelect('u');
        return $qb
            ->orderBy('m.updated_at', 'DESC')
            ->getQuery()
            ->getResult()
        ;
    }
}
```

# Doctrine Microposts Repository 2

---

```
public function new_blog($user,$blog)
{
    $micropost = new Micropost();
    $micropost->setUser($user);
    $micropost->setContent($blog);
    $micropost->setCreated_at(new \DateTime());
    $micropost->setUpdated_at(new \DateTime());
    $this->em->persist($micropost);
    $this->em->flush();
}
```

# Symfony Security 1

---

```
$cat security.yaml
```

```
security:
  encoders:
    App\Entity\User:
      algorithm: md5
      encode_as_base64: false
      iterations: 0

    app_user_provider:
      entity:
        class: App\Entity\User
        property: email

  firewalls:
    dev:
      pattern: ^/(_(profiler|wdt)|css|images|js)/
      security: false

  main:
    anonymous: lazy
    provider: app_user_provider
    guard:
      authenticators:
        - App\Security\LoginFormAuthenticator
      logout:
        path: logout
        target: blog.index
      remember_me:
        name: 'REMEMBERME'
        secret: '%kernel.secret%'
        lifetime: 2592000 # 30 days in seconds
        remember_me_parameter: '_remember_me'
```



# Symfony Security 2

---

```
<?php

namespace App\Security;
.
.
.
use Symfony\Component\Security\Guard\Authenticator\AbstractFormLoginAuthenticator;

class LoginFormAuthenticator extends AbstractFormLoginAuthenticator
{
    private $userRepository;
    private $router;
    private $csrfTokenManager;
    private $encoder;
    private $response;
    public function __construct(UserRepository $userRepository, RouterInterface $router,
CsrfTokenManagerInterface $csrfTokenManager, UserPasswordEncoderInterface $encoder)
    {
        $this->userRepository = $userRepository;
        $this->router = $router;
        $this->csrfTokenManager = $csrfTokenManager;
        $this->encoder = $encoder;
    }

    public function supports(Request $request)
    {
        // do your work when we're POSTing to the login page
        return $request->attributes->get('_route') === 'login'
            && $request->isMethod('POST');
    }
}
```

# Symfony Security 3

---

```
public function getCredentials(Request $request)
{
    $credentials = [
        'email' => $request->request->get('email'),
        'password' => $request->request->get('password'),
        'token' => $request->request->get('token'),
    ];

    $request->getSession()->set(
        Security::LAST_USERNAME,
        $credentials['email']
    );

    return $credentials;
}

public function getUser($credentials, UserProviderInterface $userProvider)
{
    $token = new CsrfToken('login_form', $credentials['token']);
    if (!$this->csrfTokenManager->isTokenValid($token)) {
        throw new InvalidCsrfTokenException();
    }

    return $this->userRepository->findOneBy(['email' => $credentials['email']]);
}

.
.
.
```

# Symfony Form generation 1

---

```
use App\Form\RegisterType;

/**
 * @Route("/blog/register", name="register")
 */
public function register(Request $request, EntityManagerInterface $em)
{
    $form = $this->createForm(RegisterType::class);
    $form->handleRequest($request);

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

        $form_data = $form->getData();
        //process form inputs

        ...

    }

    $data['form'] = $form->createView();
    return $this->render('blog/register_template.html.twig', $data);
}
```

# Symfony Form generation 2

---

```
<?php
namespace App\Form;

use App\Form\Model\UserRegistrationFormModel;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\Form\Extension\Core\Type\TextType;
use Symfony\Component\Form\Extension\Core\Type>PasswordType;
use Symfony\Component\Form\Extension\Core\Type\EmailType;
use Symfony\Component\OptionsResolver\OptionsResolver;
use Symfony\Component\Validator\Constraints\NotBlank;

class RegisterType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('name', TextType::class, ['required' => false])
            ->add('email', EmailType::class, ['required' => false])
            ->add('password', PasswordType::class, ['required' => false, 'label' => 'Password',
'constraints' => [new NotBlank(['message' => 'Password must not be empty.'])])
            ->add('passconf', PasswordType::class, ['required' => false, 'label' => 'Password
confirmation'])
        ;
    }

    public function configureOptions(OptionsResolver $resolver)
    {
        $resolver->setDefaults([
            'data_class' => UserRegistrationFormModel::class
        ]);
    }
}
```

# Symfony Form generation 3

---

```
<div class="w3-padding-large">
  <h2 style="text-align:center;">Sign up</h2>
  {{ form_start(form, {'action': path('register'), 'method': 'POST', 'attr': {'class': 'w3-form w3-card-4'}}) }}
  {{ form_row(form.name, {'attr': {'class': 'w3-input'}}) }}
  {{ form_row(form.email, {'attr': {'class': 'w3-input'}}) }}
  {{ form_row(form.password, {'attr': {'class': 'w3-input'}}) }}
  {{ form_row(form.passconf, {'attr': {'class': 'w3-input'}}) }}
  <h3 style="text-align:center;">
    <input class="w3-btn w3-blue w3-round-large" name="submit" type="submit" value="Go" />
    <input class="w3-btn w3-red w3-round-large" name="reset" type="reset" value="Clear" />
  </h3>
  {{ form_end(form) }}
</div>
```

# Symfony Recipes (3rd party packages)



**Symfony Flex** is the way to manage Symfony applications.

It is based on **Symfony Recipes**, which are a set of automated instructions to integrate third-party packages into Symfony applications.

This page lists these great building blocks for your Symfony applications.

## Learn More

- [Using recipes in your Symfony applications](#)
- [Create a recipe for a public package](#)
- Source code for [official recipes](#), [community recipes](#) and [Symfony Flex itself](#)

## All Symfony Recipes available

ad3n/ratchet-bundle

contrib

[Package details](#) [Recipe](#)

adback/adback-sdk-php-symfony

contrib

[Package details](#) [Recipe](#)

adlarge/fixtures-documentation-bundle

contrib

[Package details](#) [Recipe](#)

ajardin/docker-symfony

contrib

[Package details](#) [Recipe](#)

akondas/symfony-consul-bundle

contrib

[Package details](#) [Recipe](#)

alexandret/evc-bundle

contrib

[Package details](#) [Recipe](#)

algolia/algolia-search-bundle

contrib

[Package details](#) [Recipe](#)

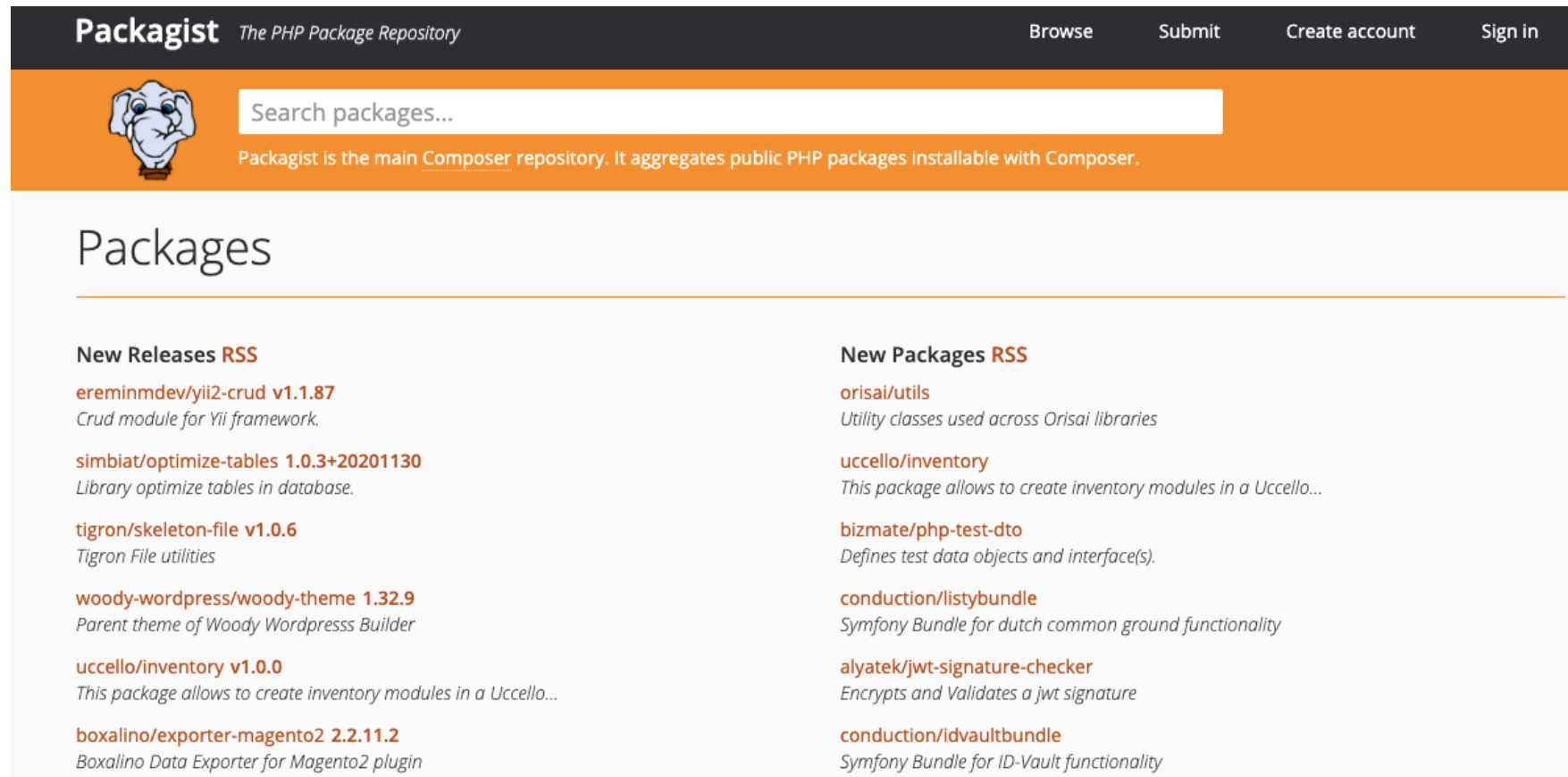
algolia/search-bundle

contrib

[Package details](#) [Recipe](#)

<https://flex.symfony.com/>

# PHP packages



The screenshot shows the Packagist website. At the top, there is a dark navigation bar with the Packagist logo and tagline 'The PHP Package Repository' on the left, and links for 'Browse', 'Submit', 'Create account', and 'Sign in' on the right. Below this is an orange header bar containing a search input field with the placeholder text 'Search packages...' and a small bulldog icon. Underneath the search bar, a line of text states: 'Packagist is the main Composer repository. It aggregates public PHP packages installable with Composer.'

## Packages

---

### New Releases [RSS](#)

- ereminmdev/yii2-crud v1.1.87**  
*Crud module for Yii framework.*
- simbiat/optimize-tables 1.0.3+20201130**  
*Library optimize tables in database.*
- tigron/skeleton-file v1.0.6**  
*Tigron File utilities*
- woody-wordpress/woody-theme 1.32.9**  
*Parent theme of Woody Wordpress Builder*
- ucello/inventory v1.0.0**  
*This package allows to create inventory modules in a Uccello...*
- boxalino/exporter-magento2 2.2.11.2**  
*Boxalino Data Exporter for Magento2 plugin*

### New Packages [RSS](#)

- orisai/utills**  
*Utility classes used across Orisai libraries*
- ucello/inventory**  
*This package allows to create inventory modules in a Uccello...*
- bizmate/php-test-dto**  
*Defines test data objects and interface(s).*
- conduction/listybundle**  
*Symfony Bundle for dutch common ground functionality*
- alyatek/jwt-signature-checker**  
*Encrypts and Validates a jwt signature*
- conduction/idvaultbundle**  
*Symfony Bundle for ID-Vault functionality*

<https://packagist.org/>