コントローラーの作成

コマンドを使って記事のコントローラーを作ります。src/Controller に PostController.php が作られます。

```
php bin/console make:controller PostController
```

次にコントローラーのアノテーション（コメント文）でルーティングができるようにするためのコンポーネントをインストールします。

```
composer require annotations
```

そしてコントローラーに記事一覧を表示するためのメソッドを追加します。冒頭の use でルーティング関連のクラスと、先ほど作成したエンティティ、フォームクラスを読み込んでいます。

```php
<?php

namespace App\Controller;

use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use App\Entity\Post;
use App\Form\PostType;

class PostController extends Controller
{
    /**
     * @Route("/post", methods="GET", name="post")
     */
    public function index()
    {
        $entityManager = $this->getDoctrine()->getManager();

        $posts = $this->getDoctrine()
            ->getRepository(Post::class)
            ->findAll();

        return $this->render('post/index.html.twig', [
            'posts' => $posts
        ]);
    }
}
```

EntityManager はデータベースを読み書きする際に利用し、getRepository() で対象のエンティティを指定し、findAll() ですべて取得しています。これは「SELECT * FROM posts」と同じ意味です。
取得したデータは render() の箇所でテンプレートに渡しています。

次はコントローラーに新規作成処理のメソッド new() を追加します。

```php
/**
 * @Route("/post/new", methods={"GET", "POST"}, name="post_new")
 */
public function new(Request $request)
{
```

```
        $post = new Post();

        $post->setCreatedAt(new \DateTime());

        $form = $this->createForm(PostType::class, $post);
        $form->handleRequest($request);

        if ($form->isSubmitted() && $form->isValid()) {
            $entityManager = $this->getDoctrine()->getManager();
            $entityManager->persist($post);
            $entityManager->flush();

            return $this->redirectToRoute('post');
        }

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

新規投稿をする際、投稿日時の箇所にはあらかじめ現在の日時が入力されている状態にするため、setCreatedAt() で DateTime オブジェクトをセットしてあります。

isSubmitted() で投稿処理が行われていることを確かめ、isValid() で入力値の検証が通れば保存を行います。persist() で EntityManager に保存した後、flush() でデータベースに書き込みます。投稿が終わったら一覧ページにジャンプするようにしてあります。これは更新ボタンで再投稿されるのを防ぐためです。

残りのメソッドも追加します。編集は新規作成と概ね同じですが削除ボタンを表示するための処理を追加してあります。

```
    /**
     * @Route("/post/{id}/edit", methods={"GET", "POST"}, name="post_update")
     */
    public function update($id, Request $request)
    {
        $post = $this->getDoctrine()
            ->getRepository(Post::class)
            ->find($id);

        if (!$post) {
            throw $this->createNotFoundException(
                'No post found for id ' . $id
            );
        }

        $form = $this->createForm(PostType::class, $post);
        $form->handleRequest($request);

        if ($form->isSubmitted() && $form->isValid()) {
            $entityManager = $this->getDoctrine()->getManager();
            $entityManager->persist($post);
            $entityManager->flush();

            return $this->redirectToRoute('post');
        }

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

        return $this->render('post/edit.html.twig', [
            'post' => $post,
```

```
            'form' => $form->createView(),
            'deleteForm' => $deleteForm->createView()
        ]);
    }

    /**
     * @Route("/post/{id}", methods="GET", name="post_show")
     */
    public function show($id)
    {
        $post = $this->getDoctrine()
            ->getRepository(Post::class)
            ->find($id);

        if (!$post) {
            throw $this->createNotFoundException(
                'No post found for id ' . $id
            );
        }

        return $this->render('post/show.html.twig', [
            'post' => $post
        ]);
    }

    /**
     * @Route("/post/{id}", methods="DELETE", name="post_delete")
     */
    public function delete($id, Request $request)
    {
        $post = $this->getDoctrine()
            ->getRepository(Post::class)
            ->find($id);

        if (!$post) {
            throw $this->createNotFoundException(
                'No post found for id ' . $id
            );
        }

        $form = $this->createDeleteForm($id);
        $form->handleRequest($request);

        if ($form->isSubmitted() && $form->isValid()) {
            $entityManager = $this->getDoctrine()->getManager();
            $entityManager->remove($post);
            $entityManager->flush();
        }

        return $this->redirectToRoute('post');
    }

    private function createDeleteForm($id)
    {
        return $this->createFormBuilder()
            ->setAction($this->generateUrl('post_delete', ['id' => $id]))
            ->setMethod('DELETE')
            ->getForm();
    }
```

find() は一件のみ取得する際に使います。存在しない場合は適切にエラーを表示するようにします。