Symfony の基本

コースの詳細：

- **日付**: 2025年6月
- **Symfony**:
  - CLI 5.12.0
  - Symfony 7.3.0
- **Node**: 22.13.0
- **パッケージマネージャー**:
  - npm 11.3.0
  - composer 2.8.9
- **OS**: win32 x64
- **参考元**: https://www.youtube.com/watch?v=nnIBXYGdXHk&list=PLI3CtU4THqPawV0hRF8Qqn0RVEHSjYgfy

Symfony はフランスの SensioLabs によって開発された PHP のバックエンドフレームワークです。

バックエンド開発者の繰り返し作業を効率的に処理し、開発を簡略化してくれます。

最近では SPA ではないものの、フロントエンド機能との統合も進んでいます。

まずは基本を学び、より深く学びたい方は公式ドキュメントを参照してください：

https://symfony.com

1. インストール

1. PHP

Symfony は PHP フレームワークなので、PC に PHP がインストールされている必要があります。

php.ini ファイルで以下の拡張機能が有効になっていることを確認してください：

```
extension=fileinfo
extension=intl
extension=pdo_mysql
; その他必要に応じて
```

2. Composer

Symfony の動作には Composer（PHP のパッケージマネージャ）が必要です。Composer 自体も PHP を利用します。

3. Symfony CLI

以下は Windows 向けの説明です。

Symfony には専用の CLI（コマンドラインインターフェース）があります。
手動でインストールし、環境変数にパスを追加することもできます。

しかし、ここでは **SCOOP** を使って簡単にインストールしましょう：

https://scoop.sh/

PowerShell で以下のコマンドを使用します。初回の場合は 1 行目のコマンドも実行してください：

```
> Set-ExecutionPolicy RemoteSigned -Scope CurrentUser
> irm get.scoop.sh | iex
```

これで scoop が使えるようになります。
Symfony CLI をインストールしましょう：

```
scoop install symfony-cli
```

Symfony がインストールされました。次はプロジェクトを作成しましょう。

4. 推奨 VSCode 拡張機能と設定

- PHP INTELEPHENSE
- PHP NAMESPACE RESOLVER
- TWIG LANGUAGE 2
- PHP IntelliSense（任意）

設定では：

- Emmet Include Languages に Twig と HTML の関連付けを追加してください。

2. プロジェクトの作成

プロジェクトを作りたい場所で以下のコマンドを実行：

```
# 通常の Web アプリケーション用
symfony new プロジェクト名 --version="7.3.*" --webapp

# マイクロサービス用
symfony new プロジェクト名 --version="7.3.*"
```

- `--webapp` は任意ですが、一般的な Web アプリに必要なパッケージを自動で追加します。
- `--version` ではインストールしたい Symfony のバージョンを指定します。

プロジェクトの作成が完了しました。

Docker や他のローカルサーバーは不要で、Symfony 独自のサーバーが用意されています（PHP のインストールは必要です）。

```
symfony serve
```

デフォルトで以下のアドレスでアクセスできます：

- http://127.0.0.1:8000

開発中、まだホームページが設定されていなければ、ドキュメントへのリンク付きの初期ページが表示されます。

初回起動時はキャッシュの準備に時間がかかるため、読み込みが遅いことがあります。
また、キャッシュをクリアした後も同様に遅くなることがあります。

3. Symfony プロジェクトの構造

プロジェクトの構成を確認しましょう。

- `.git`（VSCodeでは非表示）: デフォルトで git init 済み
- `assets`: JavaScript や CSS のファイル
- `bin`: Symfony の内部コマンド群（変更不要）
- `config`: 使用パッケージの設定ファイル（必要な場合のみ編集）
- `migrations`: DB スキーマの更新が記録される（最初は空）
- `public`: Web サーバーのルート（`index.php` を含む）
- `src`: 開発のメイン領域
  - `Controller`: コントローラーファイル
  - `Entity`: データベースのテーブルを表すクラス
  - `Repository`: モデルクラス
  - `Kernel.php`: Symfony のコアファイル（編集不要）
- `templates`: Twig によるビュー
  - `base.html.twig`: すべてのページの共通レイアウト
- `tests`: テストコード用（今回は使用しません）
- `translations`: 多言語対応（今回は使用しません）
- `var`: ログやキャッシュなどの一時データ
- `vendor`: Composer により導入されたライブラリ
- `.editorconfig`: エディター用設定ファイル
- `.env`: 環境変数（開発環境や DB 設定など）

`.env` は git にコミットされます。
秘密情報の保護のため、`.env.local` を作成し、`.env` の値を上書きするのが推奨されます。

- `.env.dev`: 開発環境用
- `.env.test`: テスト環境用
- `.gitignore`: 一般的な無視ファイルが設定済み
- `compose.override.yaml` / `compose.yaml`: Docker 設定ファイル（DB やメールサーバーの起動に使用）
- `composer.json`: 依存パッケージの一覧

- `composer.lock`: 各パッケージのバージョン詳細
- `importmap.php`: JavaScript 依存管理
- `phpunit.xml.dist`: PHPUnit 用設定ファイル
- `symfony.lock`: Symfony のライブラリ構成記録

`compose.yaml` に次のような行があります：

`POSTGRES_PASSWORD: \${POSTGRES_PASSWORD:-!ChangeMe!}`
デフォルトパスワードを設定してから、以下のコマンドで Docker を起動してください：

```
docker compose up -d
```

Docker がインストールされ、実行中であることを確認してください。