Symfony Basics

Course details:

- **Date**: 06/2025
- **Symfony**:
  - CLI 5.12.0
  - Symfony 7.3.0
- **Node**: 22.13.0
- **Package Manager**:
  - npm 11.3.0
  - composer 2.8.9
- **OS**: win32 x64
- **Inspiration**: https://www.youtube.com/watch?v=nnIBXYGdXHk&list=PLI3CtU4THqPawV0hRF8Qqn0RVEHSjYgfy

Symfony is a PHP back-end framework developed by the French company "SensioLabs".

It simplifies the work of back-end developers by optimizing the handling of repetitive back-end tasks.

Over time, it has gradually supported more front-end features, though it's not quite a SPA framework (yet).

We will cover the basics together, but for further details, the full documentation is available on the official website:

https://symfony.com

1. Installation

1. PHP

Since Symfony is a PHP framework, you need PHP installed on your system.

Make sure the following extensions are enabled in your **php.ini** file:

```
extension=fileinfo
extension=intl
extension=pdo_mysql
; and possibly others.
```

2. Composer

Symfony relies on the PHP package manager "Composer", which itself requires a working PHP installation.

3. Symfony CLI

The following applies for Windows installation.

Symfony uses its own CLI (Command Line Interface).
You can install it manually and add it to your system's environment variables.

But to simplify things, we'll install **SCOOP** — a command-line installer for Windows:

https://scoop.sh/

You may need to use the second command below in PowerShell, and possibly the first one if it's your first time executing scripts:

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

You can now use scoop from the command line.
Let's install Symfony using scoop:

```
scoop install symfony-cli
```

Symfony is now installed. Let's create a new project.

4. Useful VSCode Extensions and Settings

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

And in settings:

- Enable Emmet for Twig under "Emmet Include Languages".

2. Creating a new project

To create a new project, navigate to the folder where you want it and run:

```
# For a typical web application
symfony new project_name --version="7.3.*" --webapp

# For a microservice
symfony new project_name --version="7.3.*"
```

- `--webapp` is optional but includes the most useful packages for web development.
- `--version` specifies the Symfony version you want to install.

Your Symfony project is now created.

To start it, no need for Docker or another local server. Symfony includes its own local server (you still need PHP installed).

```
symfony serve
```

By default, the app will be available at:

- http://127.0.0.1:8000

During development, if no homepage is defined yet, a default page with documentation links will be shown.

The first time you launch the project, Symfony caches a lot of things, so the initial load might be slow.
This also applies every time you clear the cache.

3. Symfony Project Structure

Let's explore the contents of a typical Symfony project.

- `.git` (invisible in VSCode): Symfony projects initialize a git repository by default.
- `assets`: contains all the JS and CSS of the project.
- `bin`: contains Symfony's internal commands — no need to modify.
- `config`: holds configuration files for all used packages — only change if needed.
- `migrations`: starts empty but will contain all DB schema updates.
- `public`: the web root directory (contains `index.php`).
- `src`: your main working directory:
  - `Controller`: where all your controllers go.
  - `Entity`: classes that represent your DB tables.
  - `Repository`: these are your models.
  - `Kernel.php`: the core of Symfony — do not modify.
- `templates`: holds all views (Twig templates).
  - `base.html.twig`: the base layout for all pages.
- `tests`: contains Symfony test files — not needed for now.
- `translations`: for multilingual support — not used in this course.
- `var`: stores logs (successes/failures of the app).
- `vendor`: contains all third-party packages (via Composer).
- `.editorconfig`: editor settings file.
- `.env`: stores environment variables:

    - working environment
    - DB connection
    - mailer settings
    - secret keys, etc.

  This file is committed to git. To avoid exposing secrets, create a `.env.local` file (ignored by git) which overrides `.env`.

- `.env.dev`: used only in development mode.
- `.env.test`: used in test environments.
- `.gitignore`: already configured with common ignores.
- `compose.override.yaml` and `compose.yaml`: Docker setup (e.g., DB, mail server).
- `composer.json`: list of project dependencies.
- `composer.lock`: exact versions of installed packages.
- `importmap.php`: defines JS dependencies.
- `phpunit.xml.dist`: PHPUnit configuration.
- `symfony.lock`: tracks Symfony-related libraries.

If you open **`compose.yaml`**, you'll see the line:
`POSTGRES_PASSWORD: \${POSTGRES_PASSWORD:-!ChangeMe!}`
Take time to set a default password, then run:

```
docker compose up -d
```

Make sure Docker is installed and running.