Seminario 1: Symfony “Hands-on” - Mi Primer Blog Tecnológico

Asignatura: Desarrollo en Entorno Servidor **Tecnología:** Symfony 8, Docker, Twig, Doctrine **Autor:** Manuel Prieto Macias

---

1. Objetivos del Seminario

Este seminario está diseñado para consolidar los conocimientos adquiridos en los **Temas 6 y 7**. Al finalizar, habrás construido desde cero una aplicación de blog funcional, poniendo en práctica el flujo de trabajo profesional con Symfony.

Los objetivos clave son:

- **Entorno Docker**: Levantar un entorno de desarrollo autocontenido con Docker.
- **Rutas y Controladores**: Crear las rutas y la lógica de controlador para las páginas principales del blog.
- **Vistas con Twig**: Utilizar el motor de plantillas Twig para renderizar el frontend, aplicando herencia y reutilización.
- **Introducción a Doctrine**: Crear nuestra primera entidad (`Post`) y su repositorio para interactuar con la base de datos.
- **Buenas Prácticas**: Fomentar el uso de `maker-bundle` y la depuración con el Profiler.

2. El Proyecto: “ZenPaw Tech”

Vamos a crear “ZenPaw Tech”, un blog minimalista para publicar artículos sobre desarrollo de software. La idea es centrarnos en la mecánica de Symfony, no en un diseño complejo.

Funcionalidades a Implementar:

1. **Página Principal**: Un listado de todos los artículos publicados, mostrando título y un pequeño extracto.
2. **Página de Artículo**: La vista detallada de un único artículo, mostrando su contenido completo.
3. **Página de "Acerca de"**: Una página estática simple.

3. Preparación del Entorno (Tu Tarea)

Tu primer paso es configurar el proyecto y el entorno de desarrollo. Deberás:

1. **Crear un nuevo proyecto Symfony**: Usa el comando `symfony new` con la opción `--webapp` para incluir todo lo necesario.
2. **Levantar el entorno Docker**: Revisa el `docker-compose.yaml` generado. ¿Qué servicios contiene? Usa `docker compose up -d` para iniciarlos.
3. **Crear la Base de Datos**: Los contenedores estarán corriendo, pero la base de datos estará vacía. Investiga en la documentación de DoctrineBundle qué comando de `symfony console` necesitas para crear la base de datos definida en tu `.env`.

Al finalizar, deberías poder acceder a `http://localhost` y ver la página de bienvenida de Symfony.

4. Creando la Entidad `Post` (Tu Tarea)

Una entidad es una clase PHP que representa una tabla en tu base de datos. Tu tarea es crear la entidad `Post` que almacenará los artículos del blog.

1. **Usa el MakerBundle**: Ejecuta el comando `make:entity`.
2. **Define las propiedades**: La aplicación necesita almacenar la siguiente información para cada post. Deberás elegir el tipo de dato (`string`, `text`, `datetime_immutable`, etc.) más adecuado para cada una:
    - `title` (no puede ser nulo)
    - `slug` (para la URL, no puede ser nulo)

- `summary` (un resumen corto)
- `content` (el cuerpo del artículo)
- `publishedAt` (la fecha de publicación)

3. **Aplica la Migración**: Una vez definida la entidad, tu base de datos aún no tiene la tabla `post`. ¿Qué dos comandos de `symfony console` necesitas ejecutar para (1) crear el archivo de migración y (2) aplicarlo a la base de datos? Consulta la documentación de Doctrine Migrations.

5. Rutas y Controladores (Tu Tarea)

Ahora crearás el corazón de la aplicación: el controlador que gestionará las peticiones del blog.

1. **Crea el `BlogController`**: Usa `make:controller` para generar el archivo.
2. **Implementa la Página Principal (`/`)**:
   - Modifica el método `index()` que se ha generado.
   - **Objetivo**: Obtener todos los posts de la base de datos y pasarlos a una plantilla Twig.
   - **Pista**: Necesitarás inyectar el `PostRepository` en el método. Investiga cómo usar su método `findBy()` para ordenar los resultados por fecha de publicación de forma descendente.
3. **Implementa la Página de Detalle (`/post/{slug}`)**:
   - Añade un nuevo método `show()` a tu `BlogController`.
   - **Objetivo**: Recibir un `slug` desde la URL, buscar el `Post` correspondiente y pasarlo a una plantilla.
   - **Manejo de errores**: ¿Qué ocurre si se accede a un slug que no existe? Debes lanzar una `NotFoundHttpException`. Investiga cómo hacerlo desde `AbstractController`.
   - **Píldora de Investigación**: El código anterior funciona, pero Symfony puede hacerlo aún más simple. Investiga sobre los **ParamConverters**. ¿Cómo

podrías reescribir la firma del método `show()` para que Symfony encuentre el post automáticamente por el `{slug}`?

6. Vistas con Twig (Tu Tarea)

Es hora de dar forma visual a los datos.

1. **Crea una Plantilla Base**: Modifica `templates/base.html.twig`. Debe contener la estructura HTML común (header, nav, main, footer) y los `{% block %}` principales (`title`, `body`). Para el estilo, puedes enlazar una librería CSS minimalista como Simple.css o Pico.css.
2. **Crea la Vista del Listado**: Modifica `templates/blog/index.html.twig`.
    - Debe extender de tu `base.html.twig`.
    - Usa un bucle `{% for %}` para iterar sobre la variable `posts` que le pasas desde el controlador.
    - Dentro del bucle, muestra el título de cada post como un enlace a su página de detalle. Usa la función `path()` para generar la URL dinámicamente.
    - Muestra también el `summary` y la fecha de publicación (formateada con el filtro `date`).
3. **Crea la Vista de Detalle**: Crea el archivo `templates/blog/show.html.twig`.
    - Debe extender de tu `base.html.twig`.
    - Muestra el `title`, `publishedAt` y el `content` del post.
    - **Píldora de Investigación**: El contenido del post puede tener saltos de línea. Investiga el filtro `nl2br` de Twig. ¿Qué pasaría si el contenido tuviera HTML? Investiga sobre el filtro `raw` y por qué debe usarse con extremo cuidado.

7. Añadiendo Contenido de Prueba (Tu Tarea)

Para poder ver algo en tu blog, necesitas datos. La forma profesional de añadir datos de prueba es mediante **Fixtures**.

1. **Instala el bundle de Fixtures**: Usa `composer require` para añadir `doctrine/doctrine-fixtures-bundle` a tu proyecto (recuerda ejecutarlo dentro del contenedor de PHP si usas Docker).
2. **Crea una clase de Fixtures**: Usa `make:fixtures`.
3. **Escribe la lógica de carga**: Dentro del método `load()`, crea 2 o 3 objetos `Post`, asígnales datos de prueba con sus `setters` y usa el `ObjectManager` para persistirlos.
4. **Carga los datos**: Ejecuta el comando de `symfony console` para cargar las fixtures. Usa la opción `--purge` para limpiar la base de datos antes.

Si todo ha ido bien, al visitar la página principal de tu proyecto, ¡verás los artículos que acabas de crear!

8. Siguientes Pasos (Opcional)

- Implementa la página “Acerca de”.
- Añade más campos a la entidad `Post`, como un `author` (autor).
- Crea un `CategoryController` y una entidad `Category` para organizar tus posts (requerirá investigar sobre Relaciones Doctrine ManyToOne/ManyToMany).