Curso Symfony 4 - Clase 3

En la clase pasada vimos que era un ORM y cuáles eran sus ventajas y desventajas. Hoy comenzaremos a trabajar y aprender sobre Doctrine que es uno de los ORM más famosos que existe para PHP.
Comenzaremos con algunos conceptos básicos:

¿Qué es Doctrine?

Doctrine es un mapeador relacional de objetos para PHP 7.1+ que proporciona persistencia transparente para objetos PHP. Utiliza el patrón Data Mapper en el corazón, con el objetivo de una separación completa de su dominio / lógica comercial de la persistencia en un sistema de administración de base de datos relacional.

El beneficio de Doctrine para el programador es la capacidad de enfocarse en la lógica empresarial orientada a objetos y preocuparse por la persistencia sólo como un problema secundario. Esto no significa que la persistencia sea minimizada por Doctrine, sin embargo, creemos que hay beneficios considerables para la programación orientada a objetos si la persistencia y las entidades se mantienen separadas.

¿Qué son las entidades?

Las entidades son objetos PHP que se pueden identificar en muchas solicitudes mediante un identificador único o clave principal. Estas clases no necesitan extender ninguna interfaz o clase base abstracta. Es como una clase tradicional de PHP, solo que contiene anotaciones especiales que referencian a las relaciones de la entidad en la base de datos y sus campos. Una entidad contiene propiedades persistentes. Una propiedad persistente es una variable de instancia de la entidad que se guarda

y recupera de la base de datos mediante las capacidades de mapeo de datos de Doctrine.

Ejemplo de una entidad:

```php
<?php
// src/Product.php
class Product
{
    /**
     * @var int
     */
    protected $id;
    /**
     * @var string
     */
    protected $name;

    public function getId()
    {
        return $this->id;
    }

    public function getName()
    {
        return $this->name;
    }

    public function setName($name)
    {
        $this->name = $name;
    }
}
```

Entity Manager:

La interfaz pública de Doctrine es a través de Entity Manager. Esta clase proporciona puntos de acceso a la gestión completa del ciclo de vida de sus entidades y transforma las entidades desde y hacia la persistencia.

Para trabajar con esta clase debemos instanciarla, para instanciarla debemos inyectar la dependencia en donde nosotros estemos trabajando. Generalmente es en los controladores o los repositorios que ya veremos más adelante que son y para qué sirven.

Repositorios:

Los repositorios son clases que contienen el código que efectivamente interactúa con la base de datos. Estos repositorios sirven para separar el código de una forma lógica y estructurada. Por ejemplo si nosotros tuviéramos un sistema o proyecto que se encarga de la compra y venta de autos, tendríamos un repositorio llamado Auto o de la forma que vamos a llamarnos en symfony que es: AutoRepository. Además tendríamos un repositorio más para los contratos de la venta de los autos y un repositorio para los empleados que se encargan de hacer las ventas de los autos. Para interactuar con los repositorios lo haremos a través del Entity Manager y podremos acceder a los métodos que nosotros definamos.
El código que define estos métodos que se encuentran dentro del repositorio que nosotros vamos a definir, tiende a contener algo de lógica pero se centra en la consulta SQL o DQL que efectivamente va a ejecutarse en la base de datos.

Association Mapping:

Association mapping o mapeo de asociaciones es una herramienta que nos proveen los ORM y en este caso Doctrine. Esta herramienta nos facilita la representación de las relaciones entre tablas en la base de datos. Por ejemplo si nosotros tenemos la relación 1 a muchos como podría ser que un empleado tiene muchos recibos de sueldo, esta relación que se presenta en la base de datos, es representada en nuestro modelo de objetos o entidades, a través de una anotación especial.
Existen varias anotaciones dependiendo el tipo de relación que queremos implementar en nuestro modelo. Algunas de ellas son:

Un consejo para trabajar con relaciones es leer la relación de izquierda a derecha, donde la palabra de la izquierda se refiere a la entidad actual. Por ejemplo:

- OneToMany: una instancia de la entidad actual tiene muchas instancias (referencias) a la entidad referida.
- ManyToOne: muchas instancias de la entidad actual se refieren a una instancia de la entidad referida.

- OneToOne: una instancia de la entidad actual se refiere a una instancia de la entidad referida.

DQL:

DQL significa Doctrine Query Language y es un derivado del Object Query Language que es muy similar al Hibernate Query Language (HQL) o al Java Persistence Query Language (JPQL).

En esencia, DQL proporciona potentes capacidades de consulta sobre su modelo de objetos. Imagina tener todos tus objetos en algún lugar de almacenamiento (como una base de datos de objetos). Al escribir consultas DQL, es como consultar ese almacenamiento para elegir un determinado subconjunto de sus objetos.

DQL como lenguaje de consulta tiene instrucciones SELECT, UPDATE y DELETE que se asignan a sus correspondientes tipos de instrucciones SQL. Las declaraciones INSERT no están permitidas en DQL, porque las entidades y sus relaciones deben introducirse en el contexto de persistencia, EntityManager->persist(), para garantizar la coherencia de su modelo de objetos.

Las declaraciones DQL SELECT son una forma muy poderosa de recuperar partes de su modelo de dominio a las que no se puede acceder a través de asociaciones. Además, le permiten recuperar entidades y sus asociaciones, o relaciones, en una única declaración de selección de SQL, lo que puede marcar una gran diferencia en el rendimiento en comparación con el uso de varias consultas.