Mapeo Objeto Relacional ORM

El **mapeo objeto-relacional** (ORM, Object-Relational Mapping) es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia.

Mapeo Objeto Relacional ORM

Se asocian los **elementos de la base de datos** con los **objetos de la aplicación**. Por tanto, la gestión de datos no se realizará directamente sobre la base de datos, sino sobre una serie de clases que la replican.

Un ORM facilita el trabajo al programador y permite reusabilidad de código.

Para hacernos una idea....

Pensemos que ***Departamento*** es una clase que contiene las propiedades id y presupuesto y los métodos necesarios para realizar las operaciones en la base de datos.

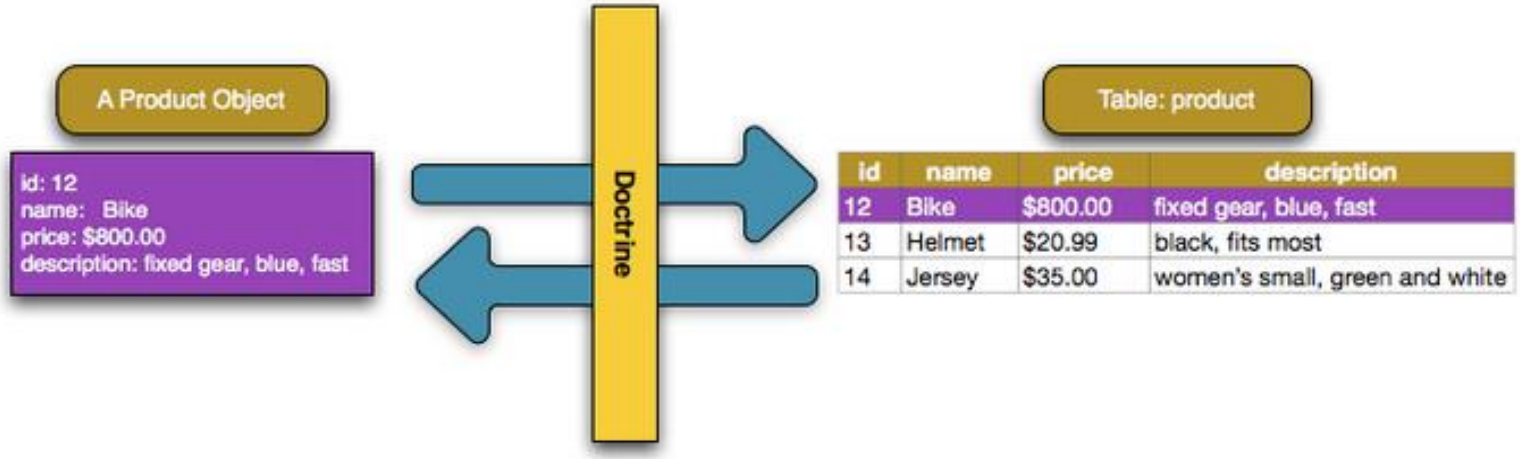
Se pueden actualizar los departamentos modificando las propiedades del objeto y guardándolo:

```php
//El método find() construye y ejecuta la consulta SQL
$dep = $entityManager->find("Departamento", 3);

//Actualizamos la propiedad presupuesto
$dep->setPresupuesto(70000);
$entityManager->flush();
```

Existen **herramientas para realizar el ORM**. Nosotros instalaremos **Doctrine**:

- Es un ***mapeador objeto-relacional*** (ORM) escrito en PHP que proporciona una capa de persistencia para objetos PHP. Esta capa se sitúa justo encima del SGBD.
- Necesita ***poca configuración para empezar un proyecto***: puede generar clases a partir de una base de datos existente y después el programador puede especificar relaciones y añadir funcionalidad extra a las clases autogeneradas. No es necesario generar o mantener complejos esquemas XML de base de datos como en otros frameworks.
- ***Permite escribir consultas*** de base de datos utiliza un dialecto de SQL denominado ***DQL*** (Doctrine Query Language) inspirado en Hibernate (Java).



Para instalar Doctrine necesito instalar **Composer:**

sistema de gestión de paquetes para programar en PHP (similar a Maven) el cual provee los formatos estándar necesarios para manejar dependencias y librerías de PHP.

Composer trabaja e instala dependencias o librerías desde la línea de comandos. Permite al usuario instalar las aplicaciones PHP que estén disponibles en "Packagist (packagist.org)". También dispone de capacidad de auto-carga para las librerías necesarias que se especifiquen en la información de arranque (require).

Packagist es el repositorio central de Composer: lugar del que se obtienen los paquetes. Packagist aspira a convertirse en el repositorio central que utilicen todos los usuarios de Composer. Por tanto, puedes incluir en el require de tu archivo composer.json cualquier paquete disponible en Packagist.

Si accedes al sitio web de Packagist, podrás encontrar miles de paquetes.

Instalación de Composer

Descargo de la página: https://getcomposer.org/

Download / Composer-setup.exe

Instalo Composer con los pasos:

- Ejecuto **composer-setup.exe** /
- Selecciono **Developer mode** /
- **Ruta** por defecto (mi usuario) /
- Compruebo la ruta **c:\xampp\php\php.exe** /
- Si no tengo **proxy** me salto el paso /
- **Resumen** de la instalación
- **Leer la información**: abrir cmd de Windows o terminal desde VS Code y ejecuto composer para ver que se ha instalado correctamente.

Comandos Composer

Cuando instalo paquetes usando Composer, creará entre otros, el fichero ***composer.json***.

Composer ofrece varios parámetros incluyendo:

- **`init`**: crea *composer.json* e información a nuestro proyecto.
- **`require`**: añade el parámetro de la librería al archivo *composer.json* y lo instala.
- **`install`**: instala todas las librerías de *composer.json*. Es el comando que se usa para descargar todas las dependencias PHP desde el repositorio.
- **`update`**: actualiza las librerías de *composer.json* de acuerdo a las versiones permitidas que se señalen.
- **`remove`**: desinstala una librería y la elimina de 'composer.json'.

Definición de librerías: ejemplo de *composer.json* generado por el comando: ***`composer require monolog/monolog`*** *(*https://getcomposer.org/doc/01-basic-usage.md#composer-json-project-setup*)*

```
{
    "require": {
        "monolog/monolog": "1.2.*"
    }
}
```

(***) Tarea Composer

- Define ORM busca información de qué es Composer y qué es Doctrine. Añade enlaces a los sitios web oficiales.
- Elabora un manual de instalación de Composer con capturas de pantalla y explicación de las mismas. Consulta la documentación: https://getcomposer.org/doc/
- Busca información sobre los siguientes comandos y explica para qué sirven:
  - init, require, install, update, remove
  - help, search, show
  - validate, depends, status, status -v, self-update
  - create-project