INGENIERÍA UNIVERSIDAD DE LOS ANDES MÉRIDA VENEZUELA | DEPARTAMENTO DE COMPUTACIÓN ESCUELA DE SISTEMAS

Laboratorio de Base de Datos
Práctica Nro. 9, Frameworks Web y ORM

Prof. Solazver Solé
Preps. Victor Albornoz, Yenifer Ramírez
Semestre B-2018

1. Framework

Es un conjunto de archivos y directorios, que incorporan funcionalidades ya desarrolladas y probadas que se implementa en un lenguaje de programación. Su objetivo principal es facilitar el desarrollo de aplicaciones, haciendo que nos centremos en las funcionalidades del usuario, esto gracias a que poseen herramientas de generación de código que evitan las tareas repetitivas para el desarrollador, además de estar constituida de una arquitectura y patrones de diseño ya probados.

1.1. Ventajas

- Uso de patrones de diseño para el desarrollo de la aplicación.
- Casi todos los frameworks utilizan el patrón arquitectónico MVC, el cual es un modelo que se divide en 3 capas:
  - **Modelo:** representa los datos de la aplicación y las reglas de negocio.
  - **Vista:** representa la capa de presentación, lo que se presenta al usuario.
  - **Controlador:** es el encargado de procesar las peticiones de los usuarios y controla el flujo de ejecución del sistema.
- El modelo MVC puede ser implementado sin la necesidad de usar un framework, pero la diferencia radica en que el framework nos obliga a utilizarlo.
- Viene con una estructura predefinida de la aplicación, lo que implica que el programador no necesita plantearse la estructura global de la aplicación.

- Todo el código que es parte del framework es testeado.
- Generalmente, según la antiguedad del framework, este cuenta con un gran comunidad de usuarios, que ayudan para el desarrollo.

1.2. Características

- **Abstracción de URLs y sesiones:** no es necesario manejar directamente las URLs ni las sesiones, ya que el framework se encarga de hacerlo.
- **Acceso a datos:** incluyen herramientas e interfaces necesarias para comunicarse con bases de datos, independientemente del tipo que estemos utilizando.
- **Uso de controladores:** suelen implementar una serie de controladores para la gestión de los eventos y peticiones realizadas a la aplicación.
- **Autentificación y control de acceso:** incluyen mecanismos para la identificación de usuarios mediante el uso de un usuario y contraseña.
- **Internalización:** son mecanismos para poder mostrar la aplicación en todos aquellos idiomas que consideremos oportunos.

1.3. ¿Cómo elegir un framework?

- Conocimientos del equipo.
- Soporte y documentación.
- Existencia de proyectos desarrollados en él.
- Soporte del patrón arquitectónico MVC.
- Seguridad del framework.

1.4. Ejemplos de Frameworks Web

- **Ruby on Rails:** framework MVC basado en Ruby orientado al desarrollo de aplicaciones web.
- **CodeIgniter:** framework basado en PHP liviano y rápido.
- **Django:** framework para Python capaz de crear diseños muy limpios.
- **Zend Framework:** es un framework de código abierto en PHP para desarrollar aplicaciones web y servicios web con PHP 5.
- **Symfony:** completo framework en PHP diseñado para optimizar el desarrollo de las aplicaciones web basado en Modelo-Vista-Controlador.
- **Yii:** framework en PHP basado en componentes.
- **Struts:** herramienta de soporte para el desarrollo de aplicaciones Web bajo el MVC y la plataforma Java EE (Java Enterprise Edition).

1.5. Framework ORM

Básicamente, el framework ORM genera objetos que mapean virtualmente las tablas en una base de datos relacional. Entonces nosotros como programadores, usaríamos estos objetos para interactuar con la base de datos. Así que la idea principal es intentar proteger al programador de tener que escribir código SQL optimizado ya que los objetos generados por ORM se encargan de eso por usted.

1.5.1. Ventajas

- Armonización de los tipos de datos entre el lenguaje orientado a objetos y la base de datos SQL.
- Rapidez en el desarrollo y facilidad para el mantenimiento del código.
- Seguridad de la capa de acceso.
- Abstracción de la base de datos.

1.5.2. Desventajas

- El aprendizaje del lenguaje de la herramienta ORM puede resultar ser complejo.
- Dependiendo de la complejidad de la base datos, puede afectar el rendimiento.

1.5.3. ¿Cuándo usar un ORM?

La decisión sobre usar o no un ORM debería girar en torno a dos ejes fundamentales: la complejidad en términos de Modelo de Entidades de nuestra aplicación y el rendimiento.

1.5.4. Algunos Frameworks ORM

En el mercado podemos encontrar diversas herramientas tanto de pago como de uso libre. Algunos programadores, prefieren invertir tiempo en desarrollar su propia herramienta ORM usando patrones de diseño bien conocidos como son el *Repository* o el *Active Record*.

- Doctrina.
- CakePHP tiene ORM.
- Entity Framework Core.
- Django.

1.5.5. Ejemplo

Digamos que tenemos el siguiente modelo de datos representado en uml:

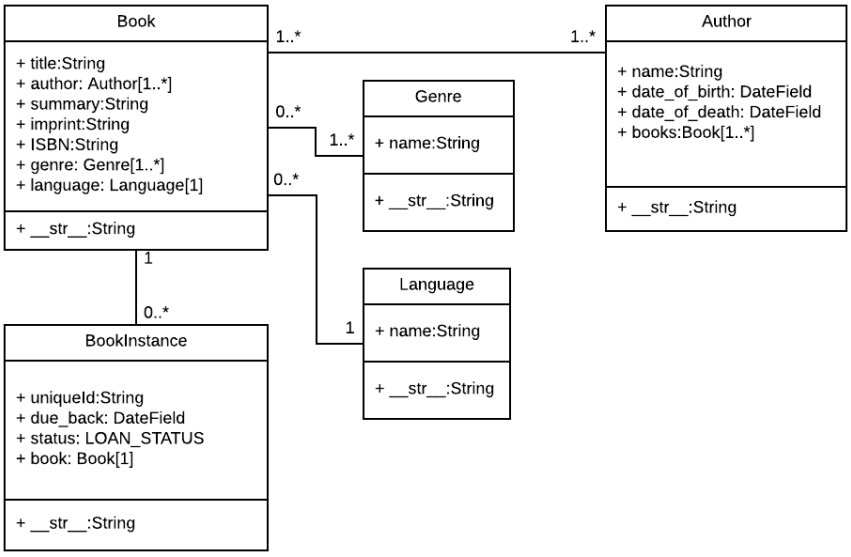

Figura 1: Modelo de datos UML de una biblioteca

De este, obtendremos un aproximado mínimo de 7 tablas, en un modelo relacional:

- Book
- BookInstance
- Genre
- Language
- Author
- Book_Genre
- Book_Author

Con un poco de configuración en cualquier framework ORM se crearían los objetos correspondientes (por ejemplo, book_object, bookinstance_object, genre_object, language_object, author_object) que manejarían toda la interacción de la base de datos. Según esto si nececitamos agregar un nuevo libro a la base de datos, solo tendríamos que usar el objeto del cliente ORM para agregar el nuevo libro. Por ejemplo, podría ser tan simple como llamar al método 'save()' del objeto:

```
book = new book\_object("Algebra de Baldor","1","Libro
de algebra con ejercicios y respuestas","..");
book.save();
```

Nota: lo anterior, por supuesto, es solo un pseudocódigo, principalmente porque la sintaxis varía según el framework ORM elegido.

1.5.6. ORM en Django

Para utilizar ORM en Django debemos crear previamente las clases de nuestro modelo, para ejemplos de clase trabajaremos con el modelo planteado en la sección anterior, en [6] nos muestran la creación de este modelo. Para proceder con el tratado de objetos en Django, trabajeremos por ahora en nuestro shell:

```
(eV)usuario@pc:~$ python manage.py shell
```

- **Creando objetos:** para representar los datos de la tabla de la base de datos en los objetos de Python, Django usa un sistema intuitivo: una clase modelo representa una tabla de la base de datos y una instancia de esa clase representa un registro particular en la tabla de la base de datos.

  Para crear un objeto, se crea una instancia utilizando argumentos de palabras claves para la clase del modelo, luego llamamos a `save()` para guardarlo en la base de datos.

```
>>> a_record = MyModelName(my_field_name="primer_instancia")
>>> a_record.save()
```

- **Actualizar Objetos:** debemos acceder a los campos del objeto usando la sintaxis de puntos y luego cambiar los valores. Tenemos que llamar a `save()` para almacenar los valores modificados en la base de datos.

```
>>> print(a_record.id)
>>> print(a_record.my_field_name)
>>> a_record.my_field_name = 'nueva_instancia'
>>> a_record.save()
```

- **Consultas de Objetos:** podemos buscar registros que coincidan con un cierto criterio usando el atributo *objects* del modelo.

  Podemos obtener todos los registros de un modelo como `QuerySet`, usando `objects.all()` (el `QuerySet` es un objeto iterable, significando que contiene un número de objetos por los que podemos iterar o hacer un bucle).

```
>>> all_books = Book.objects.all()
```

El método de Django `filter()` nos permite filtrar el `QuerySet` devuelto para que coincida un campo de: texto, numérico o con un criterio particular. Por ejemplo, para filtrar libros que contengan la palabra 'wild' en el título y luego contarlos, podemos hacer lo siguiente:

```
>>> wild_books = Book.objects.filter(title___contains='wild')
>>> number_wild_books = Book.objects.filter(title___contains='wild').count()
```

Los campos a buscar y el tipo de coincidencia son definidos en el nombre del parámetro del filtro, usando el formato: `field_name__match_type` (tengamos en cuenta el doble guión bajo entre `title` y `contains` anterior).

En el ejemplo anterior estamos filtrando `title` por un valor sensible a mayúsculas y minúsculas. Podemos hacer otros muchos tipos de coincidencias: `icontains` (no sensible a mayúsculas ni minúsculas), `iexact` (coincidencia exacta no sensible a mayúsculas ni minúsculas), `exact` (coincidencia exacta sensible a mayúsculas y minúsculas) e `in, gt`(mayor que), `startswith`, etc. Podemos ver la lista completa en la documentación de Django.

En algunos casos, necesitaremos filtrar por un campo que define una relación *uno-a-muchos* con otro modelo (por ejemplo, una *ForeignKey*). En estos casos puedes referenciar a campos dentro del modelo relacionado con un doble guión bajo adicional. Así, por ejemplo, para filtrar los libros de un género específico tenemos que referenciar el `name` a través del campo `genre` como se muestra más abajo:

```
>>> books_containing_genre = Book.objects.filter(genre___name___icontains='fiction')
```

Nota: Puedes usar guiones bajos (___) para navegar por tantos niveles de relaciones (`ForeignKey-ManyToManyField`) como queramos. Por ejemplo, un *Book* que tuviera diferentes "*types*", definidos usando una relación adicional "*cover*", podría tener un nombre de parámetro: `type__cover__name__exact='hard'`.

Referencias

[1] Es.wikipedia.org. (2019). Mapeo objeto-relacional. [online] Available at: https://es.wikipedia.org/wiki/Mapeo_objeto-relacional.

[2] Ceac. (2019). Herramientas de mapeo objeto-relacional (ORM). [online] Available at: https://www.ceac.es/blog/herramientas-de-mapeo-objeto-relacional-orm.

[3] KillerPHP.com. (2019). What are ORM Frameworks? - KillerPHP.com. [online] Available at: https://www.killerphp.com/articles/what-are-orm-frameworks.

[4] Documentación web de MDN. (2019). Tutorial Django Parte 3: Uso de modelos. [online] Available at: https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Models.

[5] Eventos.python.org.ar. (2019). [online] Available at: https://eventos.python.org.ar/media/talks/Jugando_con_el_ORM_de_Django_-_PyConAr.pdf.

[6] Riptutorial.com. (2019). Django - Transacciones atómicas | django Tutorial. [online] Available at: https://riptutorial.com/es/django/example/19740/transacciones-atomicas.