Mapeo Objecto-Documento (ODM)

Alejandro Pozo

Andrés Muñoz

Enrique Barra

Contenidos

- Introducción
- ODM
  - Concepto
    - Ejemplo NodeJS Mongo (Driver)
    - Ejemplo NodeJS Mongoose (ODM)
  - Ventajas
  - Desventajas

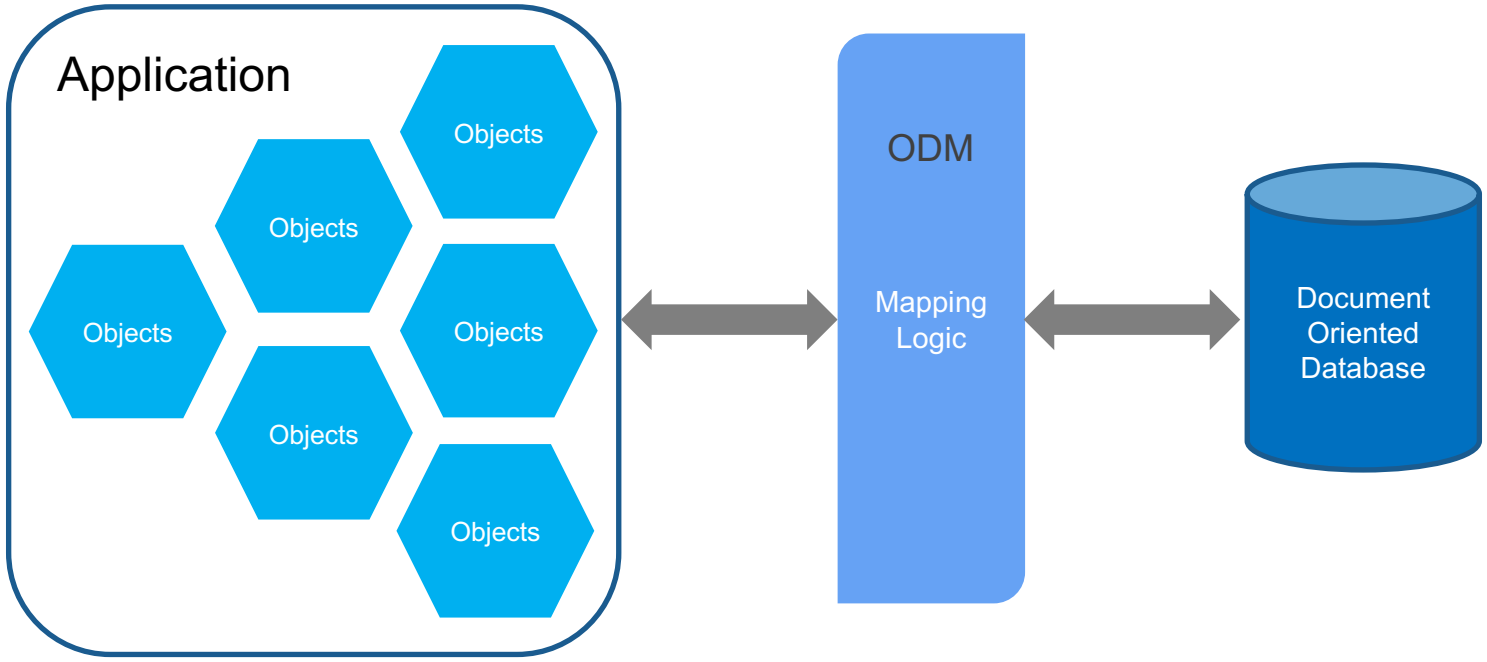
Introducción

- En el desarrollo de una aplicación suelen estar involucradas dos entidades diferentes, por una parte el código que mueve la aplicación y por otra los datos que se manejan

ODM

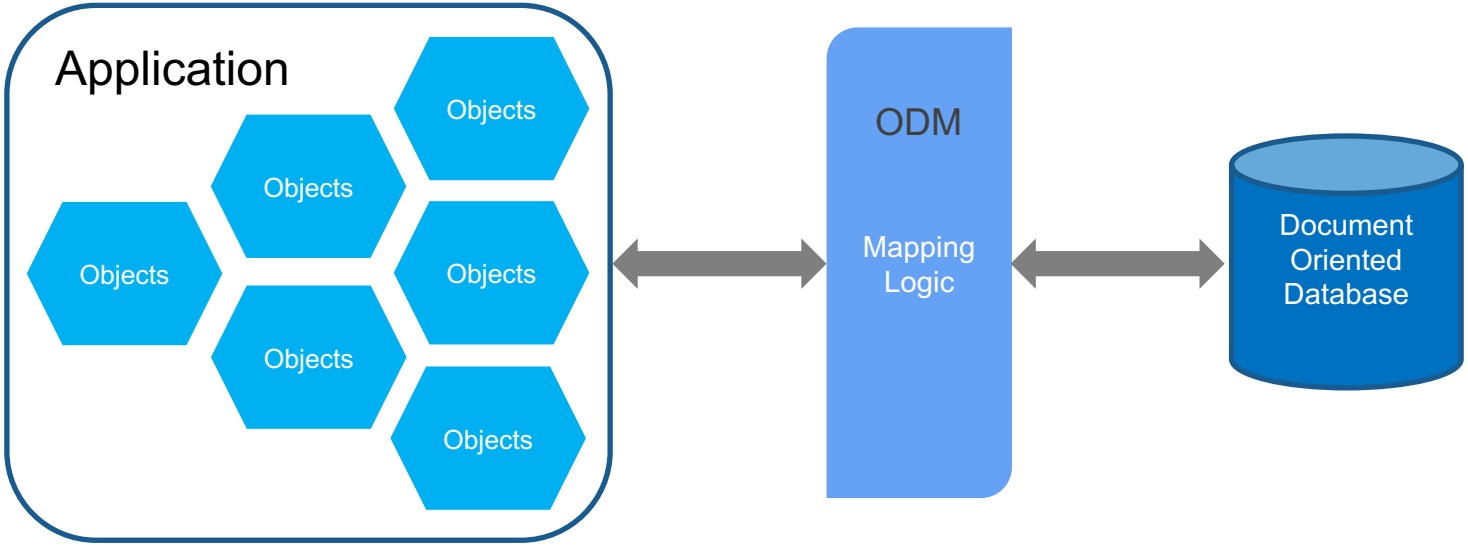
- Un ODM (Object Document Mapper o Mapeo Objeto-Documento) permite convertir los objetos de una aplicación a un formato adecuado para ser almacenados en cualquier base de datos.

- Los ODMs permiten realizar las acciones CRUD (Create, Read, Update, Delete) sin necesidad de incluir queries en el código.



ODM

- Esta técnica de programación se suele materializar en forma de librería o módulo escrita en el lenguaje de programación que estemos usando
- El modelo se encarga de asociar nuestro objeto con la colección en MongoDB
- De modo que en lugar de ejecutar queries a nuestras colecciones de documentos (en una BBDD NoSQL) ejecutaremos llamadas a objetos
- Existen diversos ODMs para diferentes lenguajes de programación:
  - NodeJS -> Mongoose, HumbleJS..
  - Java -> OdmManager, Morphia,..
  - Python -> MongoEngine, Mongothon..



Ejemplo NodeJS Mongo driver

- Código Driver Nodejs Mongo

```
const { MongoClient } = require("mongodb");

const uri = "mongodb://user:pass@localhost:27017";

const client = new MongoClient(uri);

async function run() {

      try {
            await client.connect();

            const database = client.db('sample_mflix’);

            const collection = database.collection('movies’);

            const query = { title: 'Back to the Future’ };

            const movie = await collection.findOne(query);

            console.log(movie);

      } finally {

            await client.close();
      }
}
run().catch(console.dir);
```

Importar librería

Crear el objeto conexión

Estableciendo conexión con la base de datos

Query a ejecutar

Ejecución de consulta en la base de datos

Ejemplo NodeJS Mongoonse (ODM)

- Código Mongoonse Nodejs MongoDB

```
const mongoose = require('mongoose');
```

Importar librería

```
mongoose..connect('mongodb://localhost/test',
                  { useNewUrlParser: true});
```

Crear objeto con parámetros de conexión

```
const Schema = mongoose.Schema
const BookSchema = new Schema({name : String}) ;
```

Crear el Schema

```
const Model = mongoose.model ();
const Book = Model('Books',BookSchema)
```

Definición de Modelo Book

```
const NodeJsGuide = new Book({name : 'NodeJS : A Guide' })
```

Crear el documento a insertar

```
async function insert(newBook) {
   let result= await newBook.save();
   return result;
}
```

Función para insertar Nuevo Book usando el modelo y la conexión

```
insert(NodeJsGuide);
```

Insertar NodeJsGuide usando la función insert

ODM-Ventajas

- Muchas cosas se hacen automáticamente. Por ej: conectar a la BBDD, convertir tipos (sobretodo problemáticos DATE y TIME), …
- Fomenta/fuerza el patrón MVC (Modelo Vista Controlador) y la app queda más limpia. Separación por capas.
- DRY (Don’t Repeat Yourself): el código del modelo queda en un sitio, fácil de actualizar, mantener y reutilizar
- Abstrae de la BBDD, con lo que se podrá cambiar en el futuro de tecnología de BBDD.
- Pueden tener caches, que harán que las consultas más típicas seas más rápidas (mejoran la performance)
- Suele evitar inyecciones de código y “sanitiza” en general el código
- Generación automática de código (entities y schemas)

ODM-Desventajas

- Para empezar pueden llegar a ser muy complejos.
- Puede llevar una curva de aprendizaje que podría retrasar el tiempo de desarrollo del proyecto.
- Se necesita alta estandarización de código y una buena arquitectura de la aplicación.
- Hay ocasiones en las que interviene un gran número de registros por cada petición. Puede saturar la memoria de objetos (por ejemplo reportes que incluyan muchísimos datos)

Mapeo Objecto-Documento (ODM)

Alejandro Pozo

Andrés Muñoz

Enrique Barra