

UNIVERSIDAD PRIVADA DE TACNA
INGENIERIA DE SISTEMAS

TITULO:

TRABAJO ENCARGADO N° 02
COMPARATIVA DE TECNOLOGIAS OR-M

CURSO:

BASE DE DATOS II

DOCENTE:

Ing. Patrick Cuadros Quiroga

Integrantes:

Sivirichi Falcón, Ricardo Alonso	(2018060905)
Chambilla Maquera, Araceli Noemi	(2018060897)
Alfárez Ponce, Pedro Alberto	(2020066317)
Cotrado Marino, Ana Luz	(2018060907)

Tacna - Perú
2021

1 Introduccion

Este artículo es el primero de una serie de tres que tienen como objetivo analizar las ventajas y desventajas del uso de ORMs (Object Relational Mappings) o, Mapeadores de Objetos Relacionales, así como proporcionar argumentos para decidir su inclusión o no en nuestro desarrollo de aplicaciones con especial atención en cómo afecta su uso al rendimiento global de las mismas.

2 Desarrollo

¿Qué es un ORM? Un ORM es un modelo de programación que permite mapear las estructuras de una base de datos relacional (SQL Server, Oracle, MySQL, etc.), en adelante RDBMS (Relational Database Management System), sobre una estructura lógica de entidades con el objeto de simplificar y acelerar el desarrollo de nuestras aplicaciones. Las estructuras de la base de datos relacional quedan vinculadas con las entidades lógicas o base de datos virtual definida en el ORM, de tal modo que las acciones CRUD (Create, Read, Update, Delete) a ejecutar sobre la base de datos física se realizan de forma indirecta por medio del ORM. **¿Qué es Java Hibernate?** “Hibernate es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java (y disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones. Hibernate es software libre, distribuido bajo los términos de la licencia GNU LGPL”.

Historia Se empezó a desarrollar hace unos años por Gavin King siendo hoy Gavin y Christian Bauer los principales gestores de su desarrollo. Hibernate parte de una filosofía de mapear objetos Java “normales”, también conocidos en la comunidad como “POJOs” (Plain Old Java Objects), no contempla la posibilidad de automatizar directamente la persistencia de Entity Beans tipo BMP (es decir, generar automáticamente este tipo de objetos), aunque aún así es posible combinar Hibernate con este tipo de beans utilizando los conocidos patrones para la delegación de persistencia en POJOs. Una característica de la filosofía de diseño de Hibernate ha de ser destacada especialmente, dada su gran importancia: puede utilizar los objetos Java definidos por el usuario tal cual, es decir, no utiliza técnicas como generación de código a partir de descriptores del modelos de datos o manipulación de bytecodes en tiempo de compilación (técnica conocida por su amplio uso en JDO), ni obliga a implementar interfaces determinados, ni heredar de una superclase.

3 Conclusiones

Hibernate es una herramienta ORM completa que ha conseguido una excelente reputación en la comunidad de desarrollo posicionándose claramente como el producto Open Source líder en este campo gracias a sus prestaciones, buena

documentación y estabilidad. Es valorado por muchos incluso como solución superior a productos comerciales dentro de su enfoque, siendo una muestra clara de su reputación y soporte la reciente integración dentro del grupo JBoss que seguramente generará iniciativas muy interesantes para el uso de Hibernate dentro de este servidor de aplicaciones.

4 Recomendaciones

Hibernate no sólo se ocupa del mapeo desde las clases Java a las tablas de las bases de datos (y desde los tipos de datos de Java a los tipos de datos de SQL), sino que también facilita la consulta y recuperación de datos. Esto puede reducir de manera importante el tiempo de desarrollo que se tomaría con el manejo de datos de forma manual en SQL y JDBC.

5 Ibatis

Resumen

iBatis es un framework ligero de persistencia que facilita el diseño de la capa de persistencia utilizada en las aplicaciones Java para acceder a nuestro repositorio de datos. Permite que el desarrollador se olvide de la implementación del acceso a datos, únicamente se debe preocupar por realizar una correcta configuración.

Abstract

iBatis is a lightweight persistence framework that facilitates the design of the persistence layer used in Java applications to access our data repository. It allows the developer to forget about the implementation of data access, he only has to worry about making a correct configuration.

5.1 Introducción

iBatis es un framework de código abierto basado en capas desarrollado por Apache, que se ocupa de la capa de persistencia (se sitúa entre la capa de negocio y la capa de la base de datos). Simplifica la implementación del patrón de diseño Direct Access Objects (DAO) y la persistencia de objetos en bases de datos relacionales. iBatis no es un ORM (Object Relational Mapper), es decir, no es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en base de datos; por lo que se pueden utilizar modelos de datos existentes o poco normalizados y, finalmente, no es completamente transparente (el programador programa el SQL). iBatis asocia objetos de modelo (JavaBeans) con sentencias SQL o procedimientos almacenados mediante ficheros descriptores XML, simplificando la utilización de bases de datos.

5.2 Concepto

La API iBATIS Data Mapper permite a los programadores mapear fácilmente objetos JavaBeans a PreparedStatement parámetros y Resultsets. La filosofía detrás de Data Mapper es simple: proporcionar un marco simple para proporcionar el 80

¿Cómo funciona? Data Mapper proporciona un marco muy simple para usar descriptores XML para mapear JavaBeans, Map implementaciones, tipos de envoltorios primitivos e incluso documentos XML a un SQL declaración.

5.3 Historia

En el año 2001 Clinton Begin comenzó el proyecto iBATIS. Originalmente iba a tratarse de un proyecto criptográfico. El primer producto de iBATIS fue Secrets, un sistema de criptografía. En ese mismo año Microsoft publicó un controvertido artículo para demostrar que su novedoso framework, .NET 1.0 era 10 veces más rápido y 4 veces más productivo que Java. para ello tomó la aplicación Web "Pet Store" como demostración de uso de diversos patrones de diseño en Java y construyó su propia versión en .NET.

En 2002 Clinton desarrolló una versión de la aplicación denominada JetStore que demostraba que Java podía ser más productivo que .NET, el trabajo tuvo gran repercusión. en foros informáticos y dio un gran interés por la capa de persistencia de Clinton.

Pronto se crearía el proyecto iBATIS Database Layer 1.0 que contaba de dos componentes: iBATIS DAO e iBATIS SQL Maps, la segunda versión de iBATIS se publicó en junio de 2004. Se desarrolló dentro de la comunidad de la ASF (Apache Software Foundation). Era funcionalmente similar a la primera versión pero se hizo un importante rediseño interno. Clinton cedió el código y el nombre de iBATIS a la ASF y el proyecto estuvo alojado 6 años en ASF aunque el equipo que mantenía el código original apenas cambió.

iBATIS dejó fuera de soporte el componente de iBATIS DAO por considerar que había factorías de terceros mucho más potentes como Spring Framework, el 19 de mayo de 2010 se publica la versión 3.0 de iBATIS y simultáneamente mueven el desarrollo de ASF a Google Code bajo el nuevo nombre MyBatis.

5.4 Configuración

5.5 Instanciar

Lo primero que hacemos en el código es instanciar una clase de Ibatis que lea los ficheros de configuración y que es a la que pediremos que nos realice las consultas, inserciones, modificaciones y borrados. El código para instanciar esta clase de Ibatis puede ser como este:

```
String resource = "com/chuidiang/xml/ConfiguracionIBatis.xml";  
Reader reader = Resources.getResourceAsReader(resource);  
SqlMapClient sqlMap = SqlMapClientBuilder.buildSqlMapClient(reader);
```

Este código simplemente obtiene un Reader de nuestro fichero de configuración de Ibatis y le pide a SqlMapClientBuilder un SqlMapClient. El código puede lanzar excepciones, así que deberíamos meterlo en un try-catch o bien relanzar la excepción en nuestro método que tenga este código.

5.6 Consultas

Vamos ahora a usar los distintos SELECT que hemos puesto en el fichero de configuración. El primero es un SELECT por id, al que hemos puesto el nombre de "getCoche". Para la cláusula WHERE hemos usado value y no hemos indicado el parameterClass, así que aquí deberemos pasar directamente un entero. Si pasamos una clase que no sea un tipo primitivo, Ibatis tratará de llamar al método getId() o get("id"). El código para esta consulta puede ser así `Integer claveCoche = new Integer(1); Coche coche = (Coche) sqlMap.queryForObject("getCoche", claveCoche);`

En `queryForObject()` estamos indicando que nos devuelva un solo objeto Coche. Como parámetro pasamos el nombre que le dimos a esta consulta en el fichero COCHE.xml al SELECT y el entero que hace de clave. Ibatis se encarga de poner el entero en value, hacer la consulta, hacer un new de la clase Coche y de rellenar todos sus parámetros llamando a los distintos métodos set() de la clase Coche. El nombre de los métodos set() será el de los AS que hemos puesto en la consulta. Otro SELECT que tenemos es al que hemos llamado "getCoches", que pretendemos que nos devuelva una lista de coches. Este SELECT no tiene ningún parámetro de entrada, ya que consulta todos los coches y no hay cláusula WHERE. El resultClass indica con qué clase se hará cada uno de los registros leídos de base de datos. El código para usar este SELECT puede ser como el siguiente `List<Coche> coches = sqlMap.queryForList("getCoches", null);`

Hemos llamado a `queryForList()` para indicar que queremos una lista. Como parámetro hemos pasado el nombre del SELECT que es "getCoches" y un null, puesto que no hemos puesto cláusula WHERE. Ibatis se encarga de hacer la consulta, construir tantas clases Coche como registros obtenga y devolvernos una List de Coche. Finalmente, otro SELECT que hemos puesto es para obtener un Hashtable en vez de una clase Coche. Al SELECT lo hemos llamado "getHashCoche". No hemos indicado un parameterClass valor, tendremos que pasar un entero como parámetro de entrada. Como resultClass hemos puesto un `java.util.Hashtable`, que será donde Ibatis nos devuelva el resultado. El código para usar esto puede ser

```
Map hashCoche = (Map) sqlMap.queryForObject("getHashCoche", 3);
```

Ibatis se encarga de hacer la consulta, reemplazando el valor del WHERE por un 3, luego instancia un Hashtable, lo rellena con el método `put("id",...)`, `put("marca", ...)` y lo devuelve. Los valores que usa como claves para el Hashtable son los AS que hemos puesto en el SELECT.

5.7 Insertar Registros

Para insertar usaremos la SQL que hemos llamado "insertCoche". No lleva resultClass puesto que esta SQL no devuelve ningún registro. No le hemos puesto parameterClass, así que en el método java correspondiente podremos usar indistintamente una clase Coche o un Hashtable. Este podría ser el código de ejemplo:

```
// Insertar un coche nuevo. No ponemos id porque se genera solo en la
inserción.
Coche coche = new Coche();
coche.setMarca("una marca");
coche.setMatricula("una matricula"); coche.setFechaMatricula(new Date());
sqlMap.insert("insertCoche", coche);
// Ahora insertamos con un Hashtable.
Hashtable hashCoche = new Hashtable();
hashCoche.put("marca", "la marca");
hashCoche.put("matricula", "x-2222-z");
hashCoche.put("fechaMatricula", new Date());
sqlMap.insert("insertCoche", hashCoche);
```

En el método insert() pasamos como parámetro el nombre de la SQL "insertCoche" y la clase que contiene los datos. En el caso del bean, Ibatis llamará a los métodos get() para obtener los valores, en el caso del Hashtable llamará al método get(clave) para obtener los valores.

5.8 Borrar Registros

Para el borrado de registros, usaremos la SQL de nombre "removeCoche" a la que se le debe pasar un entero con el valor del id del coche a borrar. El código de ejemplo puede ser sqlMap.delete("removeCoche", 11);

que borrará de BD el coche cuya clave es 11.

6 Ebean

Resumen En este trabajo se presenta una forma de implementar las tecnologías orm ebean y jpa en el desarrollo de una aplicación web empleando playframework, la integración de ambas tecnologías permitirá aprovechar sus mejores características realizando operaciones con la base de datos, por lo que ebean se utilizará para operaciones create, update y delete con registros de una sola tabla, mientras que JPA se utilizará para operaciones con varias tablas mediante vistas o procedimientos almacenados que retornan registros.

Abstract This work presents a way of implementing ORM ebean and jpa technologies in the development of a web application using playframework, the integration of both technologies will let improve their best characteristics making operations over database, therefore ebean will be used to create, update and delete operations with records of one unique table, whereas that JPA will be

used for operations with several tables through views or store procedures that return records.

6.1 Introducción

El mapeo objeto-relacional es una técnica de programación para convertir datos del sistema de tipos utilizado en un lenguaje de programación orientado a objetos al utilizado en una base de datos relacional. En la práctica esto crea una base de datos virtual orientada a objetos sobre la base de datos relacional. Playframework es un framework de aplicaciones web de java y scala de alta productividad que integra los componentes y APIs que necesita para el desarrollo de aplicaciones web.

6.2 Desarrollo

La tecnología Ebean permite que los objetos, de las clases modelos (de modelo-vista-controlador) que extienden a la clase ebean. Model pueden adoptar el estado y comportamiento soportados por Ebean, el comportamiento traducido a funcionalidades consisten en la realización de consultas específicas sobre los mismos modelos, por ejemplo al realizar una consulta a través de un campo id, o bien, para que se obtengan todos los registros encontrados en el propio modelo.

Ebean es un producto de mapeo relacional de objetos escrito en Java . Está diseñado para ser más simple de usar y comprender que los productos JPA (Java Persistence API) o JDO (Java Data Objects).

¿Qué es JDBC en Ebeans ? JDBC es la API de bajo nivel que todos usan al final para hablar con una base de datos. Pero sin utilizar una API de nivel superior, tiene que hacer todo el trabajo por su cuenta (escribiendo consultas SQL, asignando resultados a objetos, etc.).

6.3 Conclusión:

El desarrollo de este trabajo aprovecha las ventajas de ambas tecnologías ebean y jpa. Ya que con ebean se evitan las consultas JDBC puras y que los registros obtenidos tengan que ser mapeados a objetos de forma explícita por el desarrollador, por lo cual permite ejecutar operaciones create, update y delete en registros unitarios mediante objetos. Mientras que al utilizar jpa es posible ejecutar operaciones de tipo sql haciendo la recomendación que se realicen llamadas a procedimientos almacenados, y que sean estos lo que contengan las consultas y operaciones complejas con la base de datos que puedan incluir múltiples transacciones y operaciones sql.

7 Torque

7.1 Introducción

El mapeo objeto-relacional es una técnica de programación para convertir datos del sistema de tipos utilizado en un lenguaje de programación orientado a objetos al utilizado en una base de datos relacional. En la práctica esto crea una base de datos virtual orientada a objetos sobre la base de datos relacional. Playframework es un framework de aplicaciones web de java y escala de alta productividad que integra los componentes y APIs que necesita para el desarrollo de aplicaciones web.

7.2 Desarrollo

¿Qué es Torque? Apache Torque es un mapeador relacional de objetos para java. En otras palabras, Torque le permite acceder y manipular datos en una base de datos relacional usando objetos java. A diferencia de la mayoría de los otros mapeadores relacionales de objetos, Torque no utiliza la reflexión para acceder a las clases proporcionadas por el usuario, pero genera las clases necesarias (incluidos los objetos de datos) a partir de un esquema XML que describe el diseño de la base de datos. El archivo XML se puede escribir a mano o se puede generar un punto de partida a partir de una base de datos existente. El esquema XML también se puede utilizar para generar y ejecutar un script SQL que crea todas las tablas en la base de datos. Dado que Torque oculta los detalles de implementación específicos de la base de datos, Torque hace que una aplicación sea independiente de una base de datos específica si no se utilizan características exóticas de la base de datos. El uso de la generación de código facilita la personalización de la capa de la base de datos, ya que puede anular los métodos generados y así cambiar fácilmente su comportamiento. Una estructura de plantilla modular permite la inclusión de sus propias plantillas de generación de código durante el proceso de generación de código Originalmente Torque se desarrolló como parte del framework Turbine en la versión 2.1, pero después pasó a ser una aplicación independiente.

7.3 Conclusión:

Las herramientas de ORM funcionan como traductores entre dos modelos, por lo cual los programadores centran la atención en desarrollar las funcionalidades y delegan la tarea de comunicación con la base de datos a la herramienta. A pesar de que esta herramienta está volviéndose muy popular para los desarrolladores de software, hay una escasa documentación sobre algunos elementos para el mapping entre la base de datos y la aplicación, por lo cual la gran cantidad de foros que se encuentran en la red constituyen un soporte valioso cuando se está desarrollando una aplicación utilizando esta framework.

8 Bibliografía

Aguilar, J. (s.f.). Estructura de Datos en C. Madrid: McGrawHill. Obtenido de <https://elibro.net/es/lc/bibliotecaupt/titulos/50068>

MyBatis.org. (2010-2020). MyBatis. Obtenido de <http://www.mybatis.org/spring/es/index.html>

Mapper, D. (2006). Ibatis. a.k.a. Obtenido de <https://ibatis.apache.org/docs/java/pdf/iBATIS-SqlMaps-2en.pdf>

Weiss, M. A. (2013). Estructura de Datos en Java. Madrid: Pearson S.A. Obtenido de <https://elibro.net/es/ereader/bibliotecaupt/108459>

Yañez, C. (06 de Mayo de 2018). CEAC. Obtenido de <https://www.ceac.es/blog/herramientas-de-mapeo-objeto-relacional-orm>

Artalejo, M. R. (2011). Estructuras de datos con un enfoque moderno. Madrid: Complutense.