Facultad de Ingeniería

Pasantía

Evaluación de Arquitecturas de Software

Responsable	
Andrea Delgado	
Integrantes	
Alberto Castro	3.239.979-8
Martín Germán	3.217.050-4

Versión 1.2
Documento 3/3

ÍNDICE

1 INTRODUCCIÓN

Para la mayoría de las aplicaciones, almacenar y recuperar información implica alguna forma de interacción con una base de datos relacional. Esto ha representado un problema fundamental para los desarrolladores porque algunas veces el diseño de datos relacionales y los ejemplares orientados a objetos comparten estructuras de relaciones muy diferentes dentro de sus respectivos entornos.

Las bases de datos relacionales están estructuradas en una configuración tabular y los ejemplares orientados a objetos normalmente están relacionados en forma de árbol. Esta "diferencia de impedancia" ha llevado a los desarrolladores de varias tecnologías de persistencia de objetos a intentar construir un puente entre el mundo relacional y el mundo orientado a objetos.

Hibernate pretende ser este puente. Este framework es un mapeador objeto relacional (ORM, por sus siglas en inglés) para aplicaciones Java que alcanza este desafío proveyendo la habilidad de mapear un modelo de objetos en un modelo relacional y su correspondiente esquema de la base de datos.

De los distintos ORM en el mercado, Hibernate esta considerado como uno de los más flexibles y poderosos. Este, mapea clases Java en tablas de la base de datos y los tipo de datos Java en tipo de datos SQL. Hibernate provee soporte para relaciones de colecciones y de objetos, así como tipos compuestos. Además de persistir objetos, Hibernate provee un rico lenguaje de consulta propio, Hibernate Query Language (HQL, el cual luego es transformado a SQL), para recuperar objetos de la base de datos, así como un eficiente cache y soporte para Java Management Extensions (JMX). Más aun, la posibilidad de utilizar tipos de datos definidos por el usuario y claves primarias compuestas le dan una significativa flexibilidad para soportar aplicaciones legadas. El objetivo de Hibernate es alivianar al programador de las tareas de procesar los datos utilizando SQL y JDBC.

Hibernate esta licenciado como Lesser GNU Public License, lo cual es suficiente para utilizarlo en aplicaciones comerciales como open source. Este soporta numerosos motores de base de datos, incluyendo Oracle y DB2, así como también los motores de base de datos open source más populares como PostgreSQL y MySQL.

1.1 Como trabaja Hibernate

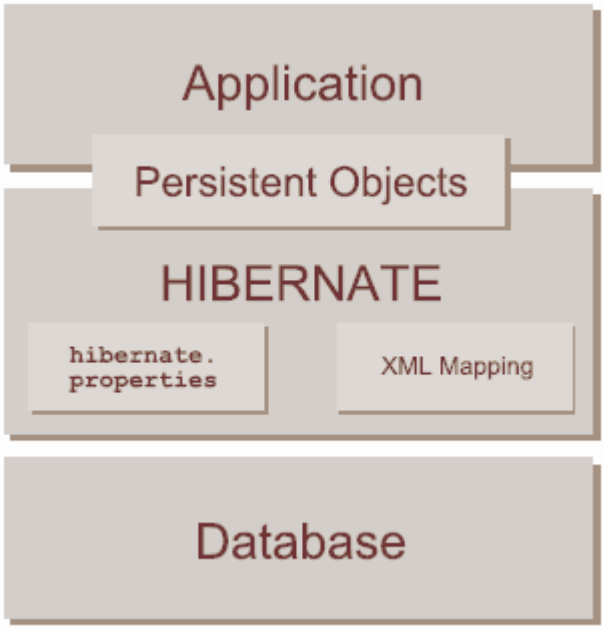
En vez de procesar bytecode o código generado, Hibernate utiliza reflection para determinar las propiedades persistentes de una clase. Los objetos a persistirse son definidos en un archivo de mapeo, el cual sirve para describir los campos persistentes y las asociaciones, así como cualquier subclase o proxy del objeto persistente. Los archivos de mapeo son compilados en el arranque de la aplicación y proveen al framework con la información necesaria de la clase. Adicionalmente, son utilizados para dar soporte a las operaciones como generación del esquema de una base de datos o creación de stubs de archivos fuente Java.

Una SessionFactory es creada a partir de la colección compilada de los archivos de mapeo. La SessionFactory provee la interfaz Session, la cual es el mecanismo para administrar las clases persistentes. La clase Session provee la interfaz entre el data store y la aplicación. La interfaz Session es un wrap de conexiones JDBC, las cuales pueden ser administradas por el usuario o controladas por Hibernate, y su propósito es ser utilizadas por un único hilo de la aplicación, y luego ser cerradas y descartadas.

2 ARQUITECTURA

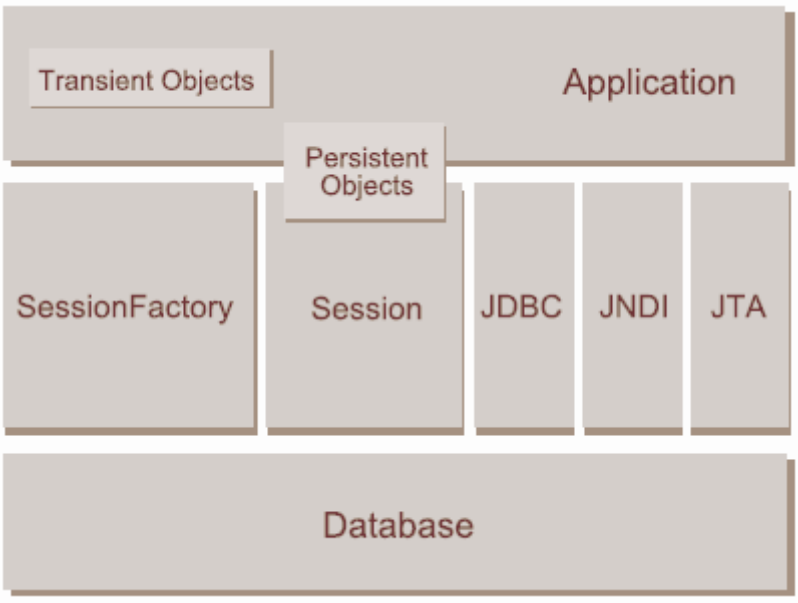
2.1 Visión general

Una vista de alto nivel de la arquitectura de Hibernate:

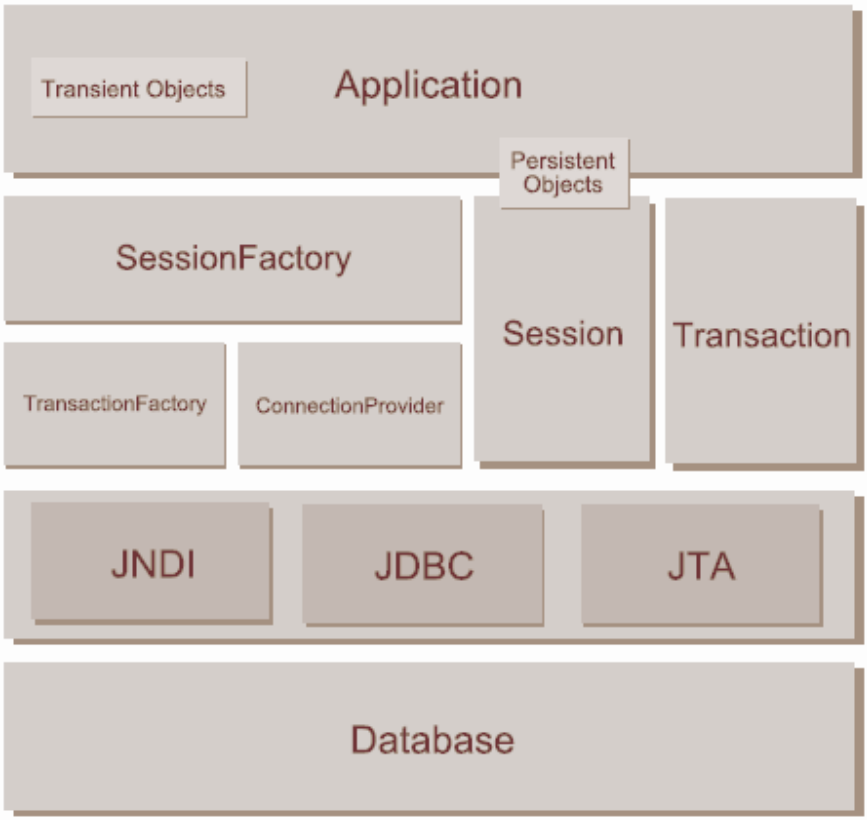


Este diagrama muestra a Hibernate utilizando la base de datos e información de configuración para proveer servicios de persistencia (y objetos persistentes) a la aplicación.

A continuación mostraremos dos vistas más detalladas de la arquitectura del runtime. En la "pequeña" arquitectura la aplicación provee su propia conexión JDBC y administra sus propias transacciones. Esta propuesta utiliza un subconjunto mínimo de la API de Hibernate.



La arquitectura "completa" abstrae a la aplicación de las APIs subyacentes JDBC – JTA y le permita a Hibernate cuidar los detalles.



Definimos algunos objetos mencionados en los diagramas:

- *SessionFactory*: Un cache (inmutable) de los archivos de mapeo compilados de una única base de datos. Una fábrica de Session y un cliente de ConnectionProvider. Puede contener un cache (de segundo nivel) opcional de datos que es reusable entre transacciones.
- *Session*: Un objeto que corre en un hilo simple de corta vida que representa una conversación entre la aplicación y los datos. Es un wrap de una conexión JDBC. Fábrica de Transaction. Mantiene un cache (de primer nivel) de objetos persistentes.
- *Persistent objects y collections*: Objetos que corren en un hilo simple y de corta vida que contienen el estado de persistencia y funciones de negocio. Estos pueden ser JavaBeans – POJO ordinarios, la única particularidad que estos tienen es que están asociados a una única Session.
- *Transient y detached objects y collections*: Instancias de clases persistentes que no están actualmente asociadas a una Session. Pueden haber sido instanciadas por la aplicación y todavía no han sido persistidas o haber sido instanciadas por una Session cerrada.
- *Transaction*: (Opcional) Un objeto que corre en un hilo simple de corta vida utilizado por la aplicación para especificar las unidades de trabajo atómicas.
- *ConnectionProvider*: (Opcional) Una fábrica (y pool) de conexiones JDBC. Aplicación abstracta para los Datasource o DriverManager subyacentes.
- *TransactionFactory*: (Opcional) Una fábrica de Transaction.
- *Extension Interfaces*: Hibernate ofrece muchas interfaces opcionales para extender que se pueden implementar para personalizar el comportamiento de la capa de persistencia.

En una arquitectura "pequeña" la aplicación omite un *Transaction/TransactionFactory* y/o *ConnectionProvider* para comunicarse con JTA o JDBC directamente.

2.2 Estados de las Instancias

Una instancia de una clase persistente puede estar en tres posibles estados, los cuales se definen de acuerdo al *contexto de persistencia*. El objeto Session de Hibernate es el contexto de persistencia:

- *Transient*: La instancia no esta y nunca ha sido asociada con algún contexto de persistencia. El objeto no tiene clave primaria.
- *Persistent*: El objeto esta actualmente asociado a un contexto de persistencia. Tiene calve primaria.
- *Detached*: La instancia estuvo una vez asociada con un contexto de persistencia, pero ese contexto fue cerrado, o la instancia fue serializada por otro proceso. Tiene clave primaria. Para las instancias en este estado Hibernate no asegura la relación entre lo persistido y lo que se encuentra en memoria.

2.3 Integración JMX

JMX es el estándar J2EE para la gestión de componentes Java. Hibernate puede ser gestionado vía los servicios del estándar JMX.

3 CARACTERÍSTICAS PRINCIPALES

3.1 Open Source

Sin costo de licencia para producción. Hibernate esta licenciado bajo GNU Lesser General Public License (LGPL) y es libre de bajar y utilizar tanto para de desarrollo como para producción.

3.2 Aumenta la productividad del desarrollo

Hibernate elimina el repetitivo y tedioso trabajo de codificar y permite centrarse más en los problemas del negocio. Sin importar la estrategia de desarrollo elegida, Hibernate reduce el tiempo de implementación.

3.3 Reduce el costo de mantenimiento

Hibernate reduce las líneas de código automatizando el ORM (Object Relational Mapping). Permite utilizar orientación a objetos del lado Java manteniendo el esquema relacional normalizado y garantizando la integridad de los datos.

3.4 Persistencia transparente

Hibernate no requiere interfaces o clases base para persistir clases y permite persistir cualquier clase o tipo de dato estructurado. Además no utiliza bytecode, ya que obtiene las propiedades a persistir mediante reflection.

3.5 Mapeo Objeto/Relacional flexible

Hibernate esta orientado a archivos XML de mapeo que definen el ORM. Todas las propiedades de Hibernate son configurables desde estos archivos XML, no es necesario recodificar.

3.6 APIs simples

Hibernate incluye una API núcleo para código de aplicación, una API extensible para las personalizaciones, y una API de metadata para las aplicaciones que requieren acceso a la metadata de persistencia de Hibernate.

3.7 Leguaje de consulta orientado a objetos

Hibernate provee un poderoso leguaje de consulta llamado Hibernate Query Language (HQL) cuya sintaxis es familiar a SQL e incluye soporte para consultas orientadas a objetos y polimórficas. Las consultas también pueden ser expresadas en SQL nativo.

3.8 Opera en ambientes gestionados y no gestionados

Hibernate es utilizado comúnmente en aplicaciones Java Swing, Java Servlet o J2EE.
Hibernate es capaz de operar con cualquier servidor de aplicaciones J2EE, pudiendo ser configurado y gestionado vía JMX. El sistema de transacciones de Hibernate pude ser integrado a un servidor de aplicaciones transaccional J2EE vía JTA, o también puede correr por fuera del contenedor del servidor de aplicaciones.

3.9 Alta performance

Hibernate incluye inicialización perezosa, outer join, union, así como soporte para bloqueo optimista con versionado automático. Hibernate no requiere tabla o campo especial alguno. La mayoría de la generación del SQL se realiza en la inicialización del sistema en vez de en tiempo de ejecución.

3.10 Cache de doble capa

La arquitectura de cache de doble capa de Hibernate permite seguridad a los hilos, acceso no bloqueante a los datos, sesión a nivel de cache, y un cache opcional de consultas. Hibernate asimismo trabaja bien en ambientes distribuidos donde otras aplicaciones están accediendo a los datos de la base en forma simultánea.

4 REFERENCIAS

- Hibernate Reference Documentation v3.0.5.
- Manual de Hibernate v2.0; Héctor Suárez González.
- Object Relational Tool Comparison; http://c2.com/cgi-bin/wiki?ObjectRelationalToolComparison - 28/12/2005
- Introduction to Hibernate; Nick Heudecker;
- Hibernate Brochure - 02/2005
- www.hibernate.org - 28/12/2005
- www.answers.com - 28/12/2005