ACCESO-A-DATOS-TEMA-5.pdf

user_4383848

Acceso a datos

1º Desarrollo de Aplicaciones Multiplataforma

Estudios España

ACCESO A DATOS - TEMA 5
MANEJO DE CONECTORES II

1. GESTORES DE BD EMBEBIDOS E INDEPENDIENTES:

Bases de datos en memoria: Almacena toda la información de la misma en memoria principal del sistema (RAM). La información sólo permanece mientras la aplicación esté ejecutándose.

Bases de datos embebidas: Aquella que es parte de la aplicación que se ha desarrollado. Accederemos a este tipo de bases de datos por medio de los ya conocidos JDBC driver. El motor de la base de datos corre con el mismo motor de la aplicación Java (JVM-Java Virtual Machine) mientras la aplicación está en ejecución. Una posible desventaja de usar este tipo de base de datos puede ser la cohesión y la **dificultad de mantenimiento** que podemos encontrar en estos casos.

Bases de datos independientes: Tenemos gestor o administrador dedicado a resolver ciertos problemas de mantenimiento(SGBD). **Las bases de datos cliente/servidor son independientes y las más pesadas y potentes**. Nos referimos a pesadas no por lentitud, sino por la cantidad de procesos adicionales que engloba una base de datos de este tipo.

2. GESTORES DE BD EMBEBIDOS I: HYPERSQL Y OBJECTDB:

HyperSQL: Se ajusta a la versión estándar SQL 2011 y a la especificación JDBC 4. Además, soporta cada característica clásica de las bases de datos modernas de tipo relacionales. **Se puede ejecutar tanto en modo embebido como en modo servidor.** La base de datos es bastante estable, es de una confianza considerable. A partir de la versión 2.3.X en adelante soporta el mecanismo MVCC (Multi version concurrency control). Corre sin problema en nuestra JVM, facilitándonos una interfaz JDBC para el acceso a datos.

ObjectDB: también **contiene los dos modos, embebido y modo cliente/servidor.** Es una base de datos orientada a objetos con soporte para la especificación JPA2. Como resultado, el uso de una capa de abstracción, como la de Hibernate, tiene un mejor rendimiento que cualquier otra base de datos. Debido a su compatibilidad por defecto con JPA, se podría eliminar la capa ORM directamente, si es que nuestro aplicativo tuviera una para aprovechar el rendimiento.

3. GESTORES DE BD EMBEBIDOS II: JAVA DB, APACHE DERBY Y H2:

Java DB y Apache Derby: Son muy similares, Java DB está construida sobre el motor de la base de datos Derby. Está escrita en lenguaje Java y puede ser incluida, fácilmente, en cualquier aplicación Java.
Derby soporta todas las funcionalidades estándar de una base de datos relacional. Puede ser **desplegada en el modo simple embebido o también en el modo cliente/servidor** como sus competidores.

H2 Database: las principales características de la base de datos H2 son:
• Es muy rápida, de código abierto, JDBC API.
• Contiene modo cliente/servidor, modo embebido y modo desplegable en memoria.
• Se puede usar perfectamente para aplicaciones web.
• Muy manejable y transportable, el fichero .jar ocupa 2MB de espacio total.
• Soporta MVCC (Multiversion concurrency control).
• Se puede usar Postgres ODBC driver.

4. GESTORES DE BD EMBEBIDOS III: COMPARATIVA:

EJEMPLO	H2	DERBY	HYPERSQL	MYSQL	POSTGRESS
Java Puro	SI	SI	SI	NO	NO
Modo memoria	SI	SI	SI	NO	NO
Base de datos encriptada	SI	SI	**SI**	NO	NO
Driver ODBC	SI	NO	NO	SI	SI
Búsqueda texto completo	SI	NO	NO	SI	SI
MVCC	SI			SI	SI
Espacio (embebido)	~2 MB	~3 MB	~1.5 MB	-	-
Espacio (cliente)	~500 KB	~600 KB	~1.5 MB	~1 MB	~700 KB

Podemos observar que la base de datos H2 es una muy buena opción para agregar una base de datos embebida, en memoria, o de tipo cliente/servidor a nuestro aplicativo.

SPRING BOOT:

Spring es un framework. Spring Boot es un módulo dentro de Spring, y que está en auge por su usabilidad y el hacer fácil ciertas partes del proceso que por defecto son más complejas. Spring Boot nace con el objetivo de facilitar al máximo la parte de selección de dependencias y la del despliegue de la aplicación.

(Puntos 7,8 y 9 del PDF son instrucciones en Spring Boot, tanto web, como en IDE)

5. GESTORES DE BASE DE DATOS INDEPENDIENTES:

Quizás, una de las diferencias más importantes es que **una base de datos independiente no puede ejecutarse bajo la misma máquina virtual de Java que usa nuestra aplicación en ejecución.**
Una base de datos independiente puede ser instalada en local, pero con fines de prueba o aprendizaje en la mayoría de casos. Lo común es tener una base de datos independiente separada e instalada en otra máquina.
Una base de datos independiente consumirá siempre más recursos que una embebida, es por ello que al estar en una máquina aislada para dicha base de datos, aprovechará mejor los recursos que tiene disponibles. A nivel de funcionalidad y operativo, las bases de datos embebidas están restringidas, sin embargo, las bases de datos independientes se preparan en dicha máquina separada y aislada para volcar todo el potencial que ofrece con todos sus procedimientos y operativos.