Acceso a datos

ORM – Mapeo objeto-relacional

Bases de datos embebidas – Base de datos H2

IES Clara del Rey – Madrid

Bases de datos embebidas

Una base de datos embebida es un sistema de acceso a base de datos que se integra dentro de la aplicación, y que permite acceder a los datos como si se dispusiera de un servidor SQL.

El sistema de gestión de base de datos se ejecuta dentro del mismo programa que accede a los datos, lo que permite una integración total, sin dependencias adicionales.

No es necesario instalar de forma independiente un servidor SQL como MariaDB, MySQL u Oracle.

Hay multitud de bases de datos embebidas, una lista no exhaustiva puede consultarse en: https://en.wikipedia.org/wiki/Embedded_database

Uso de bases de datos embebidas

Las aplicaciones más habituales son:

- Aplicaciones que aprovechen los beneficios de una BD (relaciones, fk, triggers, etc.), aunque no se disponga de un servidor. Ejemplos:
  - Aplicaciones en dispositivos móviles y sistemas embebidos. Por ejemplo, aplicaciones IoT, que registren datos y los envíen periódicamente.
  - Aplicaciones de escritorio, que trabajen de forma aislada, sin red.
- Escenarios de pruebas (testing) para no modificar o contaminar una base de datos "real". Se puede usar una réplica para pruebas.
- Prototipado rápido de aplicaciones, sin necesidad de infraestructura de datos.

Base de datos embebida H2

Es posiblemente la base de datos embebida más usada en Java, junto a SQLite, que se usa mucho en aplicaciones móviles.

Programada en Java, sus características principales son:

- Open Source
- Cumple con el estándar JDBC API
- Puede ejecutarse embebida o como servidor independiente. Tiene un modo (mixed) en el que la BD embebida recibe conexiones exteriores.
- Puede trabajar en memoria (desaparece la información al finalizar la aplicación) o en ficheros (se conserva la información)
- Incorpora una consola web para facilitar su gestión

H2 – Dependencia – Cadena de conexión

La dependencia Maven para usar esta base de datos en Java es:

```
<dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <scope>runtime</scope>
</dependency>
```

La cadena de conexión determina cómo opera la BD, y cómo se realizan las conexiones. Dos opciones básicas:

- Embebida en memoria: jdbc:h2:mem:<nombreBaseDatos>
- Embebida con un fichero de datos: jdbc:h2:file:<ficheroDatos>

Más opciones – Ver: http://www.h2database.com/html/features.html#database_url

Info. sobre scopes Maven: https://www.baeldung.com/maven-dependency-scopes

H2 – En memoria

La base de datos se crea en memoria, sin un fichero asociado.

Cuando la aplicación se cierra, los datos almacenados en la BD se pierden.

Suele ser necesario inicializar la base de datos:

- Crear tablas, relaciones, índices, etc.
- Insertar datos iniciales en las tablas.

Aunque técnicamente es posible conectar a una base de datos en memoria desde otro proceso, la conexión no es trivial, así que no es fácil gestionar la BD desde la herramienta "Databases" de IntelliJ.

Cuando se usa H2 con Spring, se puede usar una herramienta web integrada, que permite gestionar la base de datos.

H2 – En memoria – Spring Boot

En Spring Boot, la BD se configura en el fichero "application.properties".

Para la cadena de conexión se usa la variable "spring.datasource.url".
Para una base de datos embebida en memoria sería:

```
spring.datasource.url=jdbc:h2:mem:base-datos
```

Para activar la consola (false por defecto):

```
spring.h2.console.enabled=true
```

Ruta para acceder a la consola ("h2-console" por defecto):

```
spring.h2.console.path=/h2
```

En este caso, suponiendo que el puerto (variable server.port) sea el 8080, se puede encontrar la consola en http://localhost:8080/h2

H2 – En fichero

Las bases de datos en fichero son persistentes. Se almacenan los datos, y se mantienen entre ejecuciones de la aplicación.

La inicialización de la base de datos debe comprobar si ya existen las tablas o datos antes de crear nuevos elementos, para evitar errores.

La cadena de conexión para una base de datos en fichero es:

```
spring.datasource.url=jdbc:h2:file:~/base-datos
```

El path del fichero tiene que ser un path absoluto, o comenzar con "~" (relativo al home del usuario), o con "." (relativo a la aplicación).

Si el fichero no existe, se crea al conectar por primera vez a la BD.

Se puede usar la consola web de H2 para gestionar la BD.

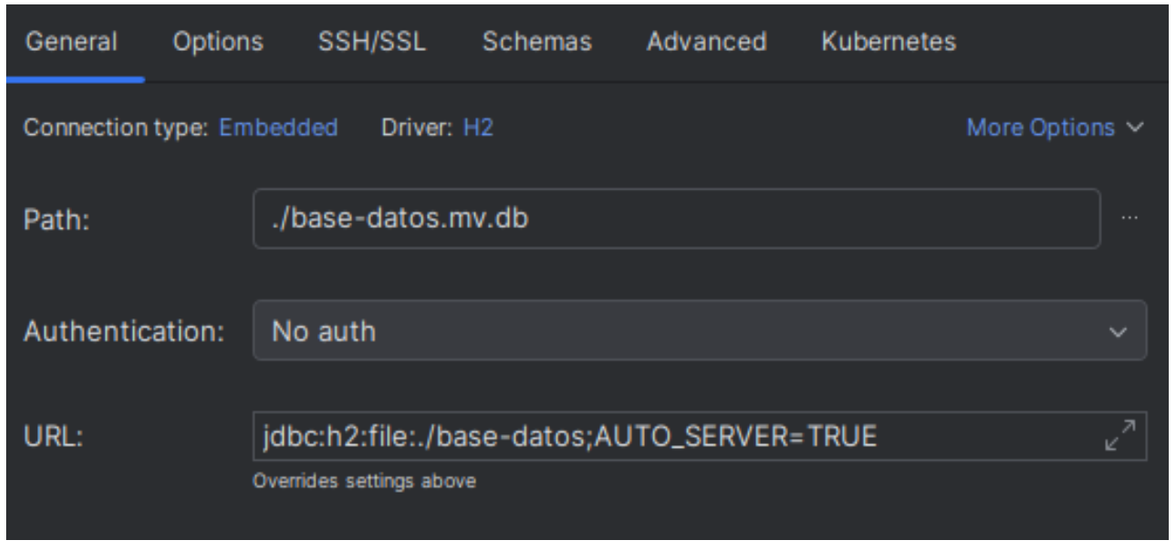
H2 – En fichero en modo "mixed"

H2 admite un modo "mixed" que funciona con acceso a fichero, y que admite conexiones externas, de otros procesos. Para habilitar este modo, hay que modificar la conexión:

jdbc:h2:file:./base-datos;AUTO_SERVER=TRUE

Esto permite, por ejemplo, gestionar la BD desde las herramientas de gestión de BD de IntelliJ.

Para configurar la fuente de datos en IntelliJ, añadir la fuente de datos en la ventana "Database", con la misma cadena de conexión.



H2 – Compatibilidad con otros SGBD

H2, al estar orientada en parte a pruebas, tiene una serie de opciones de compatibilidad con otros sistemas de gestión de bases de datos.

Las opciones se añaden a la cadena de conexión.

Por ejemplo, para fichero con compatibilidad con MariaDB, la cadena de conexión sería:

```
jdbc:h2:file:~/fichero;MODE=MariaDB;DATABASE_TO_LOWER=TRUE
```

Si, además se quiere que no distinga entre mayúsculas y minúsculas en lo que se refiere a nombres de tablas, columnas, etc., se añade:

```
CASE_INSENSITIVE_IDENTIFIERS=TRUE
```

Más compatibilidad: http://www.h2database.com/html/features.html#compatibility