Laboratorio de Bases de Datos

Aplicación JDBC

Departamento de Electricidad, Electrónica y Computación
Facultad de Ciencias Exactas y Tecnología
Universidad Nacional de Tucumán

Primer Semestre 2017

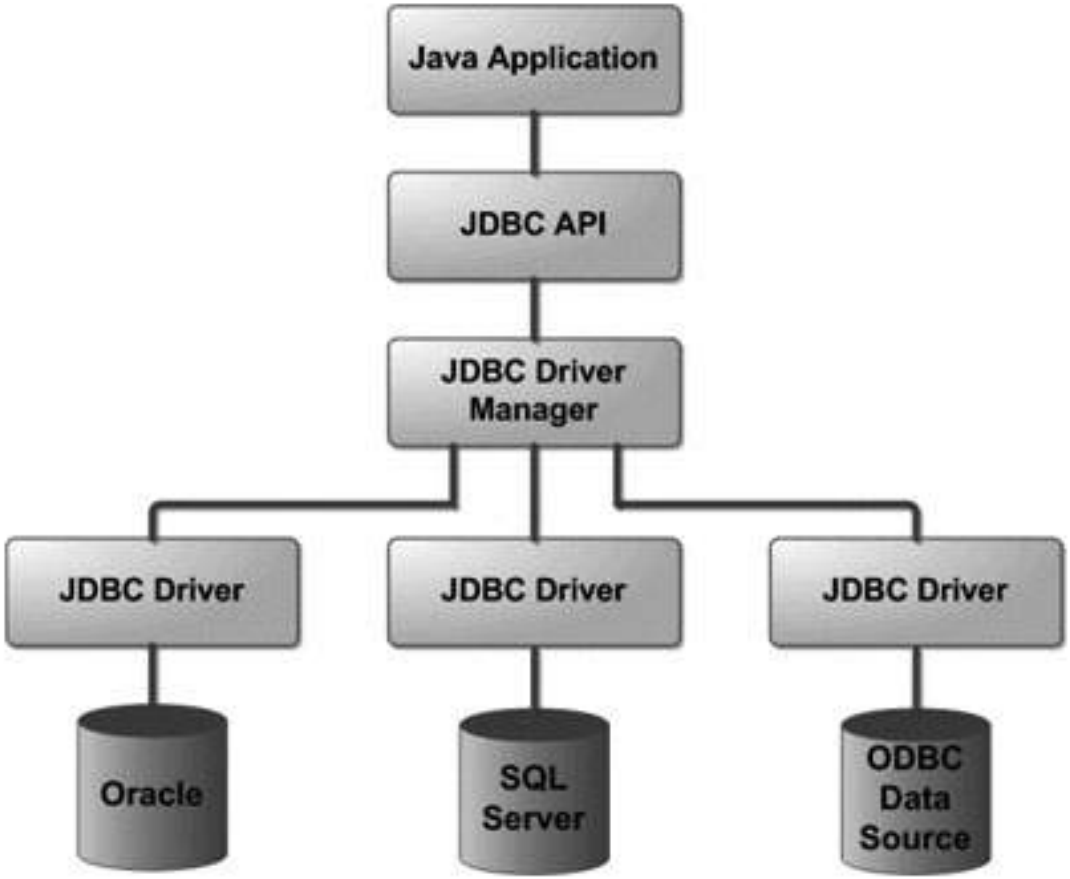
Introducción [1 | 1]

- JDBC
  - Introducción
  - Creación de una aplicación JDBC

Introducción [1 | 7]

- **JDBC:**
  - Java Database Connectivity
  - API de Java para conectividad entre Java y BD:
    - Conexión a una BD
    - Creación y ejecución de sentencias SQL

Introducción [2 | 7]



Introducción [3 | 7]

- **Administrador de drivers:**
  - JDBC usa un administrador de drivers (driver manager) y drivers específicos para una BD para brindar conectividad transparente a BD heterogéneas.
  - Asegura que se use el driver correcto para acceder a cada fuente de dato.
  - Soporta múltiples drivers conectados concurrentemente a BD heterogéneas.

Introducción [4 | 7]

- **Interfaces:**
  - **DriverManager**: maneja una lista de drivers. Según los pedidos de conexión de la aplicación Java, busca el driver adecuado.
  - **Driver**: maneja la comunicación con la BD. No se interactúa directamente con estos objetos, sino con objetos del tipo DriverManager, quien abstrae la complejidad de los primeros.

Introducción [5 | 7]

- **Interfaces:**
  - **Connection**: contiene métodos para contactarse con la BD. Toda comunicación con la BD se realiza a través de este objeto.
  - **Statement**: se utilizan objetos creados a partir de esta interfaz para ejecutar sentencias SQL en la BD.

Introducción [6 | 7]

- **Interfaces:**
  - **ResultSet**: estos objetos contienen los datos obtenidos de la BD. Actúan como iterador, permitiendo moverse dentro de los mismos.
  - **SQLException**: maneja todos los errores que puedan ocurrir en la aplicación que interactúa con la BD.

Introducción [7 | 7]

- **Pasos:**
  - 1. Instalar Java
  - 2. Instalar la BD y configurar acceso
  - 3. Instalar los drivers para la BD

Creación de una aplicación JDBC [1 | 24]

- **1. Importar los paquetes**
  - Se deben incluir los paquetes con las clases JDBC necesarias.
  - Por lo general resulta suficiente con importar java.sql.*

Creación de una aplicación JDBC [2 | 24]

- **2. Registrar el driver JDBC**
  - Requiere inicializar un driver para que se pueda abrir un canal de comunicación con la BD.
  - El registro de un driver es el proceso mediante el cual el archivo de la clase del driver se carga en memoria para que pueda ser utilizado como una implementación de las interfaces JDBC.

Creación de una aplicación JDBC [3 | 24]

- **2. Registrar el driver JDBC**
  - Hay 2 formas de registrar el driver JDBC:
    - Método `Class.forName()`
    - Método `DriverManager.registerDriver()`

Creación de una aplicación JDBC [4 | 24]

- **2. Registrar el driver JDBC**
  - Ejemplo de `Class.forName()`:

```
try {

Class.forName("oracle.jdbc.driver.OracleDriver");
}
catch(ClassNotFoundException ex) {
   System.out.print("Error al cargar el driver");
   System.exit(1);
}
```

Creación de una aplicación JDBC [5 | 24]

- **2. Registrar el driver JDBC**
  - Ejemplo de `DriverManager.registerDriver()`.

```
try {
                    Driver      miDriver      =      new
oracle.jdbc.driver.OracleDriver();
   DriverManager.registerDriver(miDriver);
}
catch(ClassNotFoundException ex) {
   System.out.print("Error al cargar el driver");
   System.exit(1);
}
```

Creación de una aplicación JDBC [6 | 24]

- **3. Abrir una conexión**
  - Después de cargar del driver, se puede establecer una conexión mediante el método `DriverManager.getConnection()`:
    - `getConnection(String url)`
    - `getConnection(String url, Properties prop)`
    - `getConnection(String url, String usuario, String clave)`

Creación de una aplicación JDBC [7 | 24]

- **3. Abrir una conexión**
  - La URL de una BD es una dirección que apunta a la BD.
  - MySQL: `jdbc:mysql://servidor:puerto/bd`
  - SQL Server: `jdbc:sqlserver://servidor:puerto;DatabaseName=bd`

Creación de una aplicación JDBC [8 | 24]

- **4. Ejecutar una consulta**
  - Requiere usar un objeto del tipo `Statement`, `CallableStatement` o `PreparedStatement` para construir y enviar una sentencia SQL a la BD.
  - Estos objetos también tienen métodos para convertir los tipos de datos de la BD en los de Java.

Creación de una aplicación JDBC [9 | 24]

- **4. Ejecutar una consulta**
  - `Statement`:
    - Se usa para acceso de propósito general a la BD.
    - Útil cuando se usan sentencias SQL estáticas en tiempo de ejecución.
    - No acepta parámetros de entrada.

Creación de una aplicación JDBC [10 | 24]

- **4. Ejecutar una consulta**
  - `PreparedStatement`:
    - Subinterfaz de `Statement`.
    - Se usa cuando se planean ejecutar sentencias SQL múltiples veces.
    - Acepta parámetros de entrada.

Creación de una aplicación JDBC [11 | 24]

- **4. Ejecutar una consulta**
  - `CallableStatement`:
    - Subinterfaz de `PreparedStatement`.
    - Se usa cuando se quiere acceder a procedimientos almacenados.
    - Acepta parámetros de entarada.

Creación de una aplicación JDBC [12 | 24]

- **4. Ejecutar una consulta**
  - Ejemplo de creación de objeto `Statement`:

```
Statement stmt = null;
try {
    stmt = conn.createStatement();
    ...
}
catch (SQLException e) {
    . . .
}
```

Creación de una aplicación JDBC [13 | 24]

- **4. Ejecutar una consulta**
  - Una vez creado un objeto `Statement`, se puede ejecutar la sentencia SQL usando los siguientes métodos:
    - `boolean execute(String SQL)`
    - `int executeUpdate(String SQL)`
    - `ResultSet executeQuery(String SQL)`

Creación de una aplicación JDBC [14 | 24]

- **4. Ejecutar una consulta**
  - `boolean execute(String SQL)`:
    - Devuelve true si se puede obtener un objeto `ResultSet`.
    - Se lo usa para sentencias DDL o cuando se deben ejecutar consultas dinámicas.

Creación de una aplicación JDBC [15 | 24]

- **4. Ejecutar una consulta**
  - `int executeUpdate(String SQL)`:
    - Devuelve la cantidad de filas afectadas por la consulta SQL .
    - Se lo usa cuando se debe recuperar la cantidad de filas afectadas, por ejemplo, sentencias `INSERT`, `UPDATE` o `DELETE`.

Creación de una aplicación JDBC [16 | 24]

- **4. Ejecutar una consulta**
  - `ResultSet executeQuery(String SQL)`:
    - Devuelve un objeto `ResultSet`.
    - Se lo usa cuando se espera obtener un result set, por ejemplo, cuando se ejecuta un `SELECT`.

Creación de una aplicación JDBC [17 | 24]

- **4. Ejecutar una consulta**
  - Ejemplo de creación de objeto `PreparedStatement`:

```
PreparedStatement pstmt = null;
try {
 String SQL="Update Employees SET age= ? WHERE id = ?";
 pstmt = conn.prepareStatement(SQL);
 ...
}
catch (SQLException e) {...}
```

Creación de una aplicación JDBC [18 | 24]

- **4. Ejecutar una consulta**
  - Ejemplo de creación de objeto `PreparedStatement`:
    - Todos los parámetros en JDBC se representan mediante el símbolo ?.
    - Los métodos `setXXX()` pasan los valores a los parámetros (XXX representa el tipo de datos en Java).
    - Cada parámetro se referencia por su posición (el primero es el 1)

Creación de una aplicación JDBC [19 | 24]

- **4. Ejecutar una consulta**
  - Ejemplo de creación de objeto `PreparedStatement`:
    - Los métodos `execute()`, `executeQuery()` y `executeUpdate()` de `Statement` también trabajan con estos objetos.

Creación de una aplicación JDBC [20 | 24]

- **4. Ejecutar una consulta**
  - Ejemplo de creación de objeto `CallableStatement`:

```
CallableStatement cstmt = null;
try {
   String SQL = "{call getEmpName(?, ?)}";
   cstmt = conn.prepareCall(SQL);
   ...
}
catch (SQLException e) { . . .}
```

Creación de una aplicación JDBC [21 | 24]

- **4. Ejecutar una consulta**
  - Ejemplo de creación de objeto `CallableStatement`:
    - Los métodos `setXXX()` pasan los valores a los parámetros.
    - A los parámetros de salida se los debe especificar mediante el método `registerOutputParameter()`. Luego de llamar al procedimiento, se obtienen los valores de los parámetros de salida mediante los métodos `getXXX()`.

Creación de una aplicación JDBC [22 | 24]

- **5. Extraer datos del ResultSet**
  - Para moverse por el objeto ResultSet hay distintos métodos:
    - `first()`
    - `last()`
    - `previous()`
    - `next()`

Creación de una aplicación JDBC [23 | 24]

- **5. Extraer datos del ResultSet**
  - Se debe usar el método `ResultSet.getXXX()` apropiado para obtener los datos del ResultSet.
  - Para cada método `getXXX()` hay 2 versiones: en una se puede especificar una columna por su posición, y en la otra por su nombre.

Creación de una aplicación JDBC [24 | 24]

- **6. Cerrar la conexión**
  - Implica cerrar todos los recursos de la BD.
  - Con el método `close()` se puede cerrar un objeto `Connection`, `Statement`, `PreparedStatement` y `CallableStatement`.

Fuente [1 | 1]

- http://www.tutorialspoint.com/jdbc/index.htm