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 2018

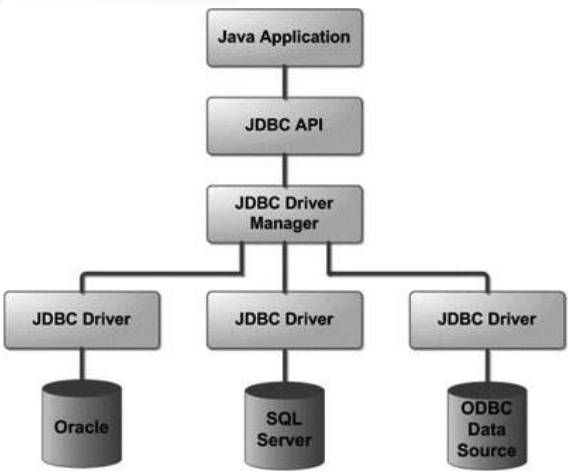
Introducción [1 | 1]

- JDBC
  - Introducción
  - Creación de una aplicación JDBC

Introducción [1 | 6]

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

Introducción [2 | 6]



Introducción [3 | 6]

- **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 | 6]

- **Interfaces:**
  - **`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
  - **`DriverManager`**: maneja una lista de *drivers*. Según los pedidos de conexión de la aplicación Java, busca el driver adecuado

Introducción [5 | 6]

- **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 | 6]

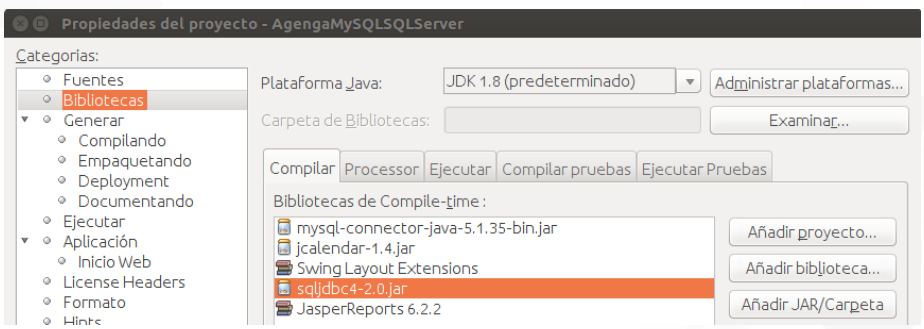
- **Interfaces:**
  - **`ResultSet`**: estos objetos contienen los datos obtenidos de la BD. Actúan como iterador, permitiendo moverse dentro de los mismos
  - **`SQLException`**: maneja todos los errores que puedan ocurrir en la aplicación que interactúa con la BD

Creación de una aplicación JDBC [1 | 23]

- **1. Instalar la librería con el *driver* JDBC para una BD específica**
  - Descargar el *driver* JDBC (conector) apropiado:
    - MySQL: `mysql-connector-java-XXX-bin.jar`
    - SQL Server: `sqljdbcXXX.jar`

Creación de una aplicación JDBC [2 | 23]

- **1. Instalar la librería con el *driver* JDBC para una BD específica**
  - Registrar la librería en el proyecto:



Creación de una aplicación JDBC [3 | 23]

- **2. Registrar el *driver* JDBC**
  - Requiere inicializar el *driver* para poder abrir un canal de comunicación con la BD
  - Hay 2 formas de registrar el driver JDBC:
    - Método `Class.forName()`

      AgendaMySQLSQLServer.prueba.Principal1.java
    - Método `DriverManager.registerDriver()`

      AgendaMySQLSQLServer.prueba.Principal2.java

- El registro del *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.

- De las 2 formas, la primera (`Class.forName()`) es la más usada.

Creación de una aplicación JDBC [4 | 23]

- **3. Abrir una conexión**
  - Después de registrar del *driver*, realizar la conexión a la BD mediante el método `getConnection()`:
    - `getConnection(String url)`
    - `getConnection(String url, Properties prop)`
    - `getConnection(String url, String usuario, String clave)`

AgendaMySQLSQLServer.prueba.Principal3.java

- El método `getConnection()` se encuentra sobrecargado. Siempre se le debe especificar la URL de la BD, y según la configuración de la misma, se le pueden especificar propiedades adicionales o un usuario y clave.

- La URL de una BD es una dirección que apunta a la misma, y depende del motor de BD.

Creación de una aplicación JDBC [5 | 23]

- **3. Abrir una conexión**
  - ¿Cuántas conexiones contra la BD debería abrir una aplicación?

Creación de una aplicación JDBC [6 | 23]

- **4. Crear una consulta**
  - Implica usar objetos del tipo:
    - `Statement`
    - `PreparedStatement`
    - `CallableStatement`

- Estos objetos tienen métodos para convertir los tipos de datos de la BD en los de Java.

Creación de una aplicación JDBC [7 | 23]

- **4. Crear una consulta (`Statement`)**
  - Para consultas de propósito general
  - No acepta parámetros de entrada
  - Usada cuando la consulta se ejecuta una única vez
  - Usada generalmente para sentencias del tipo DDL (`CREATE`, `ALTER`, `DROP`)
  - Muy bajo rendimiento

Creación de una aplicación JDBC [8 | 23]

- **4. Crear una consulta (`PreparedStatement`)**
  - Subinterfaz de `Statement`
  - Acepta parámetros de entrada (consultas parametrizadas)
  - Usada cuando la consulta se ejecuta varias veces
  - Mejor rendimiento que `Statement` (cuando se ejecuta varias veces la misma consulta)

Creación de una aplicación JDBC [9 | 23]

- **4. Crear una consulta (`CallableStatement`)**
  - Subinterfaz de `PreparedStatement`
  - Acepta parámetros de entarada
  - Usada para llamar a procedimientos almacenados y funciones
  - Alto rendimiento

Creación de una aplicación JDBC [10 | 23]

- **4. Crear una consulta (`Statement`)**
  - Creación: método `createStatement()` de `Connection`
  - Ejecución:
    - `boolean execute(...)`
    - `int executeUpdate(...)`
    - `ResultSet executeQuery(...)`

AgendaMySQLSQLServer.prueba.Principal4.java (consultarPersonas_Statement())

- Según la consulta que se desee realizar, se puede usar uno de estos 3 métodos, los cuales se encuentran sobrecargados:
  - `boolean execute(...)`: devuelve true si se puede obtener un ResultSet. Se lo usa para sentencias DDL o cuando se deben ejecutar consultas dinámicas.
  - `int executeUpdate(...)`: 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.
  - `ResultSet executeQuery(...)`: devuelve un ResultSet.

Creación de una aplicación JDBC [11 | 23]

- **4. Crear una consulta (`PreparedStatement`)**
  - Creación: método `prepareStatement(...)` de `Connection`
  - Ejecución:
    - `boolean execute(...)`
    - `int executeUpdate(...)`
    - `ResultSet executeQuery(...)`

AgendaMySQLSQLServer.prueba.Principal4.java (consultarPersonas_PreparedStatement())

- Los métodos `prepareStatement(...)`, `execute(...)`, `executeUpdate(...)` y `executeQuery(...)` están sobrecargados.

Creación de una aplicación JDBC [12 | 23]

- **4. Crear una consulta (`CallableStatement`)**
  - Creación: método `prepareCall(...)` de `Connection`
  - Ejecución:
    - `boolean execute(...)`
    - `int executeUpdate(...)`
    - `ResultSet executeQuery(...)`

AgendaMySQLSQLServer.prueba.Principal4.java (consultarPersonas_CallableStatement())

- Los métodos `prepareCall(...)`, `execute(...)`, `executeUpdate(...)` y `executeQuery(...)` están sobrecargados.

Creación de una aplicación JDBC [13 | 23]

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

Creación de una aplicación JDBC [14 | 23]

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

AgendaMySQLSQLServer.prueba.Principal4.java (procesarResultSet())

Creación de una aplicación JDBC [15 | 23]

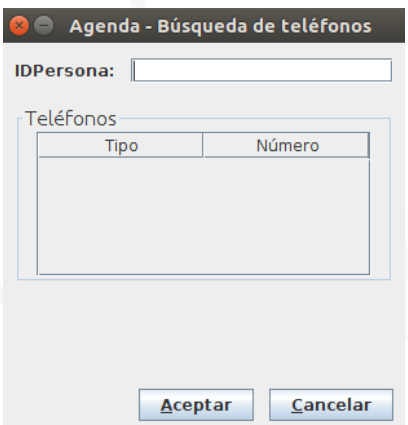
- **6. Cerrar la conexión**
  - Implica cerrar todos los recursos de la BD
  - Con el método `close()` se puede cerrar un objeto `Connection`, `Statement`, `PreparedStatement` y `CallableStatement`

Creación de una aplicación JDBC [16 | 23]

- **Características de los métodos `consultarPersonas_XXX()`:**
  - Propensos a ataques del tipo *SQL Injection*
  - Implican conocer el diseño de la BD (nombres de las tablas, columnas, claves, etc)
  - Se debe otorgar a los usuarios permiso sobre las tablas referenciadas por las consultas
  - Cada cambio implica un despliegue a los clientes

Creación de una aplicación JDBC [17 | 23]

- ***SQL Injection***:



- Suponer que se tiene una ventana como la mostrada en la figura, y lo que se quiere hacer es mostrar los teléfonos de la persona especificada mediante su ID.

Creación de una aplicación JDBC [18 | 23]

- ***SQL Injection* (continuación):**
  - Se lee el campo de texto:

```
String idPersona = this.txtIDPersona.getText().trim();
```

  - Cadena que realiza la consulta:

```
  String cadena = "select IDTelefono, Tipo, Numero from
Telefonos where IDPersona = " + idPersona;
```

Creación de una aplicación JDBC [19 | 23]

- ***SQL Injection* (continuación):**
  - Caso #1: se lee "**1**" del campo de texto

```
cadena = "select IDTelefono, Tipo, Numero from Telefonos where IDPersona = 1"
```

  - Caso #2: se lee "**105 OR 1 = 1**" del campo de texto

```
cadena = "select IDTelefono, Tipo, Numero from Telefonos where IDPersona = 105 OR 1 = 1"
```

AgendaMySQLSQLServer.prueba.Principal5.java (consultarTelefonos_Statement())

- En el segundo caso, como 1 = 1 siempre es True, si se ejecuta la sentencia representada por la cadena, se muestra todo el contenido de la tabla `Telefonos`.

Creación de una aplicación JDBC [20 | 23]

- ***SQL Injection* (continuación):**
  - Una solución para este tipo de problema consiste en usar:
    - Consultas parametrizadas

AgendaMySQLSQLServer.prueba.Principal5.java (consultarTelefonos_PreparedStatement)

    - Procedimientos almacenados

AgendaMySQLSQLServer.prueba.Principal5.java (consultarTelefonos_CallableStatement)

Creación de una aplicación JDBC [21 | 23]

- ***SQL Injection* (continuación):**
  - Los parámetros se representan mediante el símbolo `?`
  - Los métodos `setXXX()` pasan los valores a los parámetros
  - Cada parámetro se referencia por su posición (el primero es el 1)
  - A los parámetros de salida se los debe especificar mediante el método `registerOutputParameter()`

Creación de una aplicación JDBC [22 | 23]

- **Para evitar tener que conocer el diseño de la BD, se pueden emplear:**
  - Vistas (también propensas al problema de SQL Injection)
  - Procedimientos almacenados

AgendaMySQLSQLServer.prueba.Principal6.java (consultarTelefonos_Statement)

Creación de una aplicación JDBC [23 | 23]

- **Para evitar tener que otorgar permisos a los usuarios sobre las tablas, se pueden emplear:**
  - Vistas
  - Procedimientos almacenados

Fuente [1 | 1]

- http://www.tutorialspoint.com/jdbc/index.htm
- http://javaconceptoftheday.com/statement-vs-preparedstatement-vs-callablestatement-in-java/