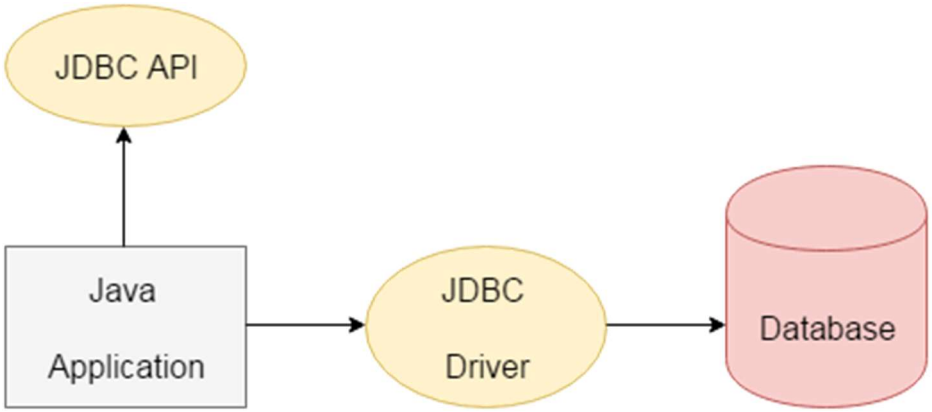
JDBC

JDBC significa Java Database Connectivity. JDBC es una API de Java para conectar y ejecutar la consulta con la base de datos. Forma parte de JavaSE (Java Standard Edition). La API JDBC usa controladores JDBC para conectarse con la base de datos. Hay cuatro tipos de controladores JDBC:

- Controlador de puente JDBC-ODBC,
- Conductor nativo
- Controlador de protocolo de red, y
- Conductor delgado

Hemos discutido los cuatro controladores anteriores en el próximo capítulo.

Podemos usar la API JDBC para acceder a datos tabulares almacenados en cualquier base de datos relacional. Con la ayuda de JDBC API, podemos guardar, actualizar, eliminar y obtener datos de la base de datos. Es como Open Database Connectivity (ODBC) proporcionado por Microsoft.



La versión actual de JDBC es 4.3. Es la versión estable desde el 21 de septiembre de 2017. Se basa en la interfaz de nivel de llamada X / Open SQL. El paquete **java.sql** contiene clases e interfaces para la API JDBC. A continuación se muestra una lista de *interfaces* populares de la API JDBC:

- Interfaz de controlador
- Interfaz de conexión
- Interfaz de declaración
- Interfaz PreparedStatement

- Interfaz CallableStatement
- Interfaz ResultSet
- Interfaz ResultSetMetaData
- Interfaz DatabaseMetaData
- Interfaz RowSet

A continuación se muestra una lista de *clases* populares de API JDBC:

- Clase DriverManager
- Clase blob
- Clase Clob
- Clase de tipos

¿Por qué deberíamos usar JDBC?

Antes de JDBC, la API de ODBC era la API de la base de datos para conectar y ejecutar la consulta con la base de datos. Pero, la API ODBC utiliza el controlador ODBC que está escrito en lenguaje C (es decir, dependiente de la plataforma y no seguro). Es por eso que Java ha definido su propia API (API JDBC) que usa controladores JDBC (escritos en lenguaje Java).

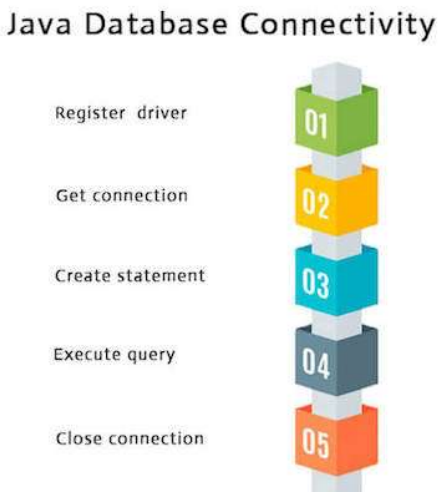
Podemos usar la API JDBC para manejar la base de datos usando el programa Java y podemos realizar las siguientes actividades:

1. Conectarse a la base de datos
2. Ejecute consultas y actualice declaraciones a la base de datos
3. Recupere el resultado recibido de la base de datos.

Conectividad de base de datos Java con 5 pasos

5 pasos para conectarse a la base de datos en java:

1. Registrar la clase de DriverManager
2. Crea el objeto de conexión
3. Crear el objeto de statement
4. Ejecutar la consulta → Procesar los datos
5. Cerrar el objeto de conexión



Conectividad de base de datos Java con MySQL

Para conectar la aplicación Java con la base de datos MySQL, debemos seguir los siguientes 5 pasos.

En este ejemplo estamos usando MySql como la base de datos. Entonces necesitamos saber la siguiente información para la base de datos mysql:

1. **Clase de controlador:** la clase de controlador para la base de datos mysql es **com.mysql.jdbc.Driver** .
2. **URL de conexión:** la URL de conexión para la base de datos mysql es **jdbc: mysql: // localhost: 3306 / sonoo** donde jdbc es la API, mysql es la base de datos, localhost es el nombre del servidor en el que se ejecuta mysql, también podemos usar la dirección IP , 3306 es el número de puerto y sonoo es el nombre de la base de datos. Podemos usar cualquier base de datos, en tal caso, necesitamos reemplazar el sonoo con nuestro nombre de base de datos.
3. **Nombre de usuario:** el nombre de usuario predeterminado para la base de datos mysql es **root** .
4. **Contraseña:** es la contraseña dada por el usuario al momento de instalar la base de datos mysql. En este ejemplo, vamos a usar root como contraseña.

Primero creemos una tabla en la base de datos mysql, pero antes de crear una tabla, primero debemos crear una base de datos.

Clase DriverManager

La clase DriverManager actúa como una interfaz entre el usuario y los controladores. Realiza un seguimiento de los controladores disponibles y se encarga de establecer una conexión entre una base de datos y el controlador apropiado. La clase DriverManager mantiene una lista de clases Driver que se han registrado llamando al método DriverManager.registerDriver ().

Métodos útiles de la clase DriverManager

Método	Descripción
1) public static void registerDriver (controlador de controlador):	se utiliza para registrar el controlador dado con DriverManager.
2) Public static void deregisterDriver (controlador de controlador):	se utiliza para cancelar el registro del controlador dado (descartar el controlador de la lista) con DriverManager.
3) conexión estática pública getConnection (URL de cadena):	se utiliza para establecer la conexión con la url especificada.
4) conexión estática pública getConnection (URL de cadena, nombre de usuario de cadena, contraseña de cadena):	se utiliza para establecer la conexión con la url, el nombre de usuario y la contraseña especificados.

Interfaz connection

Una conexión es la sesión entre la aplicación Java y la base de datos. La interfaz de Connection es una fábrica de Statement, PreparedStatement y DatabaseMetaData, es decir, el objeto de Connection se puede utilizar para obtener el objeto de Statement y DatabaseMetaData. La interfaz de conexión proporciona muchos métodos para la gestión de transacciones como commit (), rollback (), etc.

De manera predeterminada, la conexión confirma los cambios después de ejecutar consultas.

Métodos comúnmente utilizados de la interfaz de conexión:

1) Public Statement createStatement (): crea un objeto de declaración que se puede usar para ejecutar consultas SQL.

2) instrucción pública createStatement (int resultSetType, int resultSetConcurrency): crea un objeto de **instrucción** que generará objetos ResultSet con el tipo y la concurrencia dados.

3) public void setAutoCommit (estado booleano): se utiliza para establecer el estado de confirmación. Por defecto es cierto.

4) public void commit (): guarda los cambios realizados desde el commit / rollback anterior permanente.

5) reversión de void público (): elimina todos los cambios realizados desde la confirmación / reversión anterior.

6) public void close (): cierra la conexión y libera recursos JDBC de inmediato.

Interfaz Statemet

La **interfaz de declaración** proporciona métodos para ejecutar consultas con la base de datos. La interfaz de instrucción es una fábrica de ResultSet, es decir, proporciona un método de fábrica para obtener el objeto de ResultSet.

Métodos comúnmente utilizados de interfaz de declaración:

Los métodos importantes de la interfaz de declaración son los siguientes:

1) Public ResultSet executeQuery (String sql): se utiliza para ejecutar la consulta SELECT. Devuelve el objeto de ResultSet.

2) public int executeUpdate (String sql): se utiliza para ejecutar la consulta especificada, se puede crear, eliminar, insertar, actualizar, eliminar, etc.

3) ejecución pública booleana (String sql): se utiliza para ejecutar consultas que pueden devolver múltiples resultados.

4) public int [] executeBatch (): se usa para ejecutar lotes de comandos.

Interfaz ResultSet

El objeto de ResultSet mantiene un cursor apuntando a una fila de una tabla. Inicialmente, el cursor apunta antes de la primera fila.

Pero podemos hacer que este objeto avance y retroceda pasando TYPE_SCROLL_INSENSITIVE o TYPE_SCROLL_SENSITIVE en el método createStatement (int, int) y también podemos hacer que este objeto sea actualizable mediante:

1) public boolean next ():	se usa para mover el cursor a la fila siguiente desde la posición actual.
2) public boolean previous ():	se usa para mover el cursor a la fila anterior desde la posición actual.
3) público booleano primero ():	se usa para mover el cursor a la primera fila del objeto del conjunto de resultados.
4) public boolean last ():	se usa para mover el cursor a la última fila del objeto del conjunto de resultados.
5) absoluto booleano público (fila int):	se usa para mover el cursor al número de fila especificado en el objeto ResultSet.
6) pariente público booleano (fila int):	se usa para mover el cursor al número de fila relativo en el objeto ResultSet, puede ser positivo o negativo.
7) public int getInt (int columnIndex):	se usa para devolver los datos del índice de columna especificado de la fila actual como int.
8) public int getInt (String columnName):	se usa para devolver los datos del nombre de columna especificado de la fila actual como int.
9) public String getString (int columnIndex):	se utiliza para devolver los datos del índice de columna especificado de la fila actual como String.
10) public String getString (String columnName):	se usa para devolver los datos del nombre de columna especificado de la fila actual como String.

Interfaz PreparedStatement

La interfaz PreparedStatement es una subinterfaz de Statement. Se utiliza para ejecutar consultas parametrizadas.

Veamos el ejemplo de consulta parametrizada:

1. String sql = "insertar en valores emp (?,?,?)" ;

Como puede ver, estamos pasando el parámetro (?) Para los valores. Su valor se establecerá llamando a los métodos de establecimiento de PreparedStatement.

¿Por qué usar PreparedStatement?

Mejora el rendimiento : el rendimiento de la aplicación será más rápido si usa la interfaz PreparedStatement porque la consulta se compila solo una vez.

Métodos de interfaz PreparedStatement

Los métodos importantes de la interfaz PreparedStatement se detallan a continuación:

Método	Descripción
public void setInt (int paramIndex, int value)	establece el valor entero en el índice del parámetro dado.
public void setString (int paramIndex, valor de cadena)	establece el valor de la cadena en el índice del parámetro dado.
public void setFloat (int paramIndex, valor flotante)	establece el valor flotante en el índice del parámetro dado.
public void setDouble (int paramIndex, valor doble)	establece el valor doble en el índice del parámetro dado.
public int executeUpdate ()	ejecuta la consulta Se utiliza para crear, soltar, insertar, actualizar, eliminar, etc.
ResultSet public executeQuery ()	ejecuta la consulta de selección. Devuelve una instancia de ResultSet.

Interfaz Java DatabaseMetaData

La interfaz DatabaseMetaData proporciona métodos para obtener metadatos de una base de datos, como el nombre del producto de la base de datos, la versión del producto de la base de datos, el nombre del controlador, el nombre del número total de tablas, el nombre del número total de vistas, etc.

Métodos comúnmente utilizados de la interfaz DatabaseMetaData

- **public String getDriverName () genera SQLException:** devuelve el nombre del controlador JDBC.
- **public String getDriverVersion () lanza SQLException:** devuelve el número de versión del controlador JDBC.
- **public String getUserName () arroja SQLException:** devuelve el nombre de usuario de la base de datos.
- **public String getDatabaseProductName () arroja SQLException:** devuelve el nombre del producto de la base de datos.
- **public String getDatabaseProductVersion () arroja SQLException:** devuelve la versión del producto de la base de datos.
- **getTables público ResultSet (Catálogo de cadenas, String schemaPattern, String tableNamePattern, String [] tipos) arroja SQLException:** devuelve la descripción de las tablas del catálogo especificado. El tipo de tabla puede ser TABLE, VIEW, ALIAS, SYSTEM TABLE, SYNONYM, etc.

Interfaz Java ResultSetMetaData

Los metadatos significan datos sobre datos, es decir, podemos obtener más información de los datos.Si tiene que obtener metadatos de una tabla como el número total de columnas, nombre de columna, tipo de columna, etc., la interfaz ResultSetMetaData es útil porque proporciona métodos para obtener metadatos del objeto ResultSet.

public int getColumnCount () lanza SQLException	devuelve el número total de columnas en el objeto ResultSet.
public String getColumnName (int index) produce SQLException	devuelve el nombre de columna del índice de columna especificado.
public String getColumnTypeName (int index) produce SQLException	devuelve el nombre del tipo de columna para el índice especificado.
public String getTableName (int index) produce SQLException	devuelve el nombre de la tabla para el índice de columna especificado.

Interfaz Java CallableStatement

La interfaz CallableStatement se usa para llamar a los **procedimientos y funciones almacenados** .

Podemos tener lógica empresarial en la base de datos mediante el uso de procedimientos almacenados y funciones que mejorarán el rendimiento porque están precompilados.

Supongamos que necesita obtener la edad del empleado en función de la fecha de nacimiento, puede crear una función que reciba la fecha como entrada y devuelva la edad del empleado como salida.

¿Cuál es la diferencia entre los procedimientos almacenados y las funciones?

Las diferencias entre los procedimientos almacenados y las funciones se detallan a continuación:

Procedimiento almacenado	Función
Se utiliza para realizar la lógica de negocios.	Se utiliza para realizar el cálculo.
no debe tener el tipo de retorno.	debe tener el tipo de retorno.
puede devolver 0 o más valores.	puede devolver solo uno de los valores.
Podemos llamar a funciones desde el procedimiento.	El procedimiento no se puede llamar desde la función.
El procedimiento admite parámetros de entrada y salida.	La función solo admite parámetros de entrada.
El manejo de excepciones usando el bloque try / catch se puede usar en procedimientos almacenados.	El manejo de excepciones usando try / catch no se puede usar en funciones definidas por el usuario.

Gestión de transacciones en JDBC

La transacción representa **una sola unidad de trabajo** .

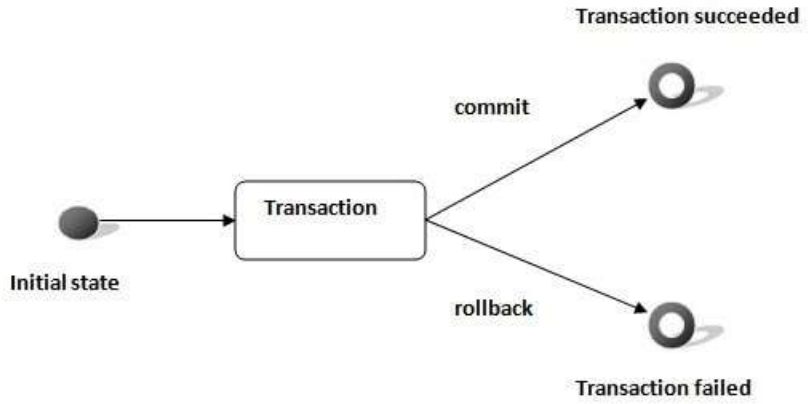
Las propiedades de ACID describen bien la gestión de transacciones. ACID significa atomicidad, consistencia, aislamiento y durabilidad.

Atomicidad significa que todo fue exitoso o ninguno.

La consistencia asegura llevar la base de datos de un estado consistente a otro estado consistente.

El aislamiento asegura que la transacción esté aislada de otra transacción.

La durabilidad significa que una vez que se ha comprometido una transacción, seguirá siéndolo, incluso en caso de errores, pérdida de potencia, etc.



Ventaja de Transaction Mangaement

rendimiento rápido Hace que el rendimiento sea rápido porque la base de datos se ve afectada en el momento de la confirmación.

En JDBC, la **interfaz de conexión** proporciona métodos para gestionar transacciones.

Método	Descripción
void setAutoCommit (estado booleano)	Es cierto por defecto significa que cada transacción se confirma por defecto.
void commit ()	confirma la transacción
revertir nulo ()	cancela la transacción.

Procesamiento por lotes en JDBC

En lugar de ejecutar una sola consulta, podemos ejecutar un lote (grupo) de consultas. Hace que el rendimiento sea rápido.

Las interfaces java.sql.Statement y java.sql.PreparedStatement proporcionan métodos para el procesamiento por lotes.

Ventaja del procesamiento por lotes

Rendimiento rápido

Métodos de interfaz de declaración

Los métodos requeridos para el procesamiento por lotes se dan a continuación:

Método	Descripción
addBatch vacío (consulta de cadena)	Agrega consulta en lote.
int [] executeBatch ()	Ejecuta el lote de consultas.

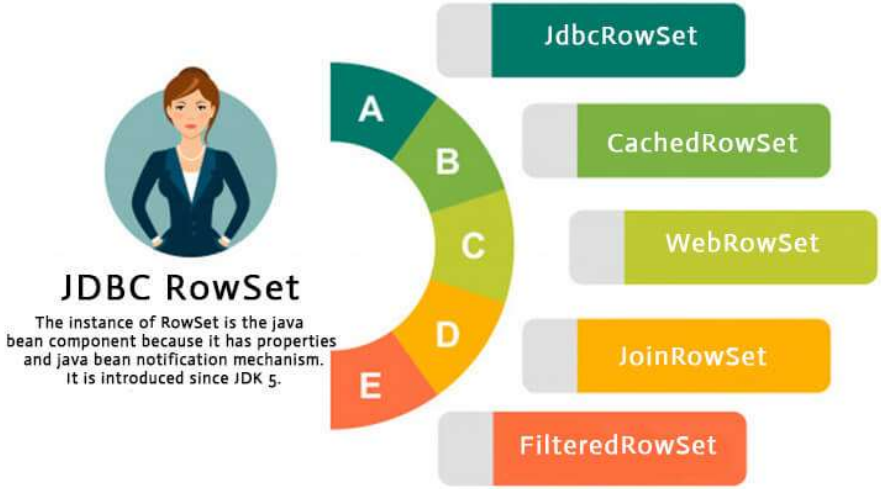
JDBC RowSet

La instancia de **RowSet** es el componente de Java Bean porque tiene propiedades y mecanismo de notificación de Java Bean. Se introduce desde JDK 5.

Es el contenedor de ResultSet. Contiene datos tabulares como ResultSet pero es fácil y flexible de usar.

Las clases de implementación de la interfaz RowSet son las siguientes:

- JdbcRowSet
- CachedRowSet
- WebRowSet
- JoinRowSet
- FilteredRowSet



Es la nueva forma de obtener la instancia de JdbcRowSet desde JDK 7.

Ventaja de RowSet

Las ventajas de usar RowSet se dan a continuación:

1. Es fácil y flexible de usar.
2. Es desplazable y actualizable por defecto