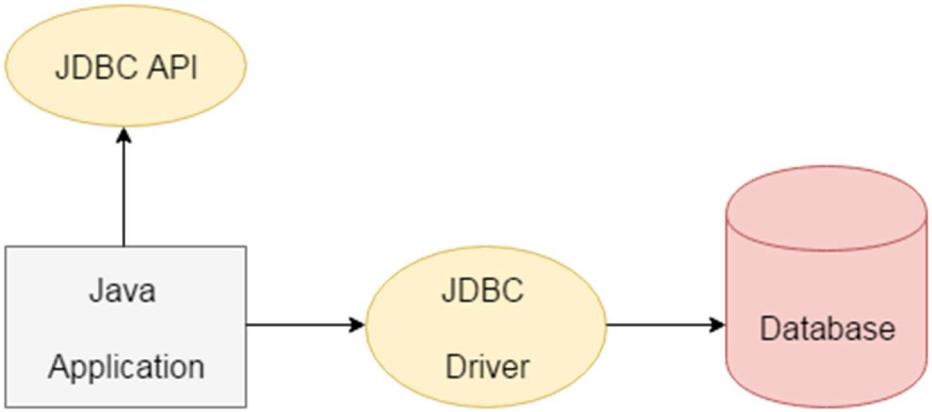
JDBC

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).

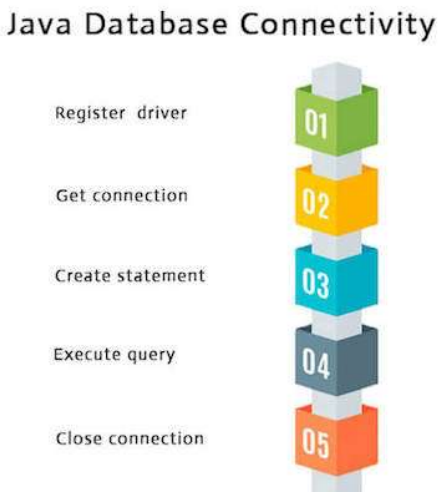
Podemos usar la API JDBC para manejar la base de datos usando el programa Java y podemos realizar las siguientes actividades:

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.