JDBC

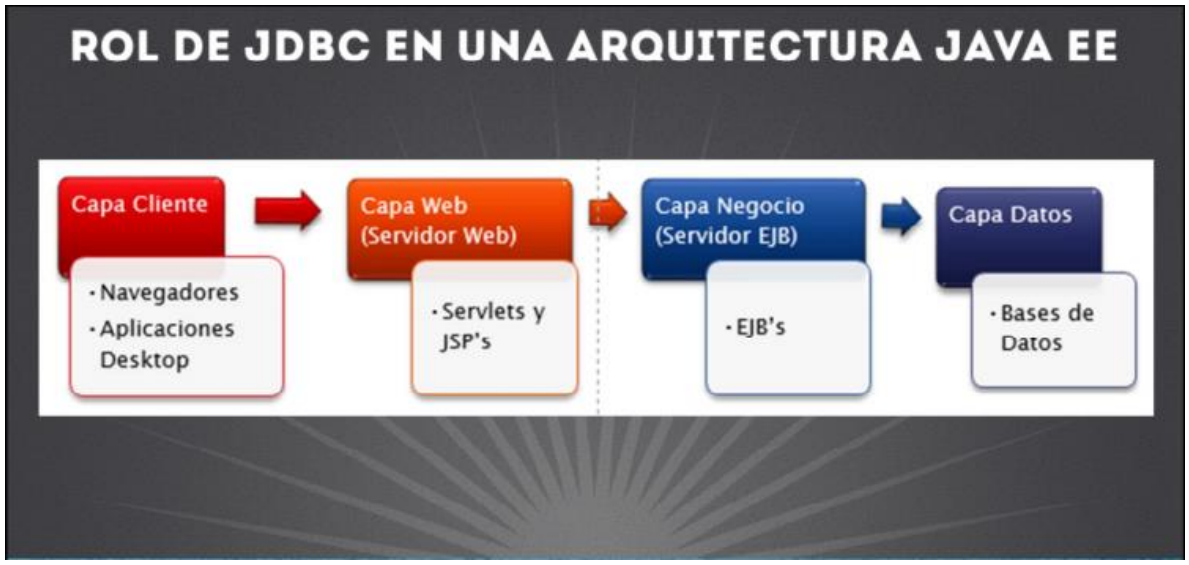
En esta lección, que todavía es introductoria vamos a revisar ya algunas características del API JDBC. En primer lugar definiremos que es un API, que es JDBC y veremos cuáles son los pasos generales para utilizar JDBC con el objetivo consultar y modificar la información de una base de datos relacional, sin importar cual sea la base de datos relacional que deseemos utilizar, como puede ser los motores de base de datos de MySql, Oracle, Microsoft SQL Server, PostgreSql, o cualquier otro motor de base de datos.

Un API (Application Program Interface), es un conjunto de clases, en este caso clases Java, las cuales normalmente se encuentran empaquetadas en un archivo conocido como JAR (Java Archive File). Estas clases cumplen una función muy específica, por ejemplo la tarea de conectarse a una base de datos en particular, como puede ser MySql, Oracle, etc. Para este curso utilizaremos el API de JDBC. Ahora veremos qué es JDBC.

JDBC (Java Database Connectivity) es un API estándar en Java, y se utiliza para conectarnos a una base de datos. El API de JDBC define un conjunto de interfaces y clases Java, las cuales deben ser implementadas por cada proveedor de base de datos con el objetivo de brindar la funcionalidad para consultar, modificar y muchas tareas más sobre la base de datos de cada proveedor, ejemplo de estos proveedores son Oracle, MySql, PostgreSql, etc.

JDBC nos permite, entre muchas otras cosas, realizar las operaciones básicas CRUD (Create-Insert, Read-Select, Update y Delete) sobre una base de datos.

En resumen, cada manejador de base de datos implementa la especificación de JDBC, y cada manejador de base de datos proporciona su propio controlador (driver) JDBC.



En cursos anteriores comentamos que existen distintas versiones de Java para el desarrollo de aplicaciones. La versión empresarial Java EE nos permite crear aplicaciones muy robustas y que soporten una gran cantidad de usuarios utilizando nuestras aplicaciones. Sin embargo para crear una aplicación empresarial conlleva varios retos, uno de ellos es crear varias capas lógicas en nuestra aplicación.

Como podemos observar en la figura, una aplicación empresarial en Java, se compone de distintas capas de información. Cada capa tiene una función muy especifica. Dividir una aplicación en capas tiene varias ventajas, como son separación de responsabilidades, mejores mantenimientos a la aplicación y especialización de los programadores en cada capa.

La BD (Base de Datos) almacena la información de la aplicación empresarial. JDBC nos permite comunicarnos a través de la capa de datos con la Base de Datos. JDBC es un API que se utiliza en la capa de acceso a datos, por lo que su rol es muy importante en una aplicación Java Empresarial.

Existen otras tecnologías como Servlets y JSPs, o EJBs, sin embargo estas tecnologías las estudiaremos en otros cursos. En este curso nos enfocaremos en crear una capa de datos lo más robusta posible, aplicando patrones de diseño y las mejores prácticas con el objetivo de que la capa de datos que creemos sea la base para sus propias aplicaciones Java del mundo real.

Ahora vamos a revisar los pasos básicos para utilizar JDBC.

1. En primer lugar lo que tenemos que hacer es descargar el driver o controlador de JDBC dependiendo de la base de datos que vamos a utilizar.

2. Una vez que hemos descargado nuestro driver, agregamos el controlador descargado al classpath de la aplicación. Cabe resaltar que el classpath de la aplicación lo único que significa es que es la ruta donde van a encontrarse las clases o archivos .jar que va a utilizar nuestra aplicación. La palabra classpath se compone de *class* que hace referencia a las clases y *path* que significa un camino o ruta.

3. Como tercer paso vamos a crear una clase de Java y en esta clase necesitamos realizar los siguientes pasos:

   a) Vamos agregar código para registrar el driver JDBC respectivo
   b) Posteriormente agregamos el código necesario para crear una conexión a la base de datos deseada,
   c) Luego creamos un objeto Statement, que nos va a permitir ejecutar el query que deseemos. Recordemos que un query o consulta de SQL es el que nos permitirá consultar o modificar la información de la base de datos que estemos utilizando.
   d) El siguiente paso es ejecutar la sentencia SQL que hayamos seleccionado y podemos procesarla dependiendo del tipo de sentencia que hayamos ejecutado
   e) Como último paso vamos a cerrar la conexión de base de datos que hayamos abierto, así como cualquier otro objeto como es el objeto Statement o cualquier otro que hayamos utilizado para ejecutar nuestras sentencias SQL.

Estos son los pasos más básicos para poder utilizar JDBC en nuestras aplicaciones Java. Posteriormente vamos a crear un ejemplo para conectarnos a nuestra base de datos MySQL que ya hemos instalado.