# JDBC Prepared Statement y su manejo

5 septiembre, 2018 por Cecilio Álvarez Caules — 4 comentarios

El uso de **JDBC Prepared Statement** es hoy en día prácticamente obligatorio. Aún así hay muchas veces que nos olvidamos de de como usarlos ya que son los frameworks de persistencia **los que los utilizan de forma transparente por  nosotros.** Aún así hay situaciones en las cuales nos podemos encontrar con la necesidad de hacer uso de ellos de forma directa. **¿Cómo funcionan exactamente los JDBC Prepared Statement ?**

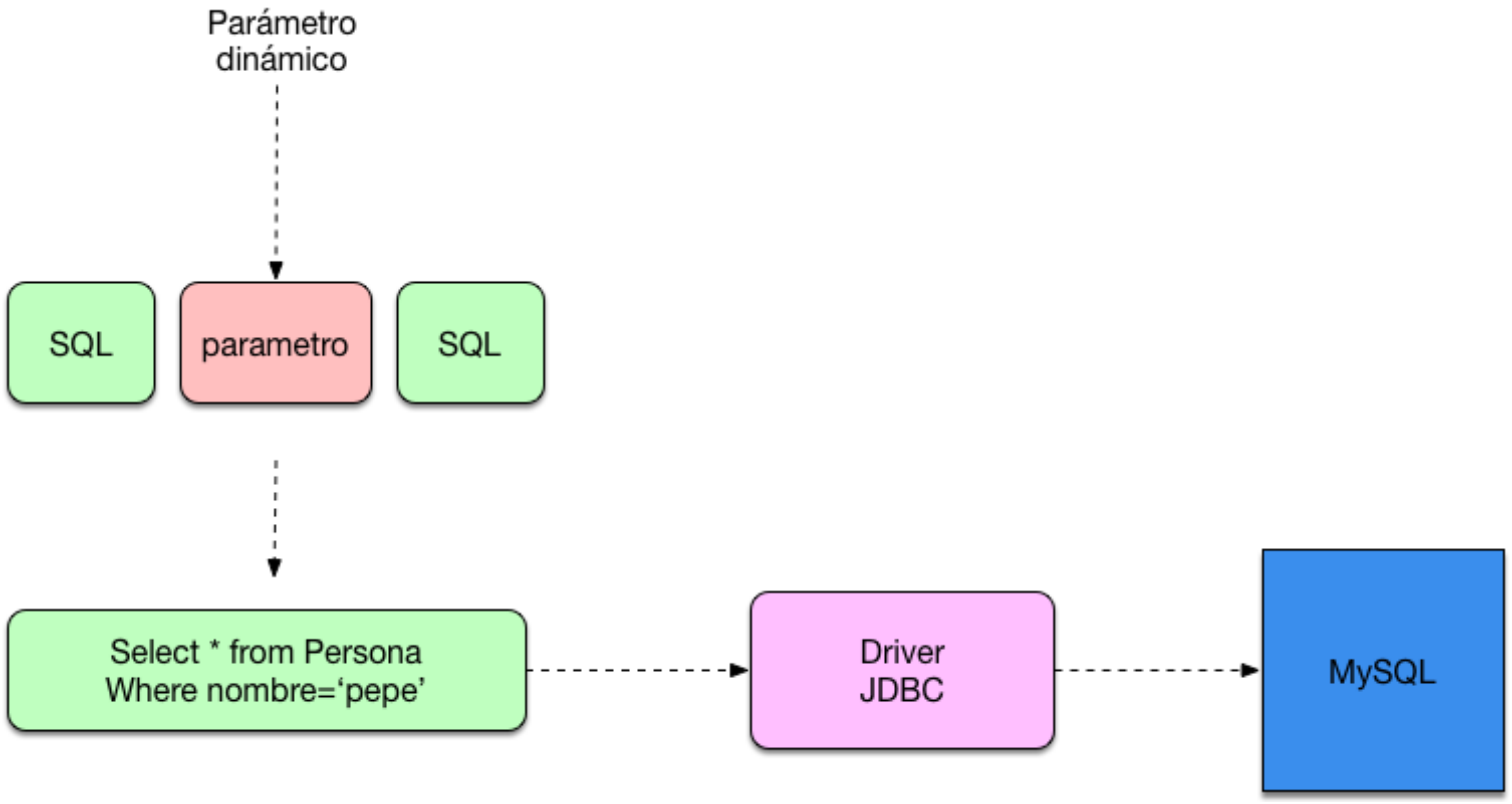
## Utilizando JDBC Prepared Statement

Lo primero que tenemos que entender es cual es la diferencia **entre un Statement de JDBC y un Prepared Statement**. Cuando nosotros construimos una consulta normal de JDBC

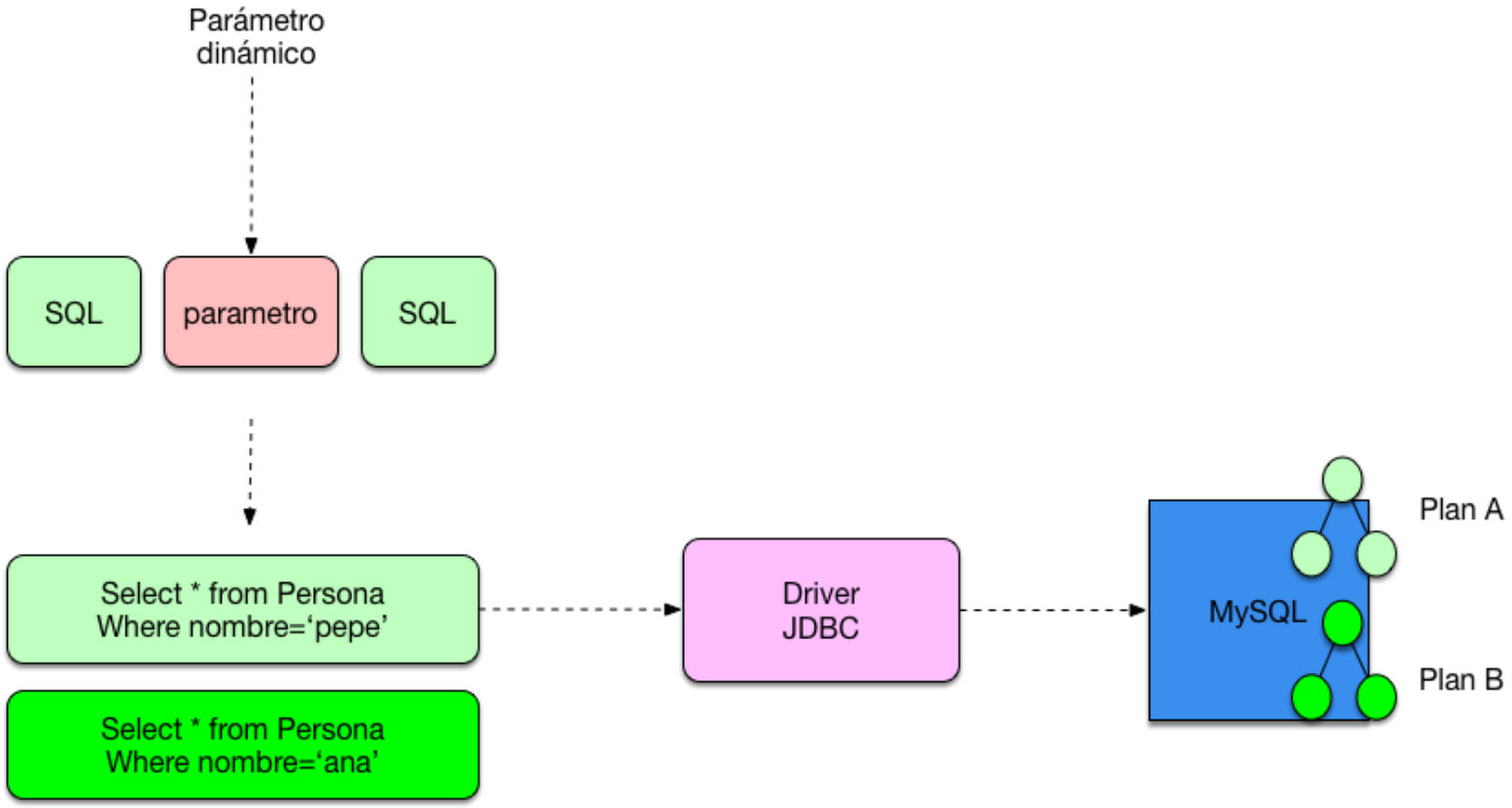
utilizamos un Statement, este Statement o sentencia lo que se encarga es de definir una consulta SQL a ejecutar contra el motor de la base de datos.

```
1  Connection conexion = DriverManager.getConnection("jdbc:mysql://lo
2  Statement sentencia = conexion.createStatement();
3  String nombre="pepe";
4  String consulta = "select * from Persona where nombre='"+nombre+"'
5  ResultSet rs=sentencia.executeQuery(consulta);
```

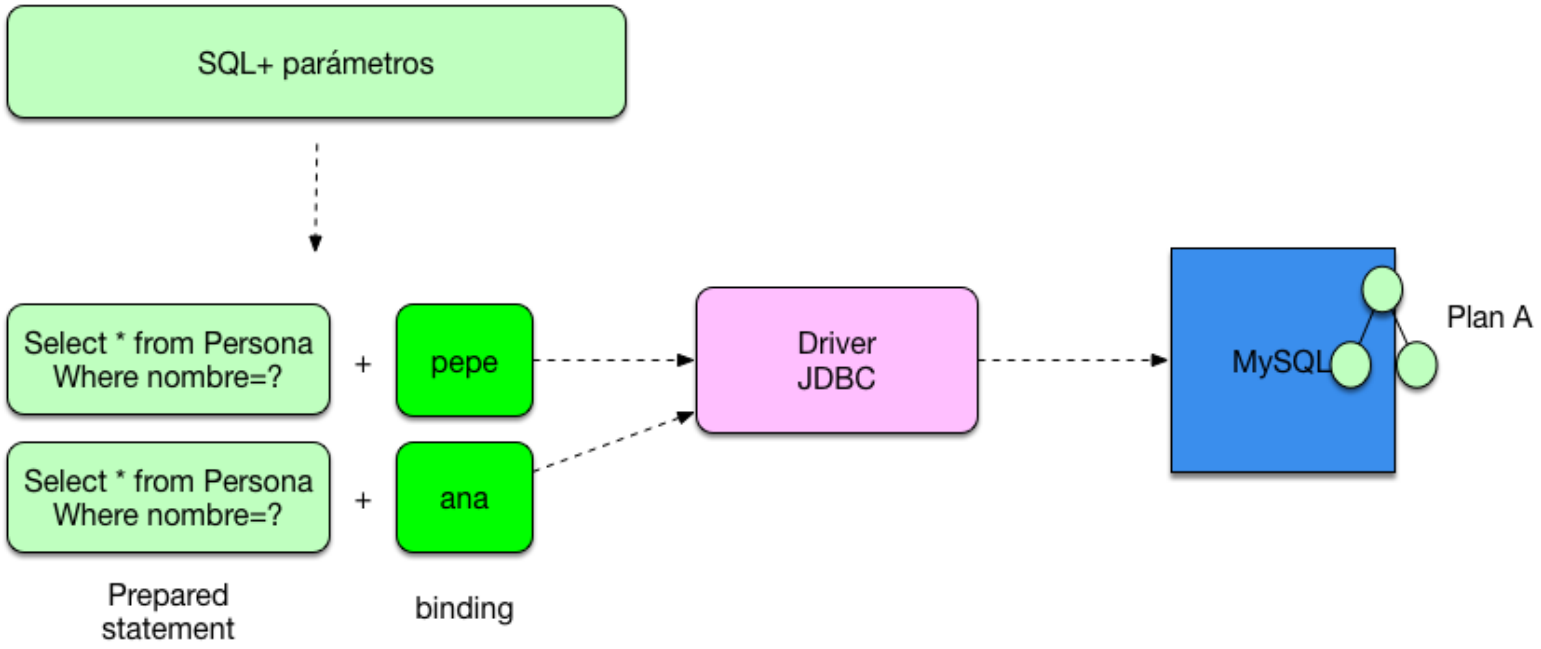
En este caso estamos construyendo una sentencia y aportando **un parámetro a la consulta de forma dinámica**. Esto básicamente se convierte en una consulta SQL que nosotros ejecutamos vía el driver JDBC contra la base de datos.




Muchas veces se nos olvida que para cada consulta SQL que construimos contra la base de datos se construye un plan de ejecución en el que la base de datos decide como esa consulta se ejecuta.




A dos consultas diferentes **se crean dos planes de ejecución diferentes** aunque ambas consultas sean realmente muy similares y únicamente entre e**n juego el valor del parámetro que las pasamos** . Para solventar este problema existen **los JDBC Prepared statement** . Estas estructuras permiten mantener **las consultas neutras** sin tener en cuenta los parámetros que se las pasa ya que realiza un binding de ellos




De esta forma cuando la base de datos genera un hash para el plan de ejecución ambas consultas , la que consulta por pepe y la que consulta por ana devuelven **el mismo hash y comparten el plan de ejecución**. Vamos a ver esto en código:

```
1  String consulta = "select * from Persona where nombre = ? ";
2  Connection conexion= DriverManager.getConnection("jdbc:mysql://loc
3  PreparedStatement sentencia= conexion.prepareStatement(consulta);
4  sentencia.setString(1, "pepe");
5  ResultSet rs = sentencia.executeQuery();
```

No solo nos estaremos ahorrando la construcción de planes de ejecución sino que también de la misma manera estamos evitando que nos inyecten SQL **ya que al parametrizar la**

**consulta el API de JDBC nos protege contra las este tipo de ataques.**Normalmente el uso de consultas parametrizadas mejora el rendimiento entre un 20 y un 30 % a nivel de base de datos.

## JDBC Prepared Statement y Logs

Hay situaciones en las que necesitamos realizar un log de la consulta SQL , al tratarse de una sentencia preparada solemos hacer un log del String algo como por ejemplo

```
1 String consulta = "select * from Persona where nombre = ? ";
2
3 log(consulta);
```

Esto a veces puede ser un problema ya que perdemos los parámetros que se pasan . Sin embargo muchos drivers soportan el log de la propia consulta SQL con los parámetros ya aplicados realizando un log de la propia sentencia

```
1 String consulta = "select * from Persona where nombre = ? ";
2     PreparedStatement ps = con.prepareStatement(consulta);
3     ps.setString(1, nombre);
4     log(ps);
```

Tengámoslo en cuenta , en este caso podremos ver salir por la consola algo del siguiente estilo:

com.mysql.jdbc.JDBC42PreparedStatement@67424e82: select  * from Persona where nombre='juan'

El parámetro aparece.

Otros artículos relacionados

1. Creando un Java 8 Custom Stream con JDBC ResultSets
2. JDBC ResultSet Types y su funcionamiento
3. JDBC, Java try with resources
4. JDBC



Archivado en: JPA
Etiquetado como: JavaeeTips

## Comentarios

Ludwing Pérez dice

7 septiembre, 2018 en 4:54

Hola Cecilio!

Sabes si es necesario usar el patrón Singleton (y cual sería la mejor forma de usarlo) para el manejo de los objetos Connection, Statement o PreparedStatement.

Saludos!

Responder

Cecilio Álvarez Caules dice
11 septiembre, 2018 en 20:59

Ninguna debe ser singleton , a ver si explico un poco esto mas a detalle

Responder

Bladimir Guardado dice

6 septiembre, 2018 en 0:37

Hola Cecilio,
Disculpa tu por casualidad sabes como puedo pasar parametros desde Java a un JSP, ya valido en Java si el usuario tiene una session iniciada pero necesito mostrarle un mensaje en pantalla que ya posee una session en el navegador XXXX, pero esto seria usando swal de java. si me puedes guiar te lo agradecere.

Saludos

Responder

Cecilio Álvarez Caules dice
6 septiembre, 2018 en 15:40

almacenaria un hashmap en el servlet context con los usuasrios que tienen inicializada sessión si ese usuario se logea y ya esta en la lista , pues es que estaba repetido y no le valido o le aviso que pierde la otra

Responder

## Deja un comentario

Tu dirección de correo electrónico no será publicada. Los campos obligatorios están marcados con *

Comentario

Nombre *

Correo electrónico *

Web

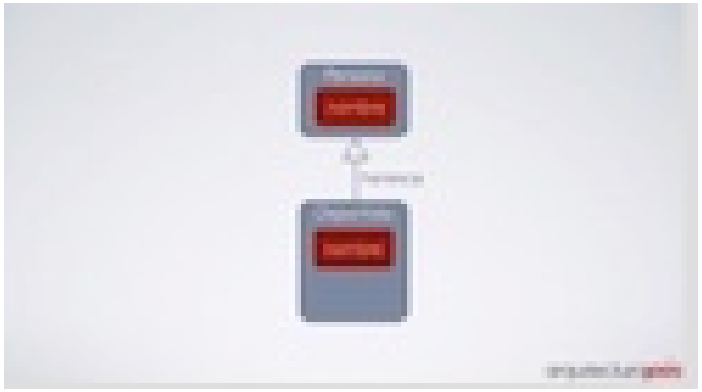
PUBLICAR COMENTARIO

Este sitio usa Akismet para reducir el spam. Aprende cómo se procesan los datos de tus comentarios.

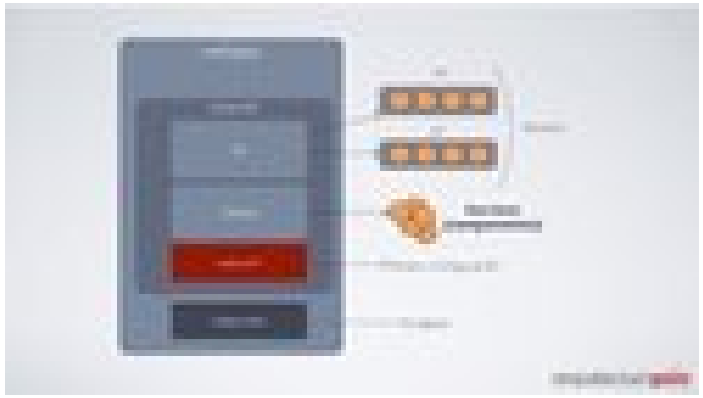
BUSCAR

Buscar en esta web

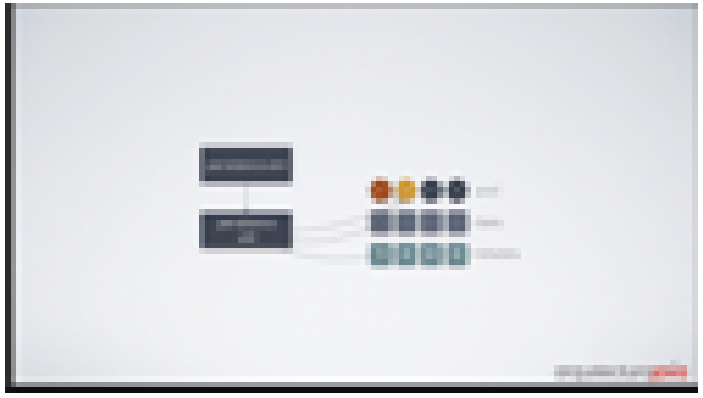
Mis Cursos de Java Gratuitos



Java Herencia Java JDBC

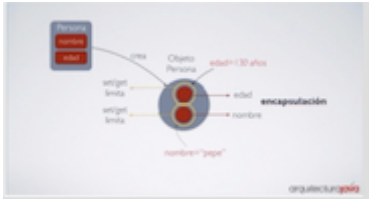


Servlets Intro JPA

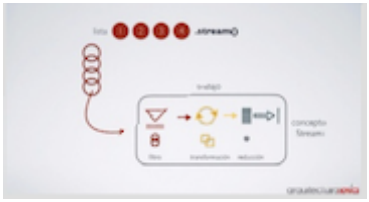


## Mis Cursos de Java

### Programación Orientada a Objeto en Java

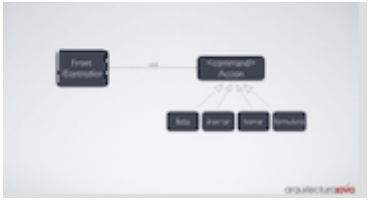


### Java APIS Core

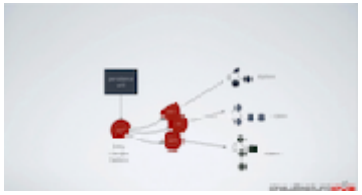


## Java Web

## Pack Java Core



## Arquitectura Java Solida con Spring



## POPULAR

Spring Boot JPA y su configuración

Los Frameworks y su lado oscuro

Spring REST CORS y su configuración

Static Method vs instance method y su uso correcto

Java new String y la creación de objetos

Arquitecturas RESTFul y agregados

Single Page Application y REST

Spring Boot JSP y su configuración

@RepositoryRestResource y Spring Framework

Angular Services Singletons o no?

## CONTACTO

contacto@arquitecturajava.com

## LO MAS LEIDO

¿Qué es Spring Boot?

Java Constructores this() y super()

El Principio de Substitución de Liskov

Usando Java Session en aplicaciones web

Ejemplo de Java Singleton (Patrones y ClassLoaders)

Introducción a Servicios REST

Java Iterator vs ForEach

Comparando java == vs equals

Usando el patron factory

Java Override y encapsulación

REST JSON y Java

Ejemplo de JPA , Introducción (I)

El modelo Entidad Relacion con DBDesigner

Uso de Java Generics (I)

¿Cuales son las certificaciones Java?

¿Qué es Gradle?

¿Qué es un Microservicio?

Eclipse JPA y clases de dominio

Mis Libros

El patrón de inyección de dependencia y su utilidad

Spring MVC Configuración (I)

¿ Que es REST ?

Angular ngFor la directiva y sus opciones

Java Finally y el cierre de recursos

Java Stream forEach y colecciones