Desarrollo Avanzado de Aplicaciones I

Introducción a JDBC 2

idat

i d a t

inicio desarrollo aplicación término

idat

Inicio

Logro de aprendizaje - Introducción

idat

Logro de Aprendizaje

"Al finalizar la sesión, el participante podrá implementar una conexión a una Base de Datos MySQL y utilizar funciones CRUD a través de Procedimientos Almacenados en la Base de Datos.

idat

Introducción

idat

- Revisión Rápida de Temas de Sesión Anterior
- Revisión de Ejercicios de Sesión Anterior
- Inquietudes y/o Preguntas
- Agenda de Sesión
  - Interface PreparedStatement
  - Procedimientos Almacenados
  - Interface CallableStatement
  - Ejemplos y Ejercicios

Desarrollo

Desarrollo del Contenido de la Sesión

idat

La Interface Statement

- ✓ La Interface “Statement” se usa para enviar sentencias SQL a la base de datos. Actúa como contenedor para la ejecución de las sentencias SQL en una conexión dada.
- ✓ Hay tres tipos de interfaces “Statement”:
  - ➢ Statement, se usa para ejecutar una sentencia SQL simple sin parámetros. Suministra métodos básicos para ejecutar sentencias y devolver resultados.
  - ➢ PreparedStatement que hereda de Statement, se usa para ejecutar sentencias SQL precompiladas con o sin parámetros IN. Añade métodos para trabajar con los parámetros IN.
  - ➢ CallableStatement que hereda de PreparedStatement, se usa para ejecutar un procedimiento almacenado de base de datos. Añade métodos para trabajar con parámetros OUT.

La Interface PreparedStatement

- ✓ La interface “PreparedStatement” nos proporciona soporte para ejecutar sentencias SQL con parámetros dinámicos, de tal forma que se optimice su ejecución en la base de datos.
- ✓ A diferencia de la interface “Statement”, donde para cada sentencia la base de datos usa un plan de ejecución diferente, con la interface “PreparedStatement” el plan de ejecución es el mismo así tenga parámetros con valores diferentes.
- ✓ La interface “PreparedStatement” se crea a partir del método “prepareStatement” de una conexión activa.
- ✓ La interface “PreparedStatement” tiene métodos “setTIPODATO” para configurar los parámetros IN de la sentencia SQL.

La Interface PreparedStatement (cont.)

idat

- ✓ La interface "PreparedStatement" nos ahorra la construcción de planes de ejecución en la base de datos y también nos protege evitando que nos inyecten SQL a nuestro programa.

```
conn = DriverManager.getConnection("jdbc:mysql://localhost/sakila?" +
        "user=root&password=admin");
pstmt = conn.prepareStatement("SELECT first_name, last_name FROM actor "
        + "WHERE actor_id = ?");
pstmt.setInt( i: 1,  i1: 1);
rs = pstmt.executeQuery();
while (rs.next()) {
    System.out.println(rs.getString( string: "first_name") + " " +
            rs.getString( string: "last_name"));
```

```
PENELOPE GUINESS
```

Procedimiento Almacenado: Introducción

- ✓ Los procedimientos almacenados, también conocidos como “Stored Procedure”, son un conjuntos de instrucciones escritas en el lenguaje SQL.
- ✓ Su objetivo es realizar una tarea determinada, desde operaciones sencillas hasta tareas muy complejas.
- ✓ La característica fundamental de los procedimientos almacenados es que estos comandos se quedan almacenados y se ejecutan en el servidor o en el motor de bases de datos. Este aspecto permite que las aplicaciones clientes las ejecuten directamente mediante llamada a una API.
- ✓ Los procedimientos almacenados se han diseñado para aligerar a las aplicaciones clientes, pudiendo ejecutar directamente en el servidor aquellas tareas pesadas y que necesitan muchos recursos.

Procedimiento Almacenado: Características

- ✓ Para crear y gestionar los procedimientos almacenados en MySQL se utilizan los comandos: “Create Procedure”, “Alter Procedure”, “Drop Procedure” y “Delimiter”.
- ✓ Para ejecutar un procedimiento almacenado se usa el comando “Call”
- ✓ Pueden recibir y devolver parámetros.
- ✓ Pueden manejar tablas, ejecutando operaciones e iteraciones de lectura/escritura.
- ✓ Pueden devolver una tabla como resultado.
- ✓ Se almacenan en la base de datos en la cual se crean.
- ✓ No dependen de ninguna tabla en particular.
- ✓ Pueden aceptar recursividad.

Procedimiento Almacenado: Ventajas

idat

- ✓ Compatibilidad: los procedimientos almacenados permiten ejecutar la misma operación sobre la base de datos también en el caso de que las aplicaciones cliente estén implementadas en distintas plataformas y programadas en distintos lenguajes.
- ✓ Seguridad: cuando necesitamos evitar el acceso directo a la base de datos, los procedimientos almacenados permiten establecer un entorno seguro, otorgando permisos y privilegios para la ejecución.
- ✓ Rendimiento: con los procedimientos almacenados todo el trabajo se ejecuta en el servidor, por lo cual se minimiza de forma importante la cantidad de información intercambiada con las aplicaciones cliente y se reduce el tráfico de acceso a la base de datos además del número de accesos.

Procedimiento Almacenado: Ventajas (cont.)

- ✓ Centralización: los procedimientos almacenados permiten centralizar la lógica funcional ofreciendo a todas las aplicaciones clientes la misma versión actualizada, lo que hace el mantenimiento más sencillo.
- ✓ Sencillez: los procedimientos almacenados permiten la creación de procedimientos o funciones, lo que facilita mucho el trabajo del programador ya que se trata de características compartidas por los principales lenguajes de programación modernos.
- ✓ Reutilización del código: al escribir un mismo código que se ejecuta por todas las aplicaciones, es posible reducir las inconsistencias.
- ✓ Integridad: gracias a los procedimientos almacenados es posible centralizar el acceso a la información.

Creación de Procedimiento Almacenado: Parámetro OUT

idat

```
1   DELIMITER $$
2   CREATE DEFINER=`root`@`localhost` PROCEDURE `city_by_country`(in p_country varchar(50), out p_count int)
3   BEGIN
4       select count(*)
5       into p_count
6       from city as a, country as b
7       where a.country_id = b.country_id
8       and country = p_country;
9   END$$
10  DELIMITER ;
11  set @p_count = 0;
12  call sakila.city_by_country('Peru', @p_count);
13  select @p_count;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

@p_count
4

La Interface CallableStatement

- ✓ La interface “CallableStatement” nos proporciona soporte para ejecutar procedimientos almacenados con parámetros dinámicos.
- ✓ La interface “CallableStatement” es una extensión de la interface “PreparedStatement” por lo que da soporte para parámetros de entrada IN y adicionalmente proporciona soporte para parámetros de salida OUT y de entrada/salida INOUT.
- ✓ La interface “CallableStatement” se crea a partir del método “prepareCall” de una conexión activa y que tiene como parámetro la sentencia SQL “Call procedure(args)”.
- ✓ La interface “CallableStatement” tiene métodos “setTIPODATO” para configurar los parámetros IN e INOUT y el método “registerOutParameter” para configurar los parámetros OUT e INOUT.

Ejemplo de la Interface CallableStatement: Parámetro OUT


```
Connection conn = null;
CallableStatement cstmt = null;
ResultSet rs = null;
String pais = "Peru";
try {
    conn = DriverManager.getConnection("jdbc:mysql://localhost/sakila?" +
            "user=root&password=admin");
    cstmt = conn.prepareCall( string:"CALL city_by_country(?,?)");
    cstmt.setString( i:1,  string:pais);
    cstmt.registerOutParameter( i:2,  i1:java.sql.Types.INTEGER);
    cstmt.execute();
    int cont = cstmt.getInt( i:2);
    System.out.println(pais + " tiene " + cont + " ciudades registradas.");
}
```

Parámetro IN

Parámetro OUT

```
run:
Peru tiene 4 ciudades registradas.
BUILD SUCCESSFUL (total time: 0 seconds)
```

Creación de Procedimiento Almacenado: Múltiples Filas

idat

```
1   DELIMITER $$
2   CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_actors_by_film`(IN p_cont INT)
3       READS SQL DATA
4   BEGIN
5       SELECT f.film_id, title, count(*)
6       FROM film_actor as fa, actor as a, film as f
7       WHERE fa.actor_id=a.actor_id and fa.film_id=f.film_id
8       GROUP BY f.film_id, title
9       HAVING count(*)>p_cont
10      ORDER BY title;
11  END$$
12  DELIMITER ;
13  call sakila.sp_actors_by_film(12);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

film_id	title	count(*)
87	BOONDOCK BALLROOM	13
146	CHITTY LOCK	13
188	CRAZY HOME	13
249	DRACULA CRYSTAL	13
508	LAMBS CINCINATTI	15
606	MUMMY CREATURES	13
714	RANDOM GO	13

Ejemplo de la Interface CallableStatement: Múltiples Filas

idat

```
Connection conn = null;
CallableStatement cstmt = null;
ResultSet rs = null;
try {
    conn = DriverManager.getConnection("jdbc:mysql://localhost/sakila?" +
            "user=root&password=admin");
    cstmt = conn.prepareCall( string:"CALL sp_actors_by_film(?)");
    cstmt.setInt( i:1,  il:12);
    cstmt.execute();
    rs = cstmt.getResultSet();
    while (rs.next()) {
        System.out.println(rs.getInt( i:1) + ", " +
                rs.getString( i:2) + ", " + rs.getInt( i:3));
    }
}
```

Parámetro IN

Conjunto de Resultados

```
87, BOONDOCK BALLROOM, 13
146, CHITTY LOCK, 13
188, CRAZY HOME, 13
249, DRACULA CRYSTAL, 13
508, LAMBS CINCINATTI, 15
606, MUMMY CREATURES, 13
714, RANDOM GO, 13
```

Aplicación

Revisar ejemplos y realizar ejercicios prácticos

idat

Término

Indicaciones generales y/o Resumen de Sesión

idat

Resumen de Sesión

idat

- Interface PreparedStatement
- Procedimientos Almacenados
- Interface CallableStatement
- Ejemplos y Ejercicios

GRACIAS