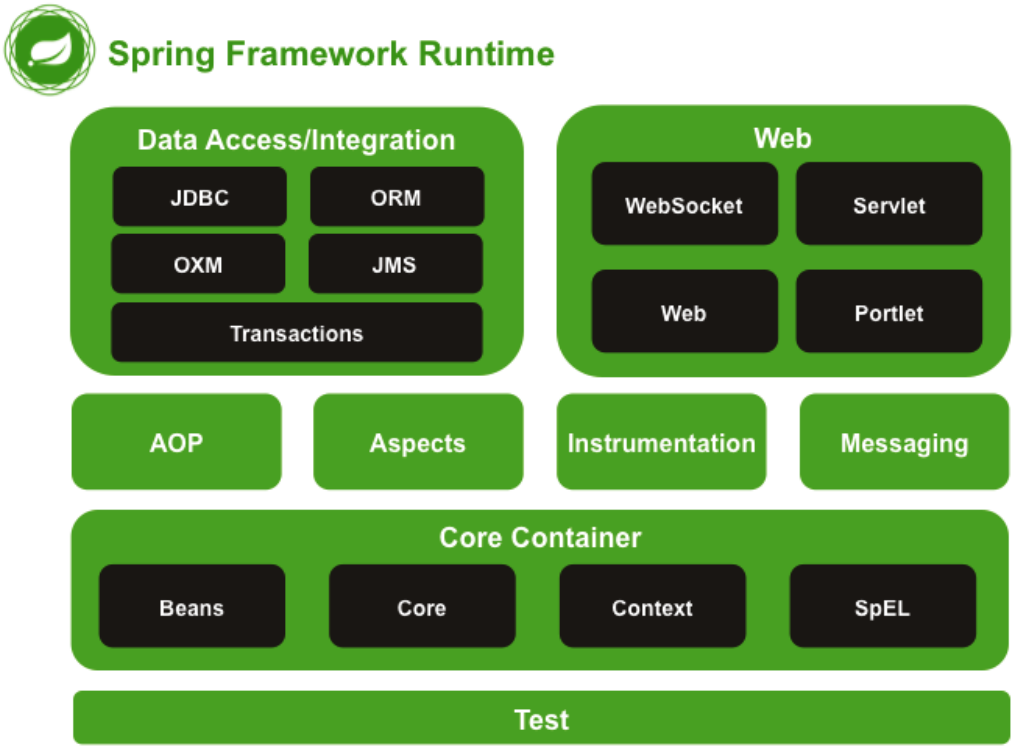
LibrosDigitales.net
Libros para Profesionales de Informática

SPRING FRAMEWORK

MODULO 04

SPRING JDBC



TEMA 05

PROCEDIMIENTOS ALMACENADOS

Eric Gustavo Coronel Castillo
gcoronelc.blogspot.com
I N S T R U C T O R

CONTENIDO

CASOS DE IMPLEMENTAR

Caso 1: Consultar Saldo de una Cuenta

Mediante el uso de un procedimiento se procederá a consultar el saldo a una cuenta, el objetivo es ilustrarte el uso de parámetros de entrada y salida.

A continuación se tiene el script para crear el procedimiento:

```
DELIMITER $$

DROP PROCEDURE IF EXISTS usp_saldo_cuenta$$

CREATE PROCEDURE usp_saldo_cuenta
(IN p_cuenta char(8), OUT p_saldo decimal(12,2))
BEGIN

   select dec_cuensaldo into p_saldo
   from cuenta
   where chr_cuencodigo = p_cuenta;

END$$

DELIMITER ;
```

A continuación se tiene una ejecución exitosa:

```
mysql> CALL usp_saldo_cuenta('00200002',@saldo);
Query OK, 1 row affected (0.00 sec)

mysql> select @saldo;
+---------+
| @saldo  |
+---------+
| 6800.00 |
+---------+
1 row in set (0.00 sec)
```

A continuación se tiene una ejecución fallida:

```
mysql> CALL usp_saldo_cuenta('00200008',@saldo);
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> select @saldo;
+--------+
| @saldo |
+--------+
|   NULL |
+--------+
1 row in set (0.00 sec)
```

En el caso de la ejecución fallida el resultado es NULL.

Caso 2: Registrar un Retiro

Enunciado

Para ilustrar el uso de procedimientos almacenados se implementará el proceso "Registrar Retiro", para lo cual se necesita previamente verificar si la cuenta tiene saldo suficiente.

Suponiendo que los movimientos tienen un costo, también se registrar el costo que involucra la operación.

Implementación

El script es el siguiente:

```
DELIMITER $$

DROP PROCEDURE IF EXISTS usp_retiro$$

CREATE PROCEDURE usp_retiro( p_cuenta char(8), p_importe decimal(12,2),
   p_clave char(6), p_empleado char(4) )
BEGIN
   -- Variables
   DECLARE moneda char(2);
   DECLARE costoMov decimal(12,2);
   DECLARE cont int;
   DECLARE saldo decimal(12,2);
```

```
-- Control de errores
DECLARE EXIT HANDLER FOR SQLEXCEPTION, SQLWARNING, NOT FOUND
BEGIN
   -- Cancela la transacción
   rollback;
   -- Retorna el estado
   select -1 as code, 'Error en el proceso Registrar Retiro.'  as message;
END;

-- Iniciar transacción
start transaction;

-- Leer datos de la cuenta
select int_cuencontmov, chr_monecodigo, dec_cuensaldo
into cont, moneda, saldo
from cuenta
where chr_cuencodigo = p_cuenta
and chr_cuenclave = p_clave
for update;

-- Costo de Transacción
select dec_costimporte into costoMov
from costomovimiento
where chr_monecodigo = moneda;

-- Verifica saldo
if saldo < (p_importe + costoMov) then
   SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Saldo insuficiente';
end if;

-- Registrar el depósito
update cuenta
set dec_cuensaldo = dec_cuensaldo - p_importe - costoMov,
   int_cuencontmov = int_cuencontmov + 2
where chr_cuencodigo = p_cuenta;

-- Registrar el movimiento
set cont := cont + 1;
insert into movimiento(chr_cuencodigo,int_movinumero,dtt_movifecha,
   chr_emplcodigo,chr_tipocodigo,dec_moviimporte,chr_cuenreferencia)
   values(p_cuenta,cont,current_date,p_empleado,'004',p_importe,null);

-- Registrar el costo del movimiento
set cont := cont + 1;
insert into movimiento(chr_cuencodigo,int_movinumero,dtt_movifecha,
   chr_emplcodigo,chr_tipocodigo,dec_moviimporte,chr_cuenreferencia)
   values(p_cuenta,cont,current_date,p_empleado,'010',costoMov,null);
```

```
   -- Confirma Transacción
   commit;

   -- Retorna el estado
   select 1 as code, 'Proceso ok' as message;

END$$

DELIMITER ;
```

Prueba

A continuación tienes una ejecución exitosa:

```
mysql> call usp_retiro('00100001',100, '123456', '0001');
+------+------------+
| code | message    |
+------+------------+
|    1 | Proceso ok |
+------+------------+
1 row in set (0.04 sec)

Query OK, 0 rows affected (0.04 sec)
```

Y la siguiente ejecución es cuando no hay saldo suficiente:

```
mysql> call usp_retiro('00100001',100000, '123456', '0001');
+------+---------------------------------------+
| code | message                               |
+------+---------------------------------------+
|   -1 | Error en el proceso Registrar Retiro. |
+------+---------------------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

EJECUCIÓN DIRECTA

Fundamentos

Puedes usar cualquiera de los métodos **`query`** de **`JdbcTemplate`** o el método **`update`** para ejecutar el procedimiento almacenado, teniendo en cuenta si retorna o no un resultado.

En caso que no retorne ningún resultado:

```
jdbcTemplate.update("call procedimiento (?, ?, ...)", parámetros);
```

En caso retorne una sola fila de datos:

```
Map<String, Object> rec;
rec = jdbcTemplate.queryForMap("call procedimiento (?, ?, ...)", parámetros);
```

En caso retorne una lista de objetos **`Map`**:

```
List<Map<String, Object>> lista;
lista = jdbcTemplate.queryForList("call procedimiento (?, ?, ...)", parámetros);
```

Ejemplo

El siguiente método ejecuta el procedimiento "**`usp_retiro`**" para procesar el retiro de una cuenta:

```
public void retiro(String cuenta, double importe, String clave, String codEmp) {
    Map<String, Object> rec;
    rec = jdbcTemplate.queryForMap("call usp_retiro (?, ?, ?, ?)", cuenta,
            importe, clave, codEmp);
    if (rec.get("code").toString().equals("-1")) {
      throw new RuntimeException(rec.get("message").toString());
    }
  }
```

CLASE STOREPROCEDURE

Fundamentos

La clase **`StoredProcedure`** te brinda mayores opciones de control para ejecutar procedimientos almacenados.

Para implementarlo debes crear una clase que herede de **`StoredProcedure`**:

```
  public class ProcedureSaldoCuenta extends StoredProcedure {

     . . .

  }
```

Ejemplo

El siguiente ejemplo implementa la ejecución del procedimiento **`usp_saldo_cuenta`**:

```
package pe.egcc.springjdbc.dao.impl;

import java.sql.Types;
import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.SqlOutParameter;
import org.springframework.jdbc.core.SqlParameter;
import org.springframework.jdbc.object.StoredProcedure;
import org.springframework.stereotype.Repository;

@Repository
public class ProcedureSaldoCuenta extends StoredProcedure {

  private static final String PROC_NAME = "usp_saldo_cuenta";

  @Autowired
  public ProcedureSaldoCuenta(JdbcTemplate jdbcTemplate) {
    super(jdbcTemplate, PROC_NAME);
    setFunction(false);
    declareParameter(new SqlParameter("p_cuenta", Types.VARCHAR));
```

```
    declareParameter(new SqlOutParameter("p_saldo", Types.DECIMAL));
    compile();
  }

  public double ejecutar(String cuenta) {
    Map<String, Object> rec = super.execute(cuenta);
    if (rec.get("p_saldo") == null) {
      throw new RuntimeException("Cuenta no existe.");
    }
    return Double.parseDouble(rec.get("p_saldo").toString());
  }

}
```

A través del método **`ejecutar`** le pasas la cuenta y obtienes el saldo o una excepción en caso que la cuenta no exista.