Working with a CallableStatement

Kevin Jones

@kevinrjones

What's in This Module

Use a CallableStatement to execute a stored procedure

Show how to pass data to the CallableStatement

Show how to retrieve data from the CallableStatement

Setup

Our database has four stored procedures

Get a list of all the acts

Report on what gigs are running when

Tell us the total sales

Try and raise the ticket price

These allow us to show IN, OUT and IN/OUT parameters

CallableStatement Syntax

Called Procedure Syntax is:

```
{ call procedure_name() }
```

Where the procedure_name is the name of the stored procedure in the database

```
create procedure GetActs()
begin
    select acts.name, acts.recordlabel
    from acts
    where acts.recordlabel IS NOT null

    order by acts.name;
end
```

Stored Procedure

This gets all the acts with a record label

```
var sql = "{ call GetActs() }";

try (CallableStatement cs = conn.prepareCall(sql)) {

    var rs = cs.executeQuery();

    while (rs.next()) {

        var name = rs.getString("name");

        var recordLabel =

          rs.getString("recordlabel");

        System.out.println(name + " "

                         + recordLabel);

    }

}
```

◀ **Create the SQL statement (note the { call })**

◀ **Prepare the call**

◀ **It's a query so execute it**

◀ **Iterate over the result set**

◀ **Get the values**

◀ **Use the values**

CallableStatement with IN Parameters

Use CallableStatement to call stored procedures

Set IN parameters just like PreparedStatement

```
create procedure GigReport(IN startdate Date, IN enddate Date)
begin
    select gigs.date, acts.name 'Act', acts.recordlabel, venues.name 'Venue', ticketssold,
           venues.capacity
    from gigs join acts on acts.id = gigs.actid
             join venues on venues.id = gigs.venueid
    where date >= startdate and date <= enddate
    order by gigs.date;
end;
```

Stored Procedure

This generates a 'Gig Report' between dates

It has 2 IN parameters

```
var sql = "{ call GigReport(?, ?) }";

try (CallableStatement cs = conn.prepareCall(sql)) {

    cs.setDate("startdate", ...);

    cs.setDate("enddate",...);

    var rs = cs.executeQuery();


    while (rs.next()) {

        var date = rs.getDate("date");

        ...

    }

}
```

◄ **Create the SQL statement (note the '?')**

◄ **Prepare the call**

◄ **Set the parameters**

◄ **Execute the query**

◄ **Use the data from the columns in the stored procedure**

Using OUT Parameters

Can Use the ?= syntax

- { ?= call sproc_name(?) }
- This is optional
- Not all JDBC drivers support this

Register the out parameters

```
create procedure GetTotalSales(OUT sales decimal(8, 2))

begin

    select sum(currentvalue) 'totalsales' from

        (select ticketssold, price, ticketssold*price 'currentvalue'

         from gigs) salestable

    into sales;

end;
```

Stored Procedure

This returns the sum of all the sales in the database

It has 1 OUT parameter

```
var sql = "{call GetTotalSales(?) }";

try (CallableStatement cs = conn.prepareCall(sql)) {


    cs.registerOutParameter(1,

                        Types.DECIMAL);

    var result = cs.execute();

    System.out.println("Total sales is: "

                  + cs.getDouble(1));


}
```

◂ **Create the SQL statement (note the '?')**

◂ **Prepare the call**

◂ **Register any out parameters**

◂ **Execute the query**

◂ **Use the data from the out parameter**

Using INOUT Parameters

A 'mixture' of IN and OUT calls

Use '?' for each parameter

For IN parameters set a value

For INOUT parameters set a value

Register the INOUT parameters

SetNewPrice Stored Procedure

This tries to update the sales price for a gig

It has 2 IN parameters

It has 1 OUT parameter

```
create procedure SetNewPrice(IN gigid int, IN percentage decimal(8,2), inout maxprice decimal(8,2))
begin
    declare gigprice decimal(8,2) default 0.0;    declare proposedprice decimal(8,2);
    set gigprice = (select max(price) from gigs where id = gigid);
    set proposedprice = gigprice + (gigprice * percentage);
    if (proposedprice < maxPrice)
    then
        set maxprice = proposedprice;
        update gigs set price = proposedprice where id = gigid;
    else
        set maxprice = gigprice;
    end if;
end;
```

```
var sql = "{call SetNewPrice(?, ?, ?) }";

try (CallableStatement cs = conn.prepareCall(sql)) {


  cs.setInt(1, 1);

  cs.setDouble(2, 0.1);

  cs.setDouble(3, 12.0);

  cs.registerOutParameter(3,
                        Types.DECIMAL);

    var result = cs.execute();

    System.out.println("New price: " +
                      cs.getDouble(3));

}
```

◂ **Create the SQL statement (note the '?')**

◂ **Prepare the call**

◂ **Set the IN values**

◂ **Set the value for the INOUT parameter**

◂ **Also register that as an OUT parameter**

◂ **Execute the query**

◂ **Use the data from the out parameter**

Summary

Use a CallableStatement to execute stored procedures

Have a specific syntax – { call ... }

Can have IN, OUT and IN/OUT parameters

Can set parameters by name or column

Columns are 1 based

Out parameters must be registered