Làm việc với Statement, PreparedStatement và CallableStatement

*Đăng bởiduaca23/09/2010 12:11 AM*

Làm việc với **Statement**, **PreparedStatement** & **CallableStatement**

**I/ - Khái quát vấn đề**

Khi làm việc với **JDBC**, bạn không thể không biết đến các giao diện (***interface***) hỗ trợ cho quá trình khai thác và lưu trữ thông tin trên cơ sở dữ liệu của chương trình chính là **Statement**,**PreparedStatement** và **CallableStatement**. Tùy theo nhu cầu, bạn có thể tạo ra các đối tượng của một trong các giao diện trên để thực hiện nhu cầu khai thác thông tin của chương trình tại thời điểm cần thiết. Có thể hiểu về các giao diện trên như mô tả dưới đây

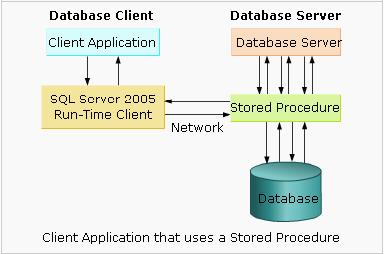
**Statement** thường dùng để tạo ra các đối tượng để phục vụ cho mục đích chính là thực thi các câu lệnh dạng SQL để tác động lên cơ sở dữ liệu. **Ví dụ :** khi bạn cần thực thi 1 câu lệnh SLQ để nhận về thông tin lấy được từ Database như danh sách chứa thông tin của nhân viên, học sinh, hàng hóa … chứa trong 1 Table nào đó thuộc cơ sở dữ liệu. Hoặc giả như bạn có nhu cầu nhập thông tin (*Insert into*) cho 1 table nào đó trong cơ sở dữ liệu bạn cũng cần phải tạo ra 1 đối tượng loại Statement để tác động lên Database. Không chỉ như vậy trong những tinh huống đặc biệt, cần phải tạo hay xóa các **function**, **stored procedure**, **trigger** hay **view** bạn cũng có thể dùng đối tượng thuộc giao diện **Statement** để tương tác với cơ sở dữ liệu của chương trình. Nói tóm lại, 1 đối tượng của giao diện **Statement** sẽ giúp bạn thực hiện được hầu như tất cả các tác động của chương trình lên cơ sở dữ liệu tại thời điểm mà chương trình thi hành.

Tuy nhiên, khả năng thay đổi giá trị được xem như các tham số để tác động lên cơ sở dữ liệu của các đối tượng thuộc dạng này không cao, ví dụ: bạn muốn lấy danh sách các sinh viên thuộc khoa toán, câu lệnh SQL có thể là “**Select** ho, ten, ngaysinh, diem **from** sinhVien **where** khoa=’Toan’”. Vậy trong trường hợp bạn muốn lấy danh sách sinh viên thuộc khoa văn thì câu lệnh sẽ là “**Select**ho, ten, ngaysinh, diem **from** sinhVien **where** khoa=’Van’”. Vấn đề ở đây, ta thấy rằng với 2 nhu cầu trên, nội dung của câu SQL gần như không thay đổi, chỉ khác là điều kiện truy vấn ứng với nhu cầu tại mỗi thời điểm cụ thể không giống nhau nhưng ta phải sử dụng đến 2 chuỗi lệnh SQL. Điều này dường như rất thiếu linh hoạt, vậy câu hỏi đặt ra trong tình huống có nhiều hơn 2 nhu cầu truy vấn, ví dụ như nhu cầu truy vấn hàng hóa theo loại hàng, nhà cung cấp, … trong 1 chương trình quản lý hàng hóa của siêu thị thì phải làm thế nào ?. Bạn sẽ phải có rất nhiều câu truy vấn không mấy khác nhau, điều này làm tốn kém rất nhiều về chi phí, tài nguyên và làm giảm hiệu suất thực thi của chương trình khi cần phải giải quyết những nhu cầu cụ thể nào đó trên thực tế. Đó là chưa nói đến vấn đề bảo trì chương trình, hoặc giả như có nhu cầu về điều chỉnh đối với chương trình sau 1 thời gian hoạt động trên thực tế, việc tìm và điều chỉnh lãi các câu SQL là rất vất vả.

Để khắc phục nhược điểm đó của giao diện **Statement**, Java cung cấp 1 giao diện khác giúp cho việc tạo ra các truy vấn lên cơ sở dữ liệu có vẻ “***động***” hơn chính là **PreparedStatement**. Giao diện **PreparedStatement** được triển khai (*implement*) từ giao diện **Statement** ban đầu nhưng cho phép lập trình viên có thể quy định truyền vào những giá trị cần thay đổi cho câu lệnh SQL để có thể sử dụng 1 cách linh hoạt hơn. Ví dụ để truy vấn và lấy ra danh sách các sinh viên trong trường tùy thuộc theo khoa tại mỗi thời điểm, bạn có thể tạo câu SQL thế này “**Select** ho, ten, ngaysinh, diem **from** sinhVien **where** khoa = ? ”. Ở đây, câu lệnh SQL mà bạn sử dụng làm tham số truyền vào khi tạo đối tượng cho giao diện **PreparedStatement** có thể quy định tham số (*dấu* **?**) để sau này khi gọi sử dụng; bạn có thể truyền giá trị vào sau, tùy theo thời điểm mà giá trị đó có thể cụ thể là gì, điều này đã giúp cho bạn giảm đi đáng kể số lượng câu truy vấn trong chương trình của mình.

Với việc truyền tham số để sử dụng cho đối tượng dạng **PreparedStatement**, bạn có thể dùng các phương thức ***setInt***, ***setString***, ***setFloat***, ***setBoolean***, ***setLong***, … để truyền vào giá trị được coi là tham số của câu lệnh SQL đã tạo ra trước đó tại vị trí của dấu ?. Ví dụ tương ứng với nhu cầu truy vấn danh sách theo khoa đã đề cập ở trên, khi cần lấy ra danh sách sinh viên khoa toán, bạn chỉ cần gọi phướng thức ***setString*** của đối tượng **PreparedStatement** như sau .***setString***(1,”Toan”), tương tự như vậu với nhu cầu cho danh sách khoa văn, khoa lý, bạn cũng có thể sử dụng lệnh .***setString***(1,”Van”); .***setString***(1,”Ly”); (**giải thích:** tham số thứ nhất trong phương thức này chính là thứ tự của tham số muốn truyền – *thứ tự của dấu ? trong câu truy vấn*, trong câu SQL ví dụ của chúng ta có 1 tham số và  như thế tham số cần truyền nằm ở vị trí thứ nhất, việc sử dụng ***setString***, ***setInt***, … tùy thuộc vào dữ liệu mà bạn truyền vào thuộc kiểu gì, trong tình huống trên, dữ liệu của tôi thuộc kiểu chuỗi nên sử dụng phương thức ***setString***)

Việc sử dụng các câu lệnh SQL để truyền giữa **client** (*máy chạy chương trình của chúng ta viết*) và**database server** (*Máy có cài đặt DBMS và chứa Database của chương trình*) sẽ làm cho lưu lượng truyền thông trên mạng rất lớn (*Ta thử hình dung phải thường xuyên thực hiện các truy vấn phức tạp với nhiều table, số lượng thông tin từ các field cần lấy trên 10 fields với điều kiện truy vấn phức hợp and, or … hay các truy vấn có điều kiện là các truy vấn con khác, số lượng ký tự trong chuỗi SQL sẽ rất nhiều*). Điều này làm tăng **traffic** trên mạng dẫn đến tốc độ thực thi của chương trình giảm, tốn kém tài nguyên tại **database server** vì cứ luôn phải cấp phát cho việc tạo và giải phóng bộ nhớ khi thực thi các lệnh SQL, cho dù là lệnh đó được thực thi nhiều lần.



**Stored procedure** là giải pháp tốt cho tình huống này, việc sử dụng **stored procedure** để hỗ trợ cho quá trình xử lý thông tin trong database tỏ ra khá hiệu quả bởi 1 số ưu điểm sau:

-          Có thể gọi thực thi nhiều công việc tác động lên Database chỉ thông qua 1 cái tên.

-          Giảm traffic do chỉ truyền 1 chuỗi ký tự ngắn chính là tên của stored procedure mà không phải truyền đi về những câu lệnh SQL dài dòng, đôi khi lên tới cả trăm ký tự.

-          **Stored procedure** được tạo và đặt tại database server và việc khai thác chỉ thông qua tên của nó cho nên có thể ngăn ngừa được những kẻ tò mò muốn biết chương trình của chúng ta đang tác động như thế nào lên Database.

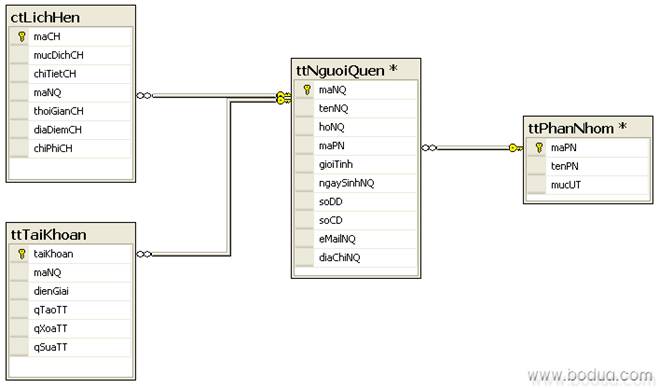
-          Do **stored procedure** được biên dịch trước và tồn tại mãi trong quá trình **DBMS** hoạt động nên khả năng thực thi và hồi đáp thông tin cho **client** rất nhanh, tăng hiệu suất thực thi của chương trình (*Phần biên dịch đối với tập lệnh của****stored procedure****được tái sử dụng nhiều lần*)

-          Tận dụng việc thực thi và giải quyến các yêu cầu của chương trình ở đồng thời cả 2 nơi :**client – server**. Vì thế tăng hiệu suất thực thi chương trình.

**Stored procedure** có nhiều ưu điểm như thế, vậy để gọi sử dụng các **stored procedure** trong Database ta sẽ dùng đối tượng nào ?. Chắc các bạn cũng đã đoán ra, đúng thế !. Java cung cấp cho chúng ta giao diện **CallableStatement** để có thể gọi các **stored procedure** trong Database (**Lưu ý**: **CallableStatement** không chứa **stored procedure**, nó chỉ chứa lời gọi hàm trỏ đến**stored procedure** đã tồn tại trong DBMS để yêu cầu thực thi và trả về kết quả cho chúng ta mà thôi)

**II/ - Hướng dẫn sử dụng Statement, PreparedStatement và CallableStatement**

Để minh họa acho việc sử dụng các đối tượng Statement, PreparedStatement và CallableStatement, chúng ta sẽ cùng thực tập thông qua cơ sở dữ liệu qlLichHen dựa trên DBMS là SQL Server 2005. Mô hình của Database qlLichHen có dạng như sơ đồ sau



Để minh họa cách khai thác thông tin trong cơ sở dữ liệu thông qua 3 loại đối tượng đã đề cập, tôi giả sử ta có 2 nhu cầu cụ thể như sau

**Yêu cầu 1:**  - Lấy danh sách người quen theo nhóm (**bb**-Nhóm bạn bè; **nt**-Nhóm nười thân, **ht**-Nhóm bạn học, **dt**-Đối tác kinh doanh, …) với những thông tin cần lấy bao gồm : Họ và tên, sinh nhật, số di động, email, địa chỉ nhà. Trong tình huống này, ta xây dựng 1 hàm để truy vấn cơ sở dữ liệu và kết quả trả về là 1 ResultSet chứa thông tin lấy được từ Database.

**Yêu cầu2:**  - Nhập thông tin cho table **ttPhanNhom** khi người dùng cần tạo ra 1 nhóm mới trong các quan hệ cá nhân của mình. Ta cũng sẽ xây dựng 1 hàm dùng cho việc nhập dữ liệu

Bây giờ, chúng ta sẽ cùng nhau thực hiện 2 yêu cầu trên lần lượt bằng cả 3 loại đối tượng, để qua đó, bạn có thể nhìn nhận rõ hơn về từng loại đối tượng và trong từng tình huống cụ thể, biết cách chọn đối tượng để sử dụng cho phù hợp.

Tình huống thứ nhất : **Truy vấn thông tin**

1 – Sử dụng **Statement**

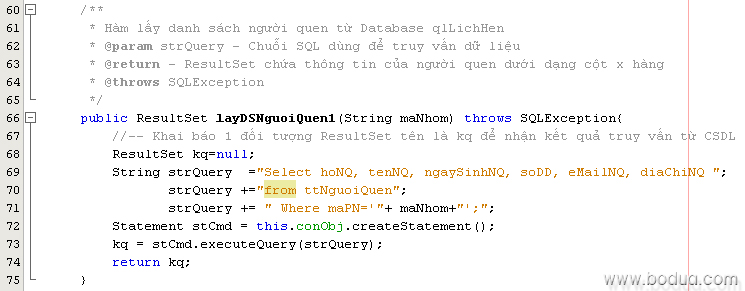
Để sử dụng 1 đối tượng của giao diện **Statement**, bạn phải nhớ các bước như sau

      Tạo 1 thể hiện của interface này thông qua phương thức ***CreateStatement***() dựa trên đối tượng **Connection** đã tạo để kết nối đến Database

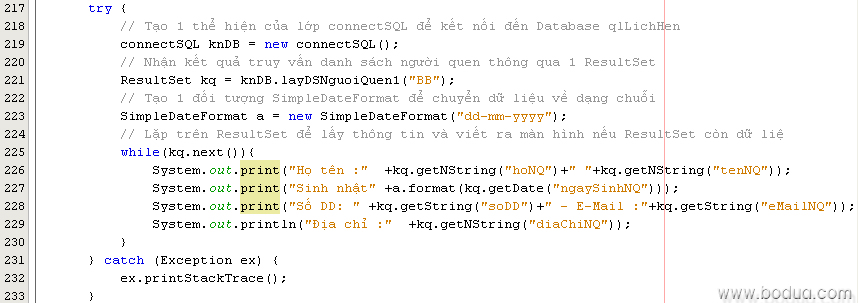
      Tạo 1 chuỗi truy vấn đến cơ sở dữ liệu dựa trên cấu trúc lệnh SQL

      Gọi phương thức ***executeQuery***() hoặc ***executeUpdate***() để thực thi câu SQL và nhận về kết quả truy vấn thông qua 1 đối tượng ResultSet (*Nếu bạn thi hành bằng phương thức****executeQuery***) hoặc 1 biến thuộc kiểu int (*nếu bạn đã yêu cầu thực thi bằng phương thức****executeUpdate***)

Như vậy, chúng ta xây dựng hàm  như mô tả sau:



Tại vị trí gọi hàm này trong chương trình, bạn phải truyền tham số cho hàm chính là mã nhóm mà bạn muốn truy xuất (**bb**, **ht**, **nt**, **dt**, ….). ví dụ



2 – Sử dụng **PreparedStatement**

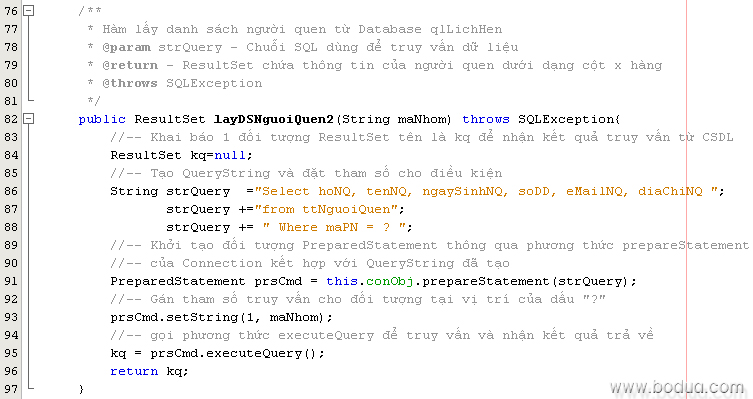
Để sử dụng 1 đối tượng của giao diện **PreparedStatement**, bạn phải nhớ các bước như sau

      Tạo 1 chuỗi truy vấn đến cơ sở dữ liệu dựa trên cấu trúc lệnh SQL đồng thời sử dụng ký tự “?” để xác định trí cho tham số muốn truyền vào.

      Tạo 1 thể hiện của interface này thông qua phương thức ***prepareStatement***() dựa trên đối tượng **Connection** đã tạo để kết nối đến Database, tham số truyền vào cho phương thức này chính là chuỗi truy vấn đã tạo ở bước 1

      Truyền tham số truy vấn cho đối tượng ***PreparedStatement*** đã tạo ra ở bước trên thông qua các phương thức ***setString***, ***setInt***, ***setFloat***, ***setBoolean***, …

      Gọi phương thức ***executeQuery***() hoặc ***executeUpdate***() để thực thi câu SQL và nhận về kết quả truy vấn thông qua 1 đối tượng ResultSet (*Nếu bạn thi hành bằng phương thức****executeQuery***) hoặc 1 biến thuộc kiểu int (*nếu bạn đã yêu cầu thực thi bằng phương thức****executeUpdate***) – Lưu ý: Khác với đối tượng của Statement. Khi gọi phương thức này của đối tượng PreparedStatement ta không truyền tham số (*Bạn hãy để ý kỹ đoạn Code ở dưới*)



3 – Sử dụng **CallableStatement**

Khác với 2 loại đối tượng trên, **CallableStatement** thường dùng để thực thi 1 stored procedure trong cơ sở dữ liệu. Chính vì thế để sử dụng đối tượng loại này trong chương trình thì bạn phải có 1 Stored Procedure trong database trước. Để tạo ra Stored Procedure, bạn có thể tạo bằng chương trình với code java và thông qua 1 đối tượng **Statement** để tạo. Tuy nhiên trong ví dụ này tôi muốn các bạn tạo Stored procedure theo cách truyền thống mà ta vẫn làm trong cơ sở dữ liệu. Tức là bạn hãy khởi động công cụ “**Microsoft SQL Server Management Studio**” để tạo 1 stored procedure có nội dung như sau,

**create procedure** nguoiQuen @nhomNQ **varchar**(5)   
**as**  
     **select** hoNQ, tenNQ, ngaySinhNQ, soDD, eMailNQ, diaChiNQ   
           **from** ttNguoiQuen  
           **where** maPN = @nhomNQ

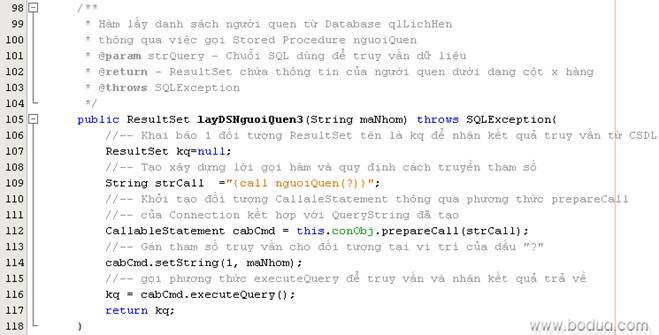
Lúc này, sau khi đã có Stored procedure “**nguoiQuen**” trong Database, ta sẽ tạo ra 1 hàm truy vấn danh sách người quen dựa trên đối tượng **CallableStatement** theo các bước như sau trong chương trình Java của chúng ta

      Tạo 1 chuỗi mô tả cho việc gọi Stored procedure trong cơ sở dữ liệu theo cú pháp sau   
”**{call  <tên\_sp>(?,?,?,...) }**” (*Trong đó tương ứng với mỗi dấu ? là 1 tham số của store procedure đã tạo trong Database*)

      Tạo 1 thể hiện của interface này thông qua phương thức ***prepareCall***() dựa trên đối tượng**Connection** đã tạo và truyền tham số thông qua lời gọi hàm cần sử dụng đã được tạo ở bước trên.

      Thiết lập tham số truy vấn truyền vào cho đối tượng ***CallableStatement*** đã tạo ra ở bước trên thông qua các phương thức ***setString***, ***setInt***, ***setFloat***, ***setBoolean***, …

      Gọi phương thức ***executeQuery***() hoặc ***executeUpdate***() để thực thi câu SQL và nhận về kết quả truy vấn thông qua 1 đối tượng ResultSet hoặc 1 biến thuộc kiểu int giống – Lưu ý: tương tự như cách thực thi câu SQL của đối tượng **PreparedStatement**, với đối tượng**CallableStatement** ta cũng không truyền tham số khi gọi phương thức này (*Bạn hãy để ý kỹ đoạn Code ở dưới*)



Việc gọi và sử dụng hàm ***layDSNguoiQuen3*** cũng giống như 2 lần trước (*Tôi không minh họa ở đây nữa vì điều này có thể làm cho bài viết trở nên quá dài, không cần thiết*)

Tình huống thứ hai : **Nhập dữ liệu cho Database**

Như vậy, trong tình huống trước chúng ta đã sử dụng object của các giao diện **Statement**,**PreparedStatement** và **CallableStatement** để thực hiện khai thác thông tin từ cơ sở dữ liệu; có lẽ các bạn cũng đã bắt đầu hình dung ra những lợi điểm của từng loại. Để nhìn nhận vấn đề rõ hơn 1 chút nữa, trong tình huống này ta sẽ sử dụng Object của các interface kể trên để thực hiện nhập dữ liệu cho table trong cơ sở dữ liệu, đây chính là nhu cầu sử dụng cơ sở dữ liệu để lưu trữ thông tin cho ứng dụng của chúng ta.

Tuy nhiên trước khi bắt đầu, bạn hãy tạo ra tương ứng với mỗi Table của cơ sở dữ liệu 1 class dùng để chứa thông tin cho từng đối tượng thuộc về Table đó. Ví dụ. Trong cơ sở dữ liệu của chúng ta có table **ttPhanNhom**, mục đích của table này là sẽ dùng để chứa các thông tin có liên quan đến những đối tượng phục vụ cho việc quản lý thông tin người quen theo nhóm, giúp việc thống kê sau này hoặc giả như tình huống, có 1 số cuộc hẹn trùng thời điểm, chương trình có thể dựa vào mức ưu tiên tùy thuộc theo nhóm người mà quyết định giúp người dùng nên thực hiện cuộc hẹn nào với ai trước, cuộc hẹn nào nên hủy bỏ, … Trong chương trình java, chúng ta sẽ đặt tên cho lớp đó là “**iPhanNhom**” để khi chương trình cần tác động đến thông tin phân nhóm của người quen thì có thể tạo ra object tương ứng dựa trên class đã tạo.

Cụ thể, tôi đã tạo ra trong chương trình 1 lớp tên là **iPhanNhom** với các biến thành viên như sau

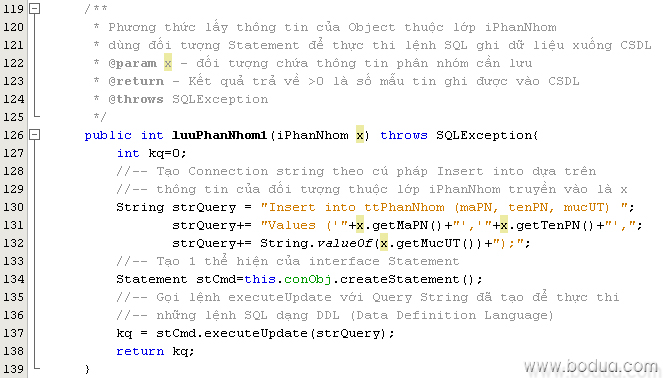
    **private** String \_maPN;              *//-- Mã của phân nhóm*    **private** String \_tenPN;              *//-- Diễn giải chi tiết cho nhóm người tương ứng*    **private** int      \_mucUT;           *//-- Mức ưu tiên là số nguyên từ 1 đến 10,   
                                                            //-- trọng số càng cao thì mức ưu tiên càng lớn*

Như các bạn thấy, ứng với các fields trong table **ttPhanNhom**, tôi tạo ra tương ứng bấy nhiêu biến thành phần của lớp **iPhanNhom** với kiểu dữ liệu tương đương trong Java. Sau đó tiến hành tạo ra các ***Setter*** và ***Getter*** cho mỗi thành viên, các hàm cơ bản như *Default constructor*, *Copy constructor*, *clone*…

Trong phạm vi bài viết này tôi không muốn mô tả phần chi tiết của từng lớp tương ứng với mỗi Table trong Database mà bạn phải tự thực hiện lấy để nội dung của bài viết không bị quá dài. Hơn nữa đây là phần cơ bản và tương đối dễ nên tôi chắc các bạn sẽ tự làm tốt phần công việc này.

1 – Sử dụng **Statement**

Bây giờ, chúng ta sẽ tạo ra 1 phương thức trong lớp connectSQL để phục vụ cho việc lưu dữ liệu vào table ttPhanNhom trong cơ sở dữ liệu với tên **luuPhanNhom1**

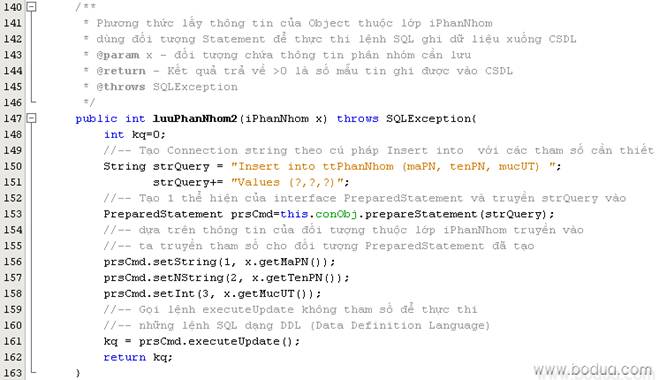


Giả sử bạn muốn lưu thông tin cho phân nhóm có mã là “kd”, tên nhóm “Đối tác kinh doanh” và bạn luôn ưu tiên hàng đầu cho các đối tượng người quen thuộc nhóm này vì thế mức ưu tiên sẽ là cao nhất =10, Ta gọi sử dụng hàm này kết hợp với đối tượng của lớp iPhanNhom như sau

*//--- Tạo đối tượng của lớp iPhanhNhom và nạp thông tin***iPhanNhom** tam= **new** **iPhanNhom**();  
tam.***setMaPN***(“kq”);  
tam.***setTenPN***(“Đối tác kinh doanh”);  
tam.***setMucUT***(10);  
*//--- Tạo đối tượng của lớp kết nối database và gọi phương thức ghi dữ liệu***connectSQL**  knDB = **new** **connectSQL**();  
knDB.***luuPhanNhom1***(tam);

2 – Sử dụng **PreparedStatement**

Với các dùng đối tượng của Interface PreparedStatement ta sẽ xây dựng hàm **luuPhanNhom2**như sau



Việc nhập thông tin thông qua hàm luuPhanNhom2 cũng giống như minh họa cho đối tượng của interface Statement ở phía trên, ta thực hiện như sau

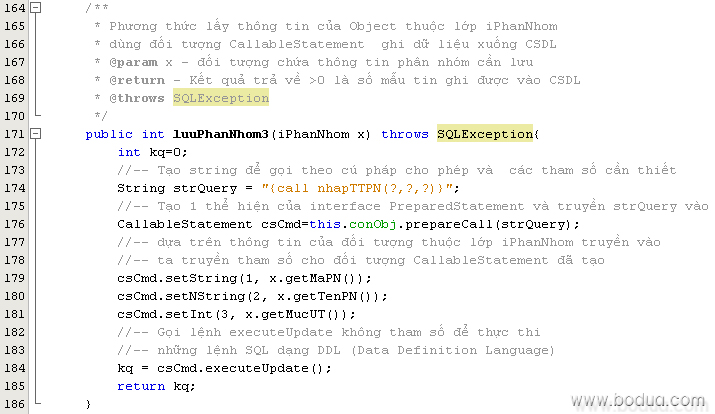
**try** {  
*//-- tạo 1 đối tượng của lớp iPhanNhomn để nạp thông tin*            **iPhanNhom** tam = **new** **iPhanNhom**();  
            tam.***setMaPN***("KD");  
            tam.***setTenPN***("Đối tác kinh doanh");  
            tam.***setMucUT***(10);  
*//-- Tạo 1 thể hiện của lớp kết nối và gọi hàm luuPhanNhom2 để ghi dữ liệu*            **connectSQL** kn = **new** **connectSQL**();  
            int kq = kn.***luuPhanNhom2***(tam);  
            **if** (kq > 0) {  
                System.out.print("Lưu thành công vào Database");  
            } **else** {  
                System.out.print("Không ghi vào CSDL được !");  
            }  
        } **catch** (Exception ex) {  
            ex.***printStackTrace***();  
        }

3 – Sử dụng **CallableStatement**

Như bạn đã biết, muốn sử dụng đối tượng **CallableStatement**, ta phải chuẩn bị 1 stored procedure trong Database. Bạn hãy vào SQL Server và tạo ra 1 Stored procedure có nội dung như sau:

**create procedure** nhapTTPN   
                        @ma **varchar**(5),   
                        @tenNhom **nvarchar**(60),   
                        @uuTien **tinyint**   
**as**  
      **insert into** ttPhanNhom   
                  (maPN, tenPN, mucUT)   
             **values**   
                  (@ma, @tenNhom, @uuTien)

Ta tạo ra 1 thủ tục có tên ***luuPhanNhom3*** sử dụng **CallableStatement** để ghi dữ liệu vào Database **qlLichHen** như mô tả dưới đây

****

Việc gọi thi hành hàm này cũng tương tự như những ví dụ trước, để dành cho bạn tự thực hiện lấy