**Giới thiệu**

Khi bạn làm việc với dữ liệu, bạn sẽ thường xuyên cần lọc kết quả dựa trên một vài điều kiện. Thông thường điều này được thực hiện bằng cách lấy dữ liệu nhập từ người dùng và tạo ra câu truy vấn SQL từ đó. Ví dụ, nhà kinh doanh cần xem tất cả hóa đơn trong một khoảng thời gian xác định.  Hoặc một truy vấn khác có thể là lọc các khách hàng theo thành phố.

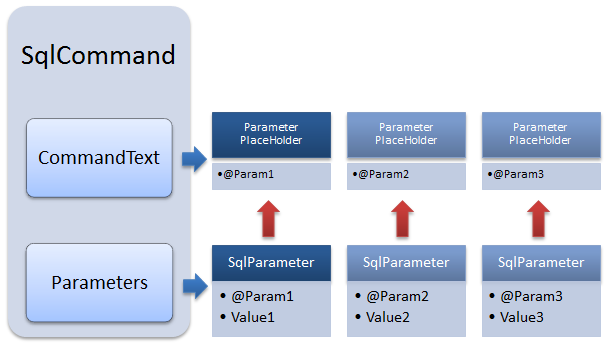
Như bạn biết, câu truy vấn SQL được gán vào đối tượng SqlCommand chỉ là một chuỗi. Vì thế, nếu bạn muốn tạo một truy vấn lọc, bạn có thể tạo một chuỗi động, nhưng bạn không cần phải làm thế. Đây là một ví dụ tồi cho việc tạo một câu truy vấn lọc.

*// don’t ever do this!*

*SqlCommand cmd = new SqlCommand(“select \* from Customers where city = ‘” + inputCity + “‘”;*

Đừng bao giờ tạo một câu truy vấn theo cách trên! Biến chứa dữ liệu nhập, *inputCity*, chỉ đơn giản là lấy dữ liệu từ TextBox trên Windows form hoặc Web Page. Bất kì thứ gì trong TextBox sẽ được đặt vào *inputCity*và được thêm vào câu SQL của bạn. Cách làm này có thể bị hacker thay thế chuỗi này bằng những thứ gây hại (SQL Injection). Trong trường hợp xấu nhất, bạn có thể bị lấy toàn quyền điều khiển của máy tính.

Thay vì tạo một chuỗi động, như bạn thấy ở ví dụ trên, hãy sử dụng parameter. Bất kì thứ gì được đặt vào một parameter sẽ được coi là một trường dữ liệu, không phải là một phần của câu lệnh SQL, điều này giúp ứng dụng của bạn an toàn hơn.

[](https://yinyangit.files.wordpress.com/2011/08/ado-net-sqlcommand-sqlparameter.png)

(**Mô hình kết hợp giữa SqlParameter và SqlCommand**)

Dùng câu truy vấn với parameter bao gồm ba bước sau:

1. Tạo một SqlCommand từ một câu lệnh có parameter.
2. Khai báo một đối tượng SqlParameter, gán giá trị thích hợp cho nó.
3. Gán đối tượng SqlParameter vào property Parameters của đối tượng SqlCommand.

Các phần sau hướng dẫn từng bước quá trình trên.

**Tạo một đối tượng SqlCommand sử dụng Parameter**

Bước đầu tiên là tạo một câu lệnh chứa các parameter placeholder (tên của parameter). Tên này sẽ được thay thế bởi giá trị thực sự của parameter khi SqlCommand thực thi. Cú pháp đúng của một parameter là dùng kí hiệu tiền tố ‘@’ trong tên của parameter như sau:

*// 1. declare command object with parameter  
SqlCommand cmd = new SqlCommand(“select \* from Customers where city = @City”, conn);*

Tro ng constructor của SqlCommand trên, tham số đầu tiên chứa một khai báo parameter, *@City*. Ví dụ này dùng một parameter, nhưng bạn có thể có nhiều parameter tùy theo số lượng bạn cần để tạo cây truy vấn. Mỗi parameter sẽ so khớp với một đối tượng SqlParameter được gán vào đối tượng SqlCommand.

**Khai báo một đối tượng SqlParameter**

Mỗi parameter trong câu lệnh SQL phải được định nghĩa. Đây là mục đích của kiểu SqlParameter. Mã nguồn của bạn phải định nghĩa một đối tượng SqlParameter cho mỗi parameter trong câu lệnh SQL của đối tượng SqlCommand. Đoạn mã sau định nghĩa một SqlParameter cho parameter *@City* trong phần trước:

*// 2. define parameters used in command object  
SqlParameter param  = new SqlParameter();  
param.ParameterName = “@City”;  
param.Value         = inputCity;*

Lưu ý rằng property ParameterName của đối tượng SqlParameter phải được viết đúng với parameter được dùng trong câu lệnh SQL của SqlCommand. Bạn cũng phải xác định giá trị cho các parameter. Khi đối tượng SqlCommand được thực thi, parameter sẽ được thay thế bằng giá trị của nó.

**Kết hợp đối tượng SqlParameter với đối tượng SqlCommand**

Với mỗi parameter được định nghĩa trong câu lệnh SQL của đối tượng SqlCommand phải được định nghĩa một SqlParameter. Bạn cũng phải để đối tượng SqlCommand biết về SqlParameter bằng cách gán đối tượng SqlParameter cho property Parameters của đối tượng SqlCommand. Dòng mã sau cho thấy cách làm điều này:

*// 3. add new parameter to command object  
cmd.Parameters.Add(param);*

Đối tượng SqlParameter là tham số trong phương thức Add() của property Parameters của đối tượng SqlCommand trên. Bạn phải thêm một SqlParameter duy nhất cho mỗi parameter đã định nghĩa trong câu lệnh SQL của đối tượng SqlCommand.

**Kết hợp tất cả lại**

Bạn đã biết cách dùng các đối tượng SqlCommand và SqlDataReader. Đoạn mã sau minh họa một chương trình làm việc sử dụng các đối tượng SqlParameter. Hiện tại, mọi thứ đã quen thuộc với bạn, ngoại trừ các phần được giới thiệu trong bài học này:

**Listing 1: Adding Parameters to Queries**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56 | using System;  using System.Data;  using System.Data.SqlClient;    class ParamDemo  {      static void Main()      {          // conn and reader declared outside try          // block for visibility in finally block          SqlConnection conn   = null;          SqlDataReader reader = null;            string inputCity = "London";            try          {              // instantiate and open connection              conn =  new                  SqlConnection("Server=(local);DataBase=Northwind;Integrated Security=SSPI");              conn.Open();                // don't ever do this!  //          SqlCommand cmd = new SqlCommand(  //              "select \* from Customers where city = '" + inputCity + "'";                // 1. declare command object with parameter              SqlCommand cmd = new SqlCommand(                  "select \* from Customers where city = @City", conn);                // 2. define parameters used in command object              SqlParameter param  = new SqlParameter();              param.ParameterName = "@City";              param.Value         = inputCity;                // 3. add new parameter to command object              cmd.Parameters.Add(param);                // get data stream              reader = cmd.ExecuteReader();                // write each record              while(reader.Read())              {                  Console.WriteLine("{0}, {1}",                      reader["CompanyName"],                      reader["ContactName"]);              }          }          finally          {              // close reader              if (reader != null)              {                  reader.Close();              }                // close connection              if (conn != null)              {                  conn.Close();              }          }      }  } |

Đoạn mã trong Listing 1 đơn giản là lấy các dòng dữ liệu của mỗi khách hàng sống tại London. Điều này giúp bảo mật hơn khi dùng parameter.