Parametrizarea interogărilor - Clasa SqlParameter

Clasa SqlParameter face parte din spațiul de nume System.Data.SqlClient și oferă o modalitate simplă și rapidă de parametrizare a interogărilor.

La definirea interogărilor parametrizate vom prefixa numele parametrilor cu caracterul „@”.

Proprietățile parametrilor se setează fie prin metodele constructor ale clasei SqlParameter fie prin asignarea lor în afara constructorului.

Prin setarea proprietății Direction stabilim tipul parametrului:

de intrare(implicit) p.Direction=ParameterDirection.Input;

de iesire p.Direction=ParameterDirection.Output;

de intrare/iesire p.Direction= ParameterDirection.InputOutput;

Proprietatea [*SourceColumn*](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlparameter.sourcecolumn(v=vs.110).aspx) oferă o modalitate simplă de mapare a parametrului cu o coloana a DataSetului pentru care sunt implementați accesorii get și set.

Asocierea unui parametru de tip SqlParameter la un obiect de tip SqlCommand se face cu ajutorul metodei Add astfel:

cmd.Parameters.Add (param);

unde cmd este de tip SqlCommand si param de tip SqlParameter

Exemplu :

string strInsert = "insert into tClienti (CodClient,NumeClient,Localitate)" +

" values(@CodClient,@NumeClient,@Localitate)";

SqlParameter p1 = new SqlParameter();

p1.ParameterName="@CodClient";

p1.Value=tbCodCl.Text;

SqlParameter p2 = new SqlParameter("@NumeClient",tbNumeCl.Text);

SqlParameter p3=new SqlParameter("@Localitate",SqlDbType.Char,30);

p3.Value = tbLocalitate.Text;

SqlCommand cmd = new SqlCommand(strInsert, Global.con);

cmd.Parameters.Add(p1);

cmd.Parameters.Add(p2);

cmd.Parameters.Add(p3);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

Următorul exemplu creează un SqlDataAdapter și stabilește proprietățile SelectCommand, InsertCommand, UpdateCommand și DeleteCommand. Se presupune avem creat deja un obiect SqlConnection.

public static SqlDataAdapter CreareCommands\_Adapter\_daFact(SqlConnection con)

{

SqlDataAdapter da = new SqlDataAdapter();

SqlCommand cmd;

string strSql = "select A.\*,NumeClient " +

" from tFacturi as A inner join tClienti as B " +

" on A.codClient=B.codClient";

cmd = new SqlCommand(strSql, con);

da.SelectCommand = cmd;

strSql = "update tFacturi " +

" set nrFact=@nrFact , dataFact=@dataFact,

codClient=@CodClient,codAngajat=@CodAngajat " +

"where nrFact=@old\_nrFact";

cmd = new SqlCommand(strSql,con);

SqlParameter p = new SqlParameter("@nrFact", SqlDbType.Char, 10, "nrFact");

cmd.Parameters.Add(p);

p = new SqlParameter("@dataFact", SqlDbType.SmallDateTime, 4, "dataFact");

cmd.Parameters.Add(p);

cmd.Parameters.Add("@codClient", SqlDbType.VarChar, 10, "codClient");

cmd.Parameters.Add("@codAngajat", SqlDbType.VarChar, 10, "codAngajat");

cmd.Parameters.Add("@old\_nrFact", SqlDbType.Char, 10, "nrFact").

SourceVersion = DataRowVersion.Original;

da.UpdateCommand = cmd;

strSql = "insert into tFacturi(nrFact,dataFact,CodClient,CodAngajat) values(@nrFact,@dataFact,@CodClient,@CodAngajat)";

cmd = new SqlCommand(strSql, con);

cmd.Parameters.Add("@nrFact", SqlDbType.Char, 10, "nrFact");

cmd.Parameters.Add("@dataFact", SqlDbType.SmallDateTime, 4, "dataFact");

cmd.Parameters.Add("@codClient", SqlDbType.VarChar, 10, "codClient");

cmd.Parameters.Add("@codAngajat", SqlDbType.VarChar, 10, "codAngajat");

da.InsertCommand = cmd;

strSql = "delete tFacturi where nrFact=@nrFact";

cmd = new SqlCommand(strSql, con);

cmd.Parameters.Add("@nrFact", SqlDbType.Char, 10, "nrFact");

da.DeleteCommand = cmd;

return da;

}

**DataView**

***DataView***-urile oferă o vedere dinamica, personalizată a continutului tabelelor dintr-un ***DataSet*** şi reflectă modificările aduse acestor date. DataView-urile ne permit să sortăm datele, să filtrăm şi să căutăm date să prelucrăm rândurile noi, șterse sau modificate sau rândurile nemodificate. De asemenea, pentru rândurile modificate putem avea acces la valorile lor originale.

Dacă nu se asociază în mod explicit o vedere de tip DataView în program, putem folosi o vedere DataView implicită, disponibilă pentru fiecare tabel al setului de date, prin accesarea proprietăţii ***DefaultView*** a tabelului.

Prin intermediul proprietăţii ***RowStateFilter*** a DataView-ului setată cu ajutorul proprietăţilor clasei ***DataViewRowState*,** putem avea acces la starea rândurilor (noi, modificate, șterse) şi astfel le putem filtra.

Proprietățiile clasei DataViewRowState sunt:

**Added**. Ne furnizează rândurile noi.  
**CurrentRows**. Ne furnizează rândurile curente: rândurile neschimbate, noi şi modificate.

**Deleted**. Ne furnizează rândurile eliminate.  
**ModifiedCurrent**. Ne furnizează rândurile modificate

**ModifiedOriginal**. Ne furnizează versiunea originală a rândurilor modificate.  
   
**OriginalRows**. Ne furnizează rândurile originale, inclusiv rândurile neschimbate şi eliminate.  
**Unchanged**. Ne furnizează rândurile neschimbate.

DataTable dt=Global.ds.Tables["Produse"];

dgvPr.DataSource=dt;

DataView dv=new DataView(dt);

dgvPr2.DataSource = dv;

dv = new DataView(dt, "UM='Kg' or UM='L'", "UM asc, DenProd desc", DataViewRowState.CurrentRows);

dgvPr2.DataSource = dv;

dv = new DataView(dt, "UM='Kg' or UM='L'", null, DataViewRowState.CurrentRows);

dgvPr2.DataSource = dv;

dv.Sort = "UM asc, DenProd desc";

dgvPr2.DataSource = dv;

dv.Sort = "CodProd";

int index = dv.Find("P3");

if (index < 0) MessageBox.Show("Cod negasit");

else MessageBox.Show(dv[index]["CodProd"] + " " + dv[index]["DenProd"]);

dv.Sort = "UM asc, DenProd desc";

index = dv.Find(new string[2] { "Kg", "Pere" });

if (index < 0) MessageBox.Show("Cod negasit");

else MessageBox.Show("Cod,DEN " + dv[index]["CodProd"] + " " + dv[index]["DenProd"]);

dv.Sort = "UM";

DataRowView[] rv=dv.FindRows("KG");

MessageBox.Show( ""+rv.Length);

foreach(DataRowView r in rv)

MessageBox.Show(r["CodProd"]+" "+r["DenProd"]);

**APLICATIE**





