

Re: Building SqlParameter: Design Time or Runtime ?

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.adonet/2004-03/2610.html>

From: William \ (Bill) Vaughn (*billvaRemoveThis_at_nwlink.com*)

Date: 03/29/04

Date: Mon, 29 Mar 2004 12:38:52 -0800

I've started dragging the SP from the server explorer to the form (or component) window. This generates functional Command object code.

hth

--

William (Bill) Vaughn
Author, Mentor, Consultant
Microsoft MVP
www.betav.com

Please reply only to the newsgroup so that others can benefit.
This posting is provided "AS IS" with no warranties, and confers no rights.

"Greg" <greg@cds-am.net> wrote in message
news:uJLjDGbFEHA.3568@tk2msftngp13.phx.gbl...
> Bill, what about output params? Any clever way to get this set at runtime
> vs. design time? I have not come up with a way.
>
>
> "William Ryan eMVP" <dotnetguru@comcast.nospam.net> wrote in message
> news:eizFUuaFEHA.628@TK2MSFTNGP10.phx.gbl...
> > Greg:
> >
> > I tend to agree that things should be as explicit as possible. However,
> > if
> > you pass in a parameter declared with the minimal constructor
> > cmd.Parameters.Add("@ParamName", ValueForParam)
> >
> > you should be good to go. I've been told by many folks that I trust
> that
> > ADO.NET is very good at determining the correct type and that the
> > performance
> > hit is not worth mentioning b/c it figures it out very quickly. So as
> > long
> > as the value you send it is legitimate, you're in good shape.
> >
> > The other option though is to query the table and use something like
> > DataReader.GetSchemaTable. Then you'll know the type (DataType) of the
> > .NET
> > framework type (and you can easily convert this to the SQL Server
> > equivalent

microsoft.public.dotnet.framework.adonet: Re: Building SqlParameter: Design Time or Runtime ?

```
> > SqlDbType...) as well as the length. You will know the column mapping
if
> > you need it but basically at this point you'll know the parameter name
> > (because you created it), the SqlDbType and the length:
> > cmd.Parameters.Add("@ParamName", SqlDbType.VarChar, 50).Value =
> > ValueForParam //where SqlDbType.Type will change depending on the schema
> > information and the 50 will change as well).
> >
> > With that info, you can query the table at the initiation of the program
> or
> > right before you need it (which done over and over is wasteful) and
build
> > the parameters with all the precise info you need. You'll have a small
> bit
> > of overhead with the getschematable, but that's negligible ,and you'll
> know
> > have they type and size for each param as soon as you need it. The only
> > possible problem is if the user changes the type in the middle of a
> session
> > when the program is running, but if you've hard coded the param
> information
> > (which you have) then that problem would exist anyway. If that's going
to
> > happen, you can requery the schematable before you fire each command.
> > However, I really doubt the user is going to change the type/sizes that
> > frequently to warrant running the same command over and over.
> >
> > I use this approach to dynamically determine the types for all cases
where
> I
> > can't use Stored procs but I want to still use parameterized queries.
in
> > each case, I query the tables at the beginning and use the datatable
that
> I
> > got the schema table into in a module or static property of a class so I
> can
> > access it over and over. Since schema changes happen here but are
> > infrequent, getting the info at the begining of a session is good enough
> > (and there's no alternative other than constantly re-querying, even with
> > stored procs). This allows me to safely use 'dynamic sql' but still
take
> > advantage of params for performance and security.
> >
> > HTH,
> >
> > Bill
> >
> > HTH,
> >
> > Bill
> > "Greg" <greg@cds-am.net> wrote in message
> > news:u8WZmKaFEHA.696@TK2MSFTNGP12.phx.gbl...
> > > I have never been a fan of building Input SqlParameter dynamically. I
> > know
> > > ADO.NET does a pretty good job figuring out datatype and size.
However,
> I
> > > like having the control and knowing I can affect the performance by
> > building
> > > them myself.
> > >
```

microsoft.public.dotnet.framework.adonet: Re: Building SqlParameter: Design Time or Runtime ?

```
> > > When I initially wrote our DAL, I built the SqlParameter dynamically
> > > because this is how my clients pretty much demand I do it in the past.
> It
> > > worked well except for one or two fields. So, I redesigned the DAL to
> > build
> > > the SqlParameter with specific code for each parameter. I added a
> simple
> > type that built the commands, as needed, for me:
> > >
> > > Public Shared Function BuildUpdateCommand(ByVal tableName As String)
As
> > >
> > > SqlCommand
> > >
> > > If tableName Is Nothing Then Throw New
> > >
> > > ArgumentNullException("tableName")
> > >
> > > Dim updateCommand As New SqlCommand
> > >
> > > Select Case tableName.ToLower
> > >
> > > Case "names"
> > >
> > > updateCommand.CommandText = "[usp_update_names]"
> > >
> > > updateCommand.CommandType =
> > >
> > > System.Data.CommandType.StoredProcedure
> > >
> > > updateCommand.Parameters.Add(New
> > >
> > > System.Data.SqlClient.SqlParameter("@id",
System.Data.SqlDbType.VarChar,
> > 6,
> > >
> > > "names_id"))
> > >
> > > updateCommand.Parameters.Add(New
> > >
> > > System.Data.SqlClient.SqlParameter("@me",
System.Data.SqlDbType.VarChar,
> > 1,
> > >
> > > "names_me"))
> > >
> > > updateCommand.Parameters.Add(New
> > >
> > > System.Data.SqlClient.SqlParameter("@ty",
System.Data.SqlDbType.VarChar,
> > 2,
> > >
> > > "names_ty"))
> > >
> > > updateCommand.Parameters.Add(New
> > >
> > > System.Data.SqlClient.SqlParameter("@lname",
> > System.Data.SqlDbType.VarChar,
> > >
> > > 20, "names_lname"))
> > >
> > > updateCommand.Parameters.Add(New
```

microsoft.public.dotnet.framework.adonet: Re: Building SqlParameter: Design Time or Runtime ?

```
> > >
> > > System.Data.SqlClient.SqlParameter("@fname",
> > System.Data.SqlDbType.VarChar,
> > >
> > > 15, "names_fname"))
> > >
> > > Case "addr"
> > >
> > > etc....
> > >
> > > Return updateCommand
> > >
> > > My prior, dynamic code, looked something like this:
> > >
> > > ' handler for DataAdapter RowUpdating event
> > >
> > > Protected Shared Sub OnRowUpdating(ByVal sender As Object, ByVal e As
> > >
> > > SqlRowUpdatingEventArgs)
> > >
> > > Dim col As DataColumn
> > >
> > > Dim drw As DataRow = e.Row
> > >
> > > Dim myParam As SqlParameter
> > >
> > > For Each col In drw.Table.Columns
> > >
> > > Try
> > >
> > > ' check the original row against the current row to see if they are
> > > different
> > >
> > > If Not (drw(col, DataRowVersion.Current).Equals(drw(col,
> > >
> > > DataRowVersion.Original))) Then
> > >
> > > ' build a parameter for the changed column
> > >
> > > myParam = New SqlParameter("@" & col.ColumnName.Remove(0,
> > >
> > > col.ColumnName.IndexOf("_") + 1), drw.Item(col).ToString)
> > >
> > > e.Command.Parameters.Add(myParam)
> > >
> > > Again, the 'dynamic' code worked nice except for 2-3 fields. I was not
> > > happy
> > > giving control over to ADO.NET, but it seems to do the job nicely.
> > >
> > > So, now my client has informed me there are cases where our client
> > > change
> > > the type and size in their sql server tables. Each client maintains
> > > there
> > > own sql server database.
> > >
> > > Now, I need to go back to the 'dynamic' approach. I am resisting
though
> as
> I
> > > once again will lose control and will have to trust ADO.NET to build
the
> > > correct type and size.
```

microsoft.public.dotnet.framework.adonet: Re: Building SqlParameter: Design Time or Runtime ?

> > >
> > > Is there happy middle ground here? Is there a way to have a little
more
> > > control over the definition of a SqlParameter but still build it
> > dynamically
> > > or at design time? I will not use the CommandBuilder nor will I query
> the
> > > sprocs to get this info. Too much overhead.
> > >
> > >
> > >
> > >
> > >
> > > Greg Robinson
> > >
> > > Custom Data Systems, Inc.
> > >
> > > www.cds-am.net
> > >
> > >
> >
> >
>
>