Assign null to a SqlParameter

Asked 8 years, 7 months ago Active 5 months ago Viewed 247k times

162

32

The following code gives an error - "No implicit conversion from DBnull to int."

```
SqlParameter[] parameters = new SqlParameter[1];
SqlParameter planIndexParameter = new SqlParameter("@AgeIndex", SqlDbType.Int);
planIndexParameter.Value = (AgeItem.AgeIndex== null) ? DBNull.Value : AgeItem.AgeIndex;
parameters[0] = planIndexParameter;
```

c# dbnull sqlparameter

edited May 16 '16 at 9:41

bluish
14.8k 19 94 152

asked Dec 29 '10 at 16:42

Relativity
2,877 16 70 113

4 You need to cast AgeItem.AgeIndex to object I think... stackoverflow.com/questions/202271/… (btw, why the `==` at the end of the 3rd line?) – Greg Dec 29 '10 at 16:48

18 Answers

300

The problem is that the `?:` operator cannot determine the return type because you are either returning an `int` value or a DBNull type value, which are not compatible.

You can of course cast the instance of AgeIndex to be type `object` which would satisfy the `?:` requirement.

You can use the `??` null-coalescing operator as follows

```
SqlParameter[] parameters = new SqlParameter[1];
SqlParameter planIndexParameter = new SqlParameter("@AgeIndex", SqlDbType.Int);
planIndexParameter.Value = (object)AgeItem.AgeIndex ?? DBNull.Value;
parameters[0] = planIndexParameter;
```

Here is a quote from the MSDN documentation for the `?:` operator that explains the problem

> Either the type of first_expression and second_expression must be the same, or an implicit conversion must exist from one type to the other.

edited Nov 23 '15 at 18:19

Lee Taylor
5,140 7 24 43

answered Dec 29 '10 at 16:53

Chris Taylor
43.9k 8 61 80

Why is there no exception thrown when trying to cast null to object? I would think it should be `AgeItem.AgeIndex as object` – Niels Brinch Jun 5 '12 at 8:34

@Niels Brinch, there would not be an exception because null is an object and as long as you do not try dereference it, it is perfectly legal. However, in this example it is not null being cast to object, it is DBNull.Value which is actually a value type. The ?? operator says 'if AgetItem.AgeIndex is null then return DBNull.Value otherwise returen AgeItem.AgeIndex' then the response is cast to object. See null coalescing operator for more details. msdn.microsoft.com/en-us/library/ms173224.aspx – Chris Taylor Jun 9 '12 at 11:23

3 Technically, your solution using the null-coalescing operator `??` is the same solution as if you were to use the regular ternary `?:` - you still need to cast `AgeItem.AgeIndex` to an object: `planIndexParameter.Value = AgeItem.AgeIndex.HasValue ? (object)AgeItem.AgeIndex : DBNull.Value;` . – newfurniturey Aug 15 '13 at 7:55

If you were to use the regular ternary `?:` to do a type-specific comparison, then casting the entire expression won't work. You have to cast the non-dbnull parameter like so: `someID == 0 ? DBNull.Value : (object)someID` – ingredient_15939 Sep 16 '15 at 0:59

That is true but if you need using null-able value as an entrance parameter of function that result consume SqlParameter and if it is null you've got error this way is not work and you should use just simple If-Else way. for example: sample.Text.Trim() != "" ? func(sample.Text) : DBNull.Value; will not work as ?: and ?? – QMaster Nov 15 '15 at 9:53

88

The accepted answer suggests making use of a cast. However, most of the SQL types have a special Null field which can be used to avoid this cast.

For example, `SqlInt32.Null` "Represents a DBNull that can be assigned to this instance of the SqlInt32 class."

```
int? example = null;
object exampleCast = (object) example ?? DBNull.Value;
object exampleNoCast = example ?? SqlInt32.Null;
```

edited May 23 '17 at 12:18
Community ♦

answered Dec 27 '13 at 19:08
Brian

2 The suggestion looked promising so I tried "System.Data.SqlTypes.SqlString.Null" but it doesn't work. It puts actual string of "Null" ('N', 'u', 'l', 'l') into the field instead leaving it blank with true (null). However, the old 2010 "accepted answer" that uses cast with (object) ?? DBNull.Value works correctly. (The ADO.NET provider I used was SQLite but I'm not sure if that makes a difference.) I suggest that others carefully test Brian's tip to make sure null behavior is working as expected. – JasDev May 18 '15 at 12:04

5 @JasDev: I vaguely recall describing this trick in a comment to a high rep user (I think Marc Gravell) and being told it only works on Microsoft SQL Server. – Brian May 18 '15 at 13:18

@JasDev the provider will be the difference this works in SQL Server as Brain point's out. – Lankymart Jun 9 '16 at 12:21

This answer only replaces an explicit cast to object with an implicit.one. In the sample code, `exampleNoCast` is declared object, so the cast to object still occurs. If, like in the OP's code, the value is assigned directly to SqlParameter.Value which is also of type object, then you still get the cast. – Scott Feb 23 '18 at 23:02

24

You need pass DBNull.Value as a null parameter within SQLCommand, unless a default value is specified within stored procedure (if you are using stored procedure). The best approach is to assign DBNull.Value for any missing parameter before query execution, and following foreach will do the job.

```
foreach (SqlParameter parameter in sqlCmd.Parameters)
{
    if (parameter.Value == null)
    {
        parameter.Value = DBNull.Value;
    }
}
```

Otherwise change this line:

```
planIndexParameter.Value = (AgeItem.AgeIndex== null) ? DBNull.Value : AgeItem.AgeIndex;
```

As follows:

```
if (AgeItem.AgeIndex== null)
    planIndexParameter.Value = DBNull.Value;
else
    planIndexParameter.Value = AgeItem.AgeIndex;
```

Because you can't use different type of values in conditional statement, as DBNull and int are different from each other. Hope this will help.

edited Dec 29 '10 at 16:58

answered Dec 29 '10 at 16:52

ShahidAzim
1,150 1 7 14

17

With one line of code, try this:

```
var piParameter = new SqlParameter("@AgeIndex", AgeItem.AgeIndex ??
(object)DBNull.Value);
```

edited Jul 28 '17 at 19:11

answered Apr 14 '17 at 17:18

Adrian
3,682 3 21 25

5

Try this:

```
SqlParameter[] parameters = new SqlParameter[1];
SqlParameter planIndexParameter = new SqlParameter("@AgeIndex", SqlDbType.Int);

planIndexParameter.IsNullable = true; // Add this line

planIndexParameter.Value = (AgeItem.AgeIndex== null) ? DBNull.Value : AgeItem.AgeIndex==
;
parameters[0] = planIndexParameter;
```

answered Dec 29 '10 at 16:53

decyclone
27.1k 5 55 73

If you use the conditional(ternary) operator the compiler needs an implicit conversion between both types, otherwise you get an exception.

3

So you could fix it by casting one of both to `System.Object` :

```
planIndexParameter.Value = (AgeItem.AgeIndex== null) ? DBNull.Value : (object)
AgeItem.AgeIndex;
```

But since the result is not really pretty and you always have to remember this casting, you could use such an extension method instead:

```
public static object GetDBNullOrValue<T>(this T val)
{
    bool isDbNull = true;
    Type t = typeof(T);

    if (Nullable.GetUnderlyingType(t) != null)
        isDbNull = EqualityComparer<T>.Default.Equals(default(T), val);
    else if (t.IsValueType)
        isDbNull = false;
    else
        isDbNull = val == null;

    return isDbNull ? DBNull.Value : (object) val;
}
```

Then you can use this concise code:

```
planIndexParameter.Value = AgeItem.AgeIndex.GetDBNullOrValue();
```

answered Sep 12 '16 at 8:58

Tim Schmelter

373k 49 497 753

1

In my opinion the better way is to do this with the Parameters property of the SqlCommand class:

```
public static void AddCommandParameter(SqlCommand myCommand)
{
    myCommand.Parameters.AddWithValue(
        "@AgeIndex",
        (AgeItem.AgeIndex== null) ? DBNull.Value : AgeItem.AgeIndex);
}
```

answered Dec 29 '10 at 16:45

user432219

But if the value is `DBNull.Value` , ADO.NET might have somewhat of a hard time guessing what SqlDbType that might be..... this is convenient - but a bit dangerous.... – marc_s Dec 29 '10 at 16:49

1

Consider using the Nullable(T) structure available. It'll let you only set values if you have them, and your SQL Command objects will recognize the nullable value and process accordingly with no hassle on your end.

answered May 2 '14 at 6:13

Kanwar Singh

571 9 18

0

Try this:

```
if (AgeItem.AgeIndex != null)
{
   SqlParameter[] parameters = new SqlParameter[1];
   SqlParameter planIndexParameter = new SqlParameter("@AgeIndex", SqlDbType.Int);
   planIndexParameter.Value = AgeItem.AgeIndex;
   parameters[0] = planIndexParameter;
}
```

In other words, if the parameter is null just don't send it to your stored proc (assuming, of course, that the stored proc accepts null parameters which is implicit in your question).

edited Jan 5 '11 at 6:27

answered Dec 29 '10 at 16:44

Flipster

2,878 3 22 34

But now, you're just omitting a parameter - I highly doubt the stored procedure will be happy about this.... most likely, the call will fail stating "no value for parameter @AgeIndex supplied which was expected"..... – marc_s Dec 29 '10 at 16:48

Wow. Harsh. Just write the stored proc to default to a value if the parameter is not passed (@AgeIndex int = 0). Happens all the time. The client can either accept the default, or override it by passing the parameter. Why the downvote? – Flipster Jan 5 '11 at 6:24

try something like this:

0

```
if (_id_categoria_padre > 0)
{
    objComando.Parameters.Add("id_categoria_padre", SqlDbType.Int).Value =
_id_categoria_padre;
}
else
{
    objComando.Parameters.Add("id_categoria_padre", DBNull.Value).Value = DBNull.Value;
}
```

edited Jul 11 '13 at 22:16

Marco
47.6k 11 111 134

answered Jul 11 '13 at 21:44

user2574441
1

0

```
int? nullableValue = null;
object nullableValueDB
{
   get{
       if(nullableValue==null)
          return DBNull.Value;
       else
          return (int)nullableValue;
   }
}
```

I'm solving like that.

answered Mar 1 '14 at 11:55

A Sinan Direk
11 4

```
if (_id_categoria_padre > 0)
{
```

0

```
    objComando.Parameters.Add("id_categoria_padre", SqlDbType.Int).Value =
_id_categoria_padre;
}
else
{
    objComando.Parameters.Add("id_categoria_padre", DBNull.Value).Value = DBNull.Value;
}
```

edited May 2 '14 at 6:13

Rugmangathan
2,535 5 26 40

answered May 2 '14 at 5:51

Anil Kumar
1 1

0

```
if (AgeItem.AgeIndex== null)
    cmd.Parameters.Add(new SqlParameter("ParaMeterName", SqlDbType.DateTime).Value =
DBNull);
else
    cmd.Parameters.Add(new SqlParameter("ParaMeterName", SqlDbType.DateTime).Value =
AgeItem.AgeIndex);
```

edited May 2 '14 at 6:13

answered May 2 '14 at 5:59

Anil Kumar
1 1

0

This is what I simply do...

```
        var PhoneParam = new SqlParameter("@Phone", DBNull.Value);
        if (user.User_Info_Phone != null)
        {
            PhoneParam.SqlValue = user.User_Info_Phone;
        }

        return this.Database.SqlQuery<CustLogonDM>("UpdateUserInfo @UserName, @NameLast,
@NameMiddle, @NameFirst, @Address, @City, @State, @PostalCode, @Phone",
            UserNameParam, NameLastParam, NameMiddleParam, NameFirstParam, AddressParam,
CityParam, StateParam, PostalParam, PhoneParam).Single();
```

answered Jun 13 '15 at 18:59

Tom Mack
1

0

```
dynamic psd = DBNull.Value;

if (schedule.pushScheduleDate > DateTime.MinValue)
{
    psd = schedule.pushScheduleDate;
}


sql.DBController.RunGeneralStoredProcedureNonQuery("SchedulePush",
         new string[] { "@PushScheduleDate"},
         new object[] { psd }, 10, "PushCenter");
```

answered Jul 15 '16 at 22:24

papapa
3 1

0

A simple extension method for this would be:

```
    public static void AddParameter(this SqlCommand sqlCommand, string parameterName,
        SqlDbType sqlDbType, object item)
    {
        sqlCommand.Parameters.Add(parameterName, sqlDbType).Value = item ??
DBNull.Value;
    }
```

answered Mar 27 '17 at 18:19

Mark
479 5 15

0

I use a simple method with a null check.

```
public SqlParameter GetNullableParameter(string parameterName, object value)
{
    if (value != null)
    {
        return new SqlParameter(parameterName, value);
    }
    else
    {
        return new SqlParameter(parameterName, DBNull.Value);
    }
}
```

edited Nov 19 '18 at 6:29

answered Apr 9 '17 at 4:38

Zhi An
11 2

1 Is that conditional logic backwards? Should DBNull.Value be in the first one? – Mark Schultheiss May 4 '18 at 12:38

Surely is. Fixed. Thanks. – Zhi An Nov 19 '18 at 6:44

0

My code, working in real project Look the ternary operator beafore make the sqlparameter this is the best way for me, withou problems:

```
public bool Key_AddExisting
(
      string clave
    , int? idHito_FileServer
    , int? idTipoDocumental_Almacen
    , string tipoExp_CHJ
    , int idTipoExp_Verti2
    , int idMov_Verti2
)
{
    List<SqlParameter> pars = new List<SqlParameter>()
    {
          new SqlParameter { ParameterName = "@Clave", Value = clave }
    LOOK -> , idHito_FileServer == null ? new SqlParameter { ParameterName =
"@IdHito_FileServer", Value = DBNull.Value } : new SqlParameter { ParameterName =
"@IdHito_FileServer", Value = idHito_FileServer }
    LOOK -> , idTipoDocumental_Almacen == null ? new SqlParameter { ParameterName =
"@IdTipoDocumental_Almacen", Value = DBNull.Value } : new SqlParameter { ParameterName =
"@IdTipoDocumental_Almacen", Value = idTipoDocumental_Almacen }
          , new SqlParameter { ParameterName = "@TipoExp_CHJ", Value = tipoExp_CHJ }
```

```
            , new SqlParameter { ParameterName = "@IdTipoExp_Verti2", Value =
idTipoExp_Verti2 }
            , new SqlParameter { ParameterName = "@IdMov_Verti2", Value = idMov_Verti2 }
        };

        string sql = "INSERT INTO [dbo].[Enlaces_ClavesCHJ_MovimientosVerti2] " +
            "( " +
            "  [Clave] " +
            ", [IdHito_FileServer] " +
            ", [IdTipoDocumental_Almacen] " +
            ", [TipoExp_CHJ] " +
            ", [IdTipoExp_Verti2] " +
            ", [IdMov_Verti2] " +
            ") " +
            "VALUES" +
            "( " +
            "  @Clave" +
            ", @IdHito_FileServer" +
            ", @IdTipoDocumental_Almacen" +
            ", @TipoExp_CHJ" +
            ", @IdTipoExp_Verti2" +
            ", @IdMov_Verti2" +
            ")";

        return DbBasic.ExecNonQuery(ref this.conn, sql, pars);
    }
```

edited Feb 18 at 11:24

answered Feb 18 at 11:17

Ángel Ibáñez

79 5

protected by Rawling Feb 18 at 11:46

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?