1. TOMA UNA LISTA DE ID’s y los pone en var
2. Combierte el var en DataTable
3. Llama a StoreProcedure
4. StoreProcedure tiene parametron tipo DataTable

// GET LIST OF ID’s

var attuids = from i in deactivationItems

select new

{

i.ID

};

// Load data from the list of ID’s

DataTable employees = DataAccess.GetEmployees(attuids.CopyToDataTable());

// Convert list of ID’s into DataTable

public static class CustomLINQtoDataSetMethods

{

public static DataTable CopyToDataTable<T>(this IEnumerable<T> source)

{

return new ObjectShredder<T>().Shred(source, null, null);

}

public static DataTable CopyToDataTable<T>(this IEnumerable<T> source,

DataTable table, LoadOption? options)

{

return new ObjectShredder<T>().Shred(source, table, options);

}

}

public static DataTable GetEmployeesFromFieldGlass(DataTable attuids)

{

DataTable dt = new DataTable();

SqlConnection connection = null;

SqlCommand command = null;

SqlDataAdapter adapter = null;

try

{

string connectionstrings = ConfigurationManager.ConnectionStrings["ATTUID"].ConnectionString;

connection = new SqlConnection(connectionstrings);

command = new SqlCommand("GetEmployees", connection);

command.CommandType = CommandType.StoredProcedure;

command.Parameters.Add("@ATTUIDs", SqlDbType.Structured).Value = attuids;

adapter = new SqlDataAdapter(command);

adapter.Fill(dt);

}

catch (Exception ex)

{

log.Error(ex);

}

finally

{

if (adapter != null)

adapter.Dispose();

if (command != null)

command.Dispose();

if (connection != null)

{

if (connection.State != ConnectionState.Closed)

connection.Close();

connection.Dispose();

}

}

return dt;

}

ALTER PROCEDURE [dbo].[GetEmployeesFromFieldGlass]

@ATTUIDs StringIdTableType readonly

AS

BEGIN

SET NOCOUNT ON;

select \* from vFieldGlassReport t1

where ATTUID in (select distinct ID from @ATTUIDs)

and END\_DATE = (SELECT MAX(END\_DATE) from vFieldGlassReport

where ATTUID = t1.ATTUID )

order by ATTUID

END