Backup code of Stock control system for Imperial bedding that might be needed later

public static string CreateTABLE(string tableName, DataTable table)

{

string sqlsc;

sqlsc = "CREATE TABLE " + tableName + "(";

for (int i = 0; i < table.Columns.Count; i++)

{

sqlsc += "\n [" + table.Columns[i].ColumnName + "] ";

string columnType = table.Columns[i].DataType.ToString();

switch (columnType)

{

case "System.Int32":

sqlsc += " int ";

break;

case "System.Int64":

sqlsc += " bigint ";

break;

case "System.Int16":

sqlsc += " smallint";

break;

case "System.Byte":

sqlsc += " tinyint";

break;

case "System.Decimal":

sqlsc += " decimal ";

break;

case "System.DateTime":

sqlsc += " datetime ";

break;

case "System.String":

default:

sqlsc += string.Format(" nvarchar({0}) ", table.Columns[i].MaxLength == -1 ? "max" : table.Columns[i].MaxLength.ToString());

break;

}

if (table.Columns[i].AutoIncrement)

sqlsc += " IDENTITY(" + table.Columns[i].AutoIncrementSeed.ToString() + "," + table.Columns[i].AutoIncrementStep.ToString() + ") ";

if (!table.Columns[i].AllowDBNull)

sqlsc += " NOT NULL ";

sqlsc += ",";

}

return sqlsc.Substring(0, sqlsc.Length - 1) + "\n)";

}

public static string GetCreateTableSql(DataTable table)

{

StringBuilder sql = new StringBuilder();

StringBuilder alterSql = new StringBuilder();

sql.AppendFormat("CREATE TABLE [{0}] (", table.TableName);

for (int i = 0; i < table.Columns.Count; i++)

{

bool isNumeric = false;

bool usesColumnDefault = true;

sql.AppendFormat("\n\t[{0}]", table.Columns[i].ColumnName);

switch (table.Columns[i].DataType.ToString().ToUpper())

{

case "SYSTEM.INT16":

sql.Append(" smallint");

isNumeric = true;

break;

case "SYSTEM.INT32":

sql.Append(" int");

isNumeric = true;

break;

case "SYSTEM.INT64":

sql.Append(" bigint");

isNumeric = true;

break;

case "SYSTEM.DATETIME":

sql.Append(" datetime");

usesColumnDefault = false;

break;

case "SYSTEM.STRING":

sql.AppendFormat(" nvarchar({0})", table.Columns[i].MaxLength);

break;

case "SYSTEM.SINGLE":

sql.Append(" single");

isNumeric = true;

break;

case "SYSTEM.DOUBLE":

sql.Append(" double");

isNumeric = true;

break;

case "SYSTEM.DECIMAL":

sql.AppendFormat(" decimal(18, 6)");

isNumeric = true;

break;

default:

sql.AppendFormat(" nvarchar({0})", table.Columns[i].MaxLength);

break;

}

if (table.Columns[i].AutoIncrement)

{

sql.AppendFormat(" IDENTITY({0},{1})",

table.Columns[i].AutoIncrementSeed,

table.Columns[i].AutoIncrementStep);

}

else

{

// DataColumns will add a blank DefaultValue for any AutoIncrement column.

// We only want to create an ALTER statement for those columns that are not set to AutoIncrement.

if (table.Columns[i].DefaultValue != null)

{

if (usesColumnDefault)

{

if (isNumeric)

{

alterSql.AppendFormat("\nALTER TABLE {0} ADD CONSTRAINT [DF\_{0}\_{1}] DEFAULT ({2}) FOR [{1}];",

table.TableName,

table.Columns[i].ColumnName,

table.Columns[i].DefaultValue);

}

else

{

alterSql.AppendFormat("\nALTER TABLE {0} ADD CONSTRAINT [DF\_{0}\_{1}] DEFAULT ('{2}') FOR [{1}];",

table.TableName,

table.Columns[i].ColumnName,

table.Columns[i].DefaultValue);

}

}

else

{

// Default values on Date columns, e.g., "DateTime.Now" will not translate to SQL.

// This inspects the caption for a simple XML string to see if there is a SQL compliant default value, e.g., "GETDATE()".

try

{

System.Xml.XmlDocument xml = new System.Xml.XmlDocument();

xml.LoadXml(table.Columns[i].Caption);

alterSql.AppendFormat("\nALTER TABLE {0} ADD CONSTRAINT [DF\_{0}\_{1}] DEFAULT ({2}) FOR [{1}];",

table.TableName,

table.Columns[i].ColumnName,

xml.GetElementsByTagName("defaultValue")[0].InnerText);

}

catch(Exception f)

{

MessageBox.Show("Error is: " + f);

}

}

}

}

if (!table.Columns[i].AllowDBNull)

{

sql.Append(" NOT NULL");

}

sql.Append(",");

}

if (table.PrimaryKey.Length > 0)

{

StringBuilder primaryKeySql = new StringBuilder();

primaryKeySql.AppendFormat("\n\tCONSTRAINT PK\_{0} PRIMARY KEY (", table.TableName);

for (int i = 0; i < table.PrimaryKey.Length; i++)

{

primaryKeySql.AppendFormat("{0},", table.PrimaryKey[i].ColumnName);

}

primaryKeySql.Remove(primaryKeySql.Length - 1, 1);

primaryKeySql.Append(")");

sql.Append(primaryKeySql);

}

else

{

sql.Remove(sql.Length - 1, 1);

}

sql.AppendFormat("\n);\n{0}", alterSql.ToString());

return sql.ToString();

}