**TradeProcessor Example**

1. It reads every line from a Stream parameter, storing each line in a list of strings.
2. It parses out individual fields from each line and stores them in a more structured list of Trade-Record instances.
3. The parsing includes some validation and some logging to the console.
4. Each TradeRecord is enumerated, and a stored procedure is called to insert the trades into a database.

public class TradeProcessor  
{  
    public void ProcessTrades(System.IO.Stream stream)  
    {  
        // read rows  
        var lines = new List<string>();  
        using(var reader = new System.IO.StreamReader(stream))  
        {  
            string line;  
            while((line = reader.ReadLine()) != null)  
            {  
                lines.Add(line);  
            }  
        }  
  
        var trades = new List<TradeRecord>();  
  
        var lineCount = 1;  
        foreach(var line in lines)  
        {  
            var fields = line.Split(new char[] { ',' });  
  
            if(fields.Length != 3)  
            {  
                Console.WriteLine("WARN: Line {0} malformed. Only {1} field(s) found.",  
  lineCount, fields.Length);  
                continue;  
            }  
  
            if(fields[0].Length != 6)  
            {  
                Console.WriteLine("WARN: Trade currencies on line {0} malformed: '{1}'",  
  lineCount, fields[0]);  
                continue;  
            }  
  
            int tradeAmount;  
            if(!int.TryParse(fields[1], out tradeAmount))  
            {  
                Console.WriteLine("WARN: Trade amount on line {0} not a valid integer:  
  '{1}'", lineCount, fields[1]);  
            }  
  
            decimal tradePrice;  
            if (!decimal.TryParse(fields[2], out tradePrice))  
            {  
                Console.WriteLine("WARN: Trade price on line {0} not a valid decimal:  
  '{1}'", lineCount, fields[2]);  
            }  
  
            var sourceCurrencyCode = fields[0].Substring(0, 3);  
            var destinationCurrencyCode = fields[0].Substring(3, 3);  
  
            // calculate values  
            var trade = new TradeRecord  
            {  
                SourceCurrency = sourceCurrencyCode,  
                DestinationCurrency = destinationCurrencyCode,  
                Lots = tradeAmount / LotSize,  
                Price = tradePrice  
            };  
  
            trades.Add(trade);  
  
            lineCount++;  
        }  
  
        using (var connection = new System.Data.SqlClient.SqlConnection("Data  
  Source=(local);Initial Catalog=TradeDatabase;Integrated Security=True"))  
        {  
            connection.Open();  
            using (var transaction = connection.BeginTransaction())  
            {  
                foreach(var trade in trades)  
                {  
                    var command = connection.CreateCommand();  
                    command.Transaction = transaction;  
                    command.CommandType = System.Data.CommandType.StoredProcedure;  
                    command.CommandText = "dbo.insert\_trade";  
                    command.Parameters.AddWithValue("@sourceCurrency", trade.  
  SourceCurrency);  
                    command.Parameters.AddWithValue("@destinationCurrency", trade.  
  DestinationCurrency);  
                    command.Parameters.AddWithValue("@lots", trade.Lots);  
                    command.Parameters.AddWithValue("@price", trade.Price);  
  
                    command.ExecuteNonQuery();  
                }  
  
                transaction.Commit();  
            }  
            connection.Close();  
        }  
  
        Console.WriteLine("INFO: {0} trades processed", trades.Count);  
    }  
  
    private static float LotSize = 100000f;  
}

class TradeRecord

{

public string SourceCurrency { get; set; }

public string DestinationCurrency { get; set; }

public float Lots { get; set; }

public decimal Price { get; set; }

}

**Refactoring for clarity**