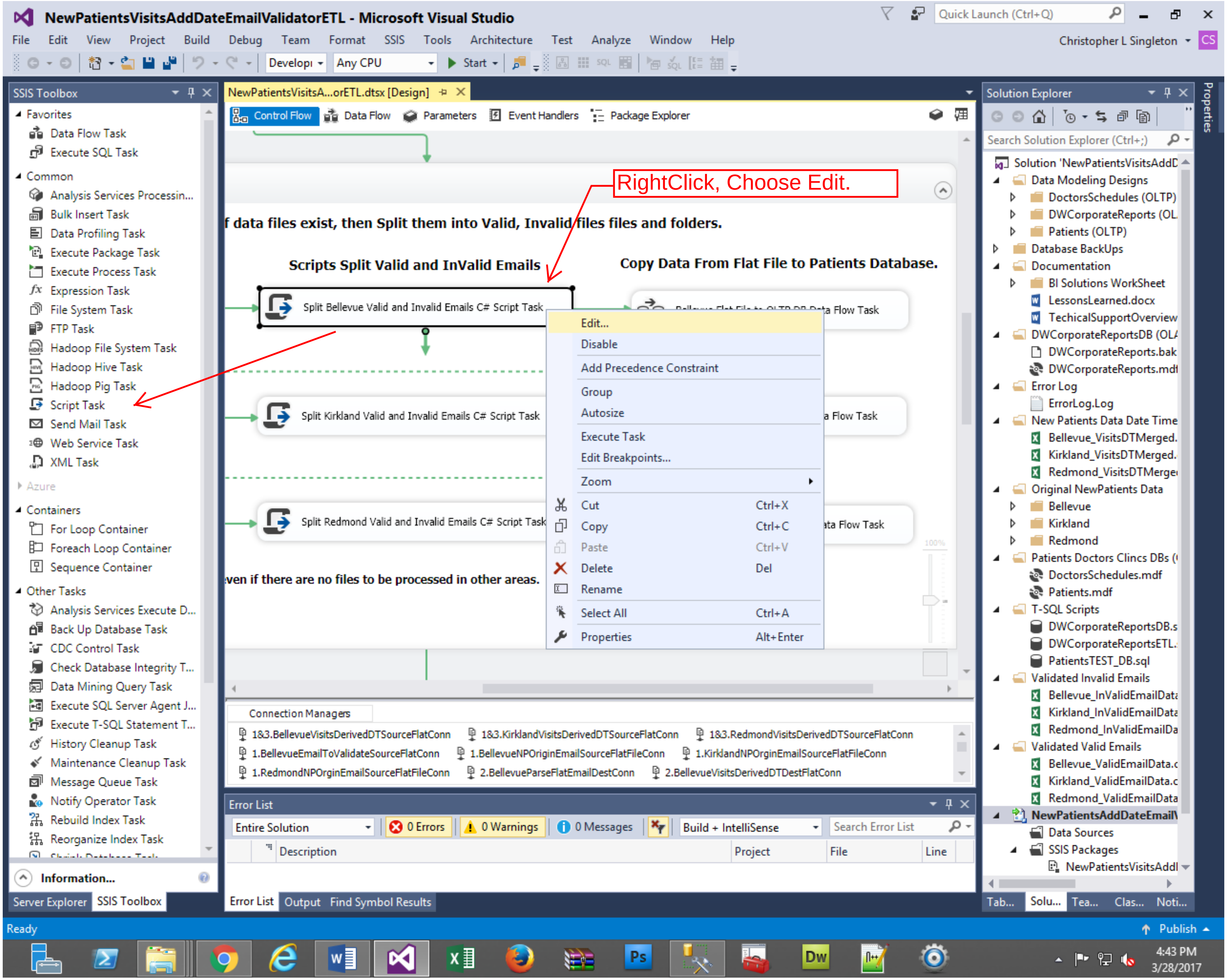
How to validate email addresses in SSIS using C# (Advanced ETL)

Programming By: Chris Singleton



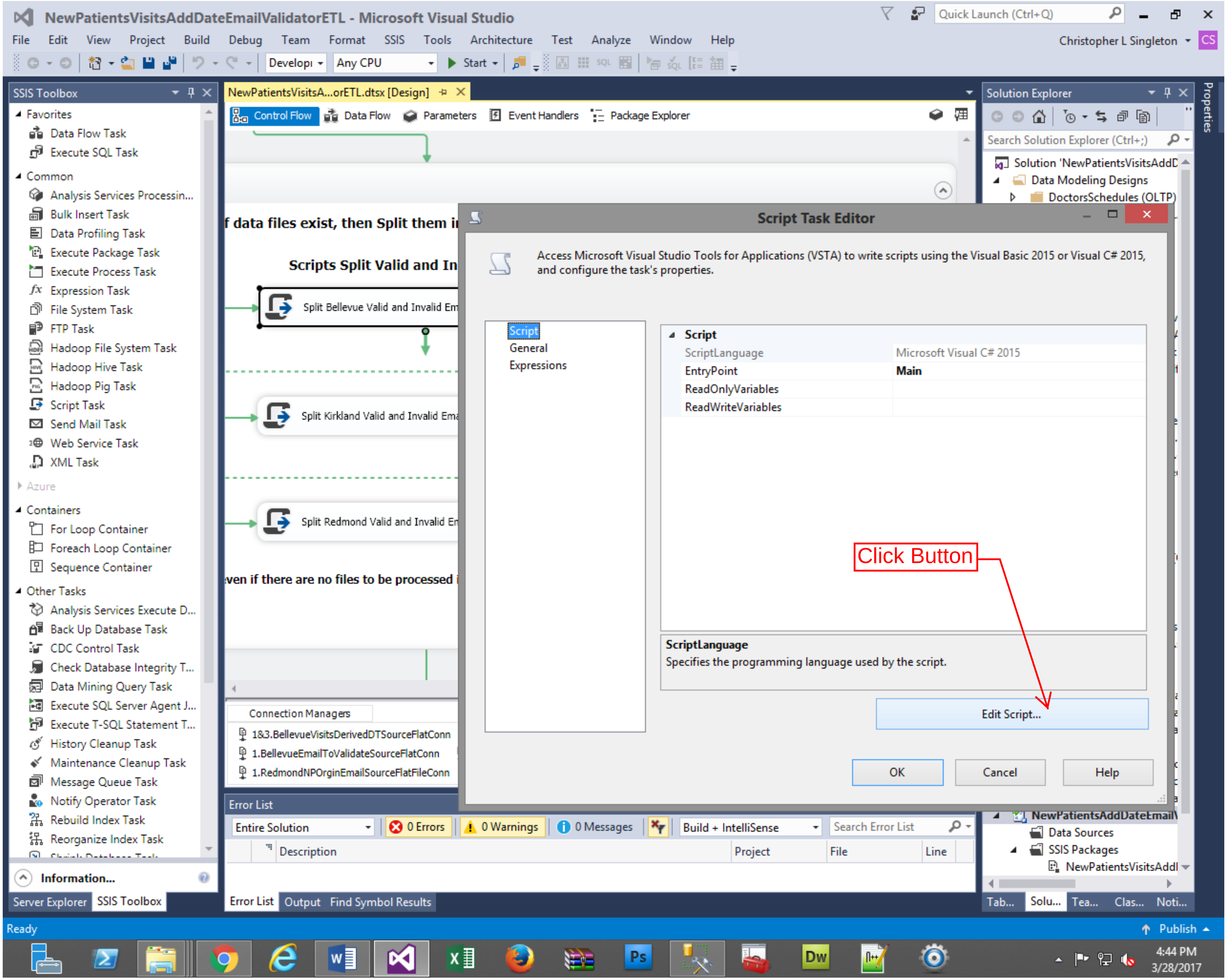
RightClick, Choose Edit.

f data files exist, then Split them into Valid, Invalid files files and folders.

Scripts Split Valid and InValid Emails

Copy Data From Flat File to Patients Database.

ven if there are no files to be processed in other areas.



NewPatientsVisitsAddDateEmailValidatorETL - Microsoft Visual Studio

Script Task Editor

Access Microsoft Visual Studio Tools for Applications (VSTA) to write scripts using the Visual Basic 2015 or Visual C# 2015, and configure the task's properties.

Script

ScriptLanguage: Microsoft Visual C# 2015

EntryPoint: Main

ReadOnlyVariables:

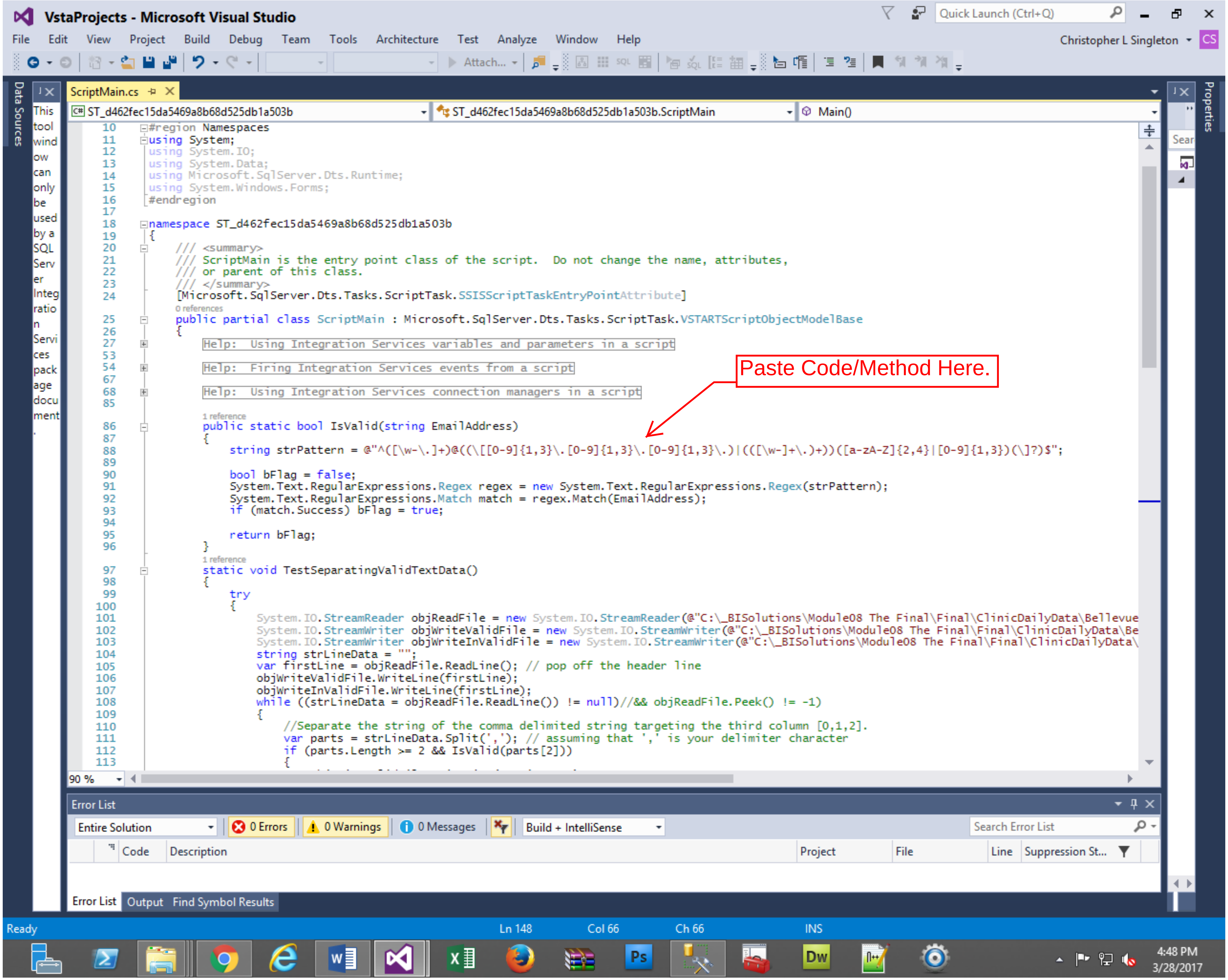
ReadWriteVariables:

ScriptLanguage

Specifies the programming language used by the script.

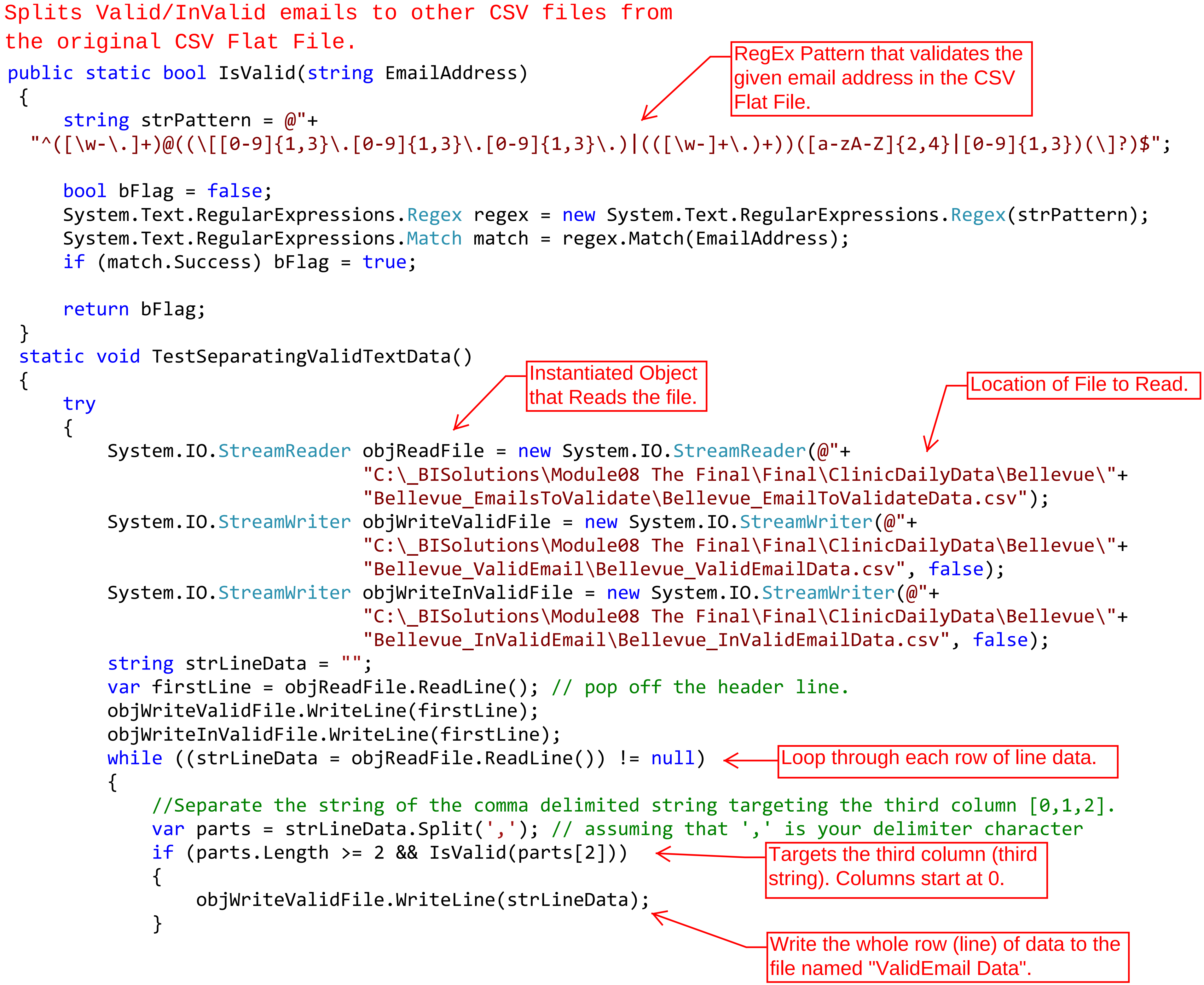
Click Button

Note: When you hit the edit button, a new instance of Visual Studio comes up with a C# window.


VstaProjects - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
File Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help
Christopher L Singleton
Attach...
Data Sources
This tool window can only be used by a SQL Server Integration Services package document.
ScriptMain.cs
ST_d462fec15da5469a8b68d525db1a503b
ST_d462fec15da5469a8b68d525db1a503b.ScriptMain
Main()
Properties
Sear
90 %
Error List
Entire Solution
0 Errors
0 Warnings
0 Messages
Build + IntelliSense
Search Error List
Code Description Project File Line Suppression St...
Error List Output Find Symbol Results
Ready Ln 148 Col 66 Ch 66 INS
4:48 PM
3/28/2017

Paste Code/Method Here.

```
#region Namespaces
using System;
using System.IO;
using System.Data;
using Microsoft.SqlServer.Dts.Runtime;
using System.Windows.Forms;
#endregion

namespace ST_d462fec15da5469a8b68d525db1a503b
{
    /// <summary>
    /// ScriptMain is the entry point class of the script.  Do not change the name, attributes,
    /// or parent of this class.
    /// </summary>
    [Microsoft.SqlServer.Dts.Tasks.ScriptTask.SSISScriptTaskEntryPointAttribute]
    0 references
    public partial class ScriptMain : Microsoft.SqlServer.Dts.Tasks.ScriptTask.VSTARTScriptObjectModelBase
    {
        Help:  Using Integration Services variables and parameters in a script

        Help:  Firing Integration Services events from a script

        Help:  Using Integration Services connection managers in a script

        1 reference
        public static bool IsValid(string EmailAddress)
        {
            string strPattern = @"^([\w-\.]+)@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.)|(([\w-]+\.)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)$";

            bool bFlag = false;
            System.Text.RegularExpressions.Regex regex = new System.Text.RegularExpressions.Regex(strPattern);
            System.Text.RegularExpressions.Match match = regex.Match(EmailAddress);
            if (match.Success) bFlag = true;

            return bFlag;
        }
        1 reference
        static void TestSeparatingValidTextData()
        {
            try
            {
                System.IO.StreamReader objReadFile = new System.IO.StreamReader(@"C:\_BISolutions\Module08 The Final\Final\ClinicDailyData\Bellevue
                System.IO.StreamWriter objWriteValidFile = new System.IO.StreamWriter(@"C:\_BISolutions\Module08 The Final\Final\ClinicDailyData\Be
                System.IO.StreamWriter objWriteInValidFile = new System.IO.StreamWriter(@"C:\_BISolutions\Module08 The Final\Final\ClinicDailyData\
                string strLineData = "";
                var firstLine = objReadFile.ReadLine(); // pop off the header line
                objWriteValidFile.WriteLine(firstLine);
                objWriteInValidFile.WriteLine(firstLine);
                while ((strLineData = objReadFile.ReadLine()) != null)//&& objReadFile.Peek() != -1)
                {
                    //Separate the string of the comma delimited string targeting the third column [0,1,2].
                    var parts = strLineData.Split(','); // assuming that ',' is your delimiter character
                    if (parts.Length >= 2 && IsValid(parts[2]))
                    {
```



Splits Valid/InValid emails to other CSV files from
the original CSV Flat File.

```
public static bool IsValid(string EmailAddress)
 {
     string strPattern = @"+
  "^([\w-\.]+)@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.)|(([\w-]+\.)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)$";

     bool bFlag = false;
     System.Text.RegularExpressions.Regex regex = new System.Text.RegularExpressions.Regex(strPattern);
     System.Text.RegularExpressions.Match match = regex.Match(EmailAddress);
     if (match.Success) bFlag = true;

     return bFlag;
 }
 static void TestSeparatingValidTextData()
 {
     try
     {
         System.IO.StreamReader objReadFile = new System.IO.StreamReader(@"+
                                "C:\_BISolutions\Module08 The Final\Final\ClinicDailyData\Bellevue\"+
                                "Bellevue_EmailsToValidate\Bellevue_EmailToValidateData.csv");
         System.IO.StreamWriter objWriteValidFile = new System.IO.StreamWriter(@"+
                                "C:\_BISolutions\Module08 The Final\Final\ClinicDailyData\Bellevue\"+
                                "Bellevue_ValidEmail\Bellevue_ValidEmailData.csv", false);
         System.IO.StreamWriter objWriteInValidFile = new System.IO.StreamWriter(@"+
                                "C:\_BISolutions\Module08 The Final\Final\ClinicDailyData\Bellevue\"+
                                "Bellevue_InValidEmail\Bellevue_InValidEmailData.csv", false);
         string strLineData = "";
         var firstLine = objReadFile.ReadLine(); // pop off the header line.
         objWriteValidFile.WriteLine(firstLine);
         objWriteInValidFile.WriteLine(firstLine);
         while ((strLineData = objReadFile.ReadLine()) != null)
         {
             //Separate the string of the comma delimited string targeting the third column [0,1,2].
             var parts = strLineData.Split(','); // assuming that ',' is your delimiter character
             if (parts.Length >= 2 && IsValid(parts[2]))
             {
                 objWriteValidFile.WriteLine(strLineData);
             }
```

RegEx Pattern that validates the given email address in the CSV Flat File.

Instantiated Object that Reads the file.

Location of File to Read.

Loop through each row of line data.

Targets the third column (third string). Columns start at 0.

Write the whole row (line) of data to the file named "ValidEmail Data".

```
            else
            {
                objWriteInValidFile.WriteLine(strLineData);
            }
        }
```

If false, then writes to InValidEmailData file.

```
        objReadFile.Close();
        objWriteValidFile.Close();
        objWriteInValidFile.Close();
```

Close Methods, Close Read/Write objects.

```
    }
    catch (System.Exception objException)
    {
        System.Console.WriteLine(objException.ToString());
        throw objException; //Must add this for the Main method to catch this exception.
    }
}
/// <summary>
/// This method is called when this script task executes in the control flow.
/// Before returning from this method, set the value of Dts.TaskResult to indicate success or failure.
/// To open Help, press F1.
/// </summary>
public void Main()
{
    try
    {
        TestSeparatingValidTextData();
```

Call Method

```
        Dts.TaskResult = (int)ScriptResults.Success;
    }
    catch (Exception objException)
```

Handle Errors

Message to show when an error happens.

```
    {
        string strMessage = @"On Error, please check the file is in C:\_BISolutions\"+
                             "Module08 The Final\Final\ClinicDailyData\Bellevue\ Folder";
        Dts.Events.FireError(0, strMessage, objException.ToString(), string.Empty, 0);
        Dts.TaskResult = (int)ScriptResults.Failure;
    }
}
```

```
#region ScriptResults declaration
/// <summary>
/// This enum provides a convenient shorthand within the scope of this class for setting the
/// result of the script.
///
/// This code was generated automatically.
/// </summary>
enum ScriptResults
{
    Success = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Success,
    Failure = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Failure
};
#endregion
```

Added the code again for re-usability.

```
public static bool IsValid(string EmailAddress)
 {
     string strPattern = @"+
  "^([\w-\.]+)@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.)|(([\w-]+\.)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)$";

     bool bFlag = false;
     System.Text.RegularExpressions.Regex regex = new System.Text.RegularExpressions.Regex(strPattern);
     System.Text.RegularExpressions.Match match = regex.Match(EmailAddress);
     if (match.Success) bFlag = true;

     return bFlag;
 }
 static void TestSeparatingValidTextData()
 {
     try
     {
         System.IO.StreamReader objReadFile = new System.IO.StreamReader(@"+
                                "C:\_BISolutions\Module08 The Final\Final\ClinicDailyData\Bellevue\"+
                                "Bellevue_EmailsToValidate\Bellevue_EmailToValidateData.csv");
         System.IO.StreamWriter objWriteValidFile = new System.IO.StreamWriter(@"+
                                "C:\_BISolutions\Module08 The Final\Final\ClinicDailyData\Bellevue\"+
                                "Bellevue_ValidEmail\Bellevue_ValidEmailData.csv", false);
         System.IO.StreamWriter objWriteInValidFile = new System.IO.StreamWriter(@"+
                                "C:\_BISolutions\Module08 The Final\Final\ClinicDailyData\Bellevue\"+
                                "Bellevue_InValidEmail\Bellevue_InValidEmailData.csv", false);
         string strLineData = "";
         var firstLine = objReadFile.ReadLine(); // pop off the header line.
         objWriteValidFile.WriteLine(firstLine);
         objWriteInValidFile.WriteLine(firstLine);
         while ((strLineData = objReadFile.ReadLine()) != null)
         {
             //Separate the string of the comma delimited string targeting the third column [0,1,2].
             var parts = strLineData.Split(','); // assuming that ',' is your delimiter character
             if (parts.Length >= 2 && IsValid(parts[2]))
             {
                 objWriteValidFile.WriteLine(strLineData);
             }
```

```
                    else
                    {
                        objWriteInValidFile.WriteLine(strLineData);
                    }
                }
                objReadFile.Close();
                objWriteValidFile.Close();
                objWriteInValidFile.Close();
            }
            catch (System.Exception objException)
            {
                System.Console.WriteLine(objException.ToString());
                throw objException; //Must add this for the Main method to catch this exception.
            }
        }
        /// <summary>
        /// This method is called when this script task executes in the control flow.
        /// Before returning from this method, set the value of Dts.TaskResult to indicate success or failure.
        /// To open Help, press F1.
        /// </summary>
        public void Main()
        {
            try
            {
                TestSeparatingValidTextData();
                Dts.TaskResult = (int)ScriptResults.Success;
            }
            catch (Exception objException)
            {
                string strMessage = @"On Error, please check the file is in C:\_BISolutions\"+
                                     "Module08 The Final\Final\ClinicDailyData\Bellevue\ Folder";
                Dts.Events.FireError(0, strMessage, objException.ToString(), string.Empty, 0);
                Dts.TaskResult = (int)ScriptResults.Failure;
            }
        }
```

```
#region ScriptResults declaration
/// <summary>
/// This enum provides a convenient shorthand within the scope of this class for setting the
/// result of the script.
///
/// This code was generated automatically.
/// </summary>
enum ScriptResults
{
    Success = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Success,
    Failure = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Failure
};
#endregion
```