Using System.IO

Using System.IO.Compression

Using System.Data.Linq

Using System.Configuration

public class ZipFileProcess

{

public void process(){

string zippath = ConfigurationManager. AppSettings["zippath"];

string extractPath = ConfigurationManager AppSettings ["extractPath"];

var fileCount = Directory. GetFiles(zippath);

foreach (var file in fileCount)

{

string aaplno = n..

string extractFoldername =

string xmlfilename = "";

//check for zip file if (file.Contains(" .zip"))

if (lISZipValid(file))

{

LogError("The Zip file is corrupt or the files inside can not be queried.");

SendNotification("Error: Corrupt Zip file, The zip file" + file + " is corrupt or the files inside can not be queried.");

return;

}

{

if (!IsZipContentValid(file))

LogError ("The Zip file contains invalid files.");

SendNotification("Error: Invalid files in ZIP, the zip file" + file + "conatins invalid files");

return;

}

if (lIsXmiValid(file,zippath,out aaplno, out xmlfilename))

LogError("The XML file is invalid");

SendNotification ("Error: Invalid" + xmlfilename + " file. The file in " + file + " is invalid.");

return;

}

ExtractZipFile (aaplno, file, extractPath, out extractFoldername);

SendNotification("Zip file processed successfully, The zip file " + file+" has been processed successfully and extracted to" + extractPath + extractFoldername);

Log ("Zip file processed successfully, The zip file " + file + " has been processed successfully and extracted to" + extractPath + extractFoldername);

}

}

}

//Check zip file is not corrupt and files inside can be querired.

1 reference

public bool ISZipValid(string path)

{

if (!File.Exists (path))

{

return false;

?

try

{

using (var zipFile = ZipFile. OpenRead (path))

{

var entries = zipFile. Entries;

return true;

8 о

}

catch (InvalidDataException ex)

{

?

ex. Message. ToStringO;

return false;

}

//Validate zip file contains only files with allowed extension and xml file

public bool IsZipContentValid(string path)

{

try

{

U processo

using (ZipArchive archive = ZipFile.OpenRead (path))

{

foreach (ZipArchiveEntry entry in archive. Entries)

{

if (entry. Name. EndsWith(" docx",

StringComparison.OrdinalIgnoreCase) ||

entry. Name. EndsWith(" .pdf" entry. Name. Endswith(" xIxs"

StringComparison. OrdinalIgnoreCase) ||

StringComparison.OrdinalIgnoreCase) ||

entry. Name. EndsWith(" png",

StringComparison.OrdinalIgnoreCase) ||

entry. Name. EndsWith(" XML" StringComparison.OrdinalIgnoreCase)

)

}

return true;

}

}

return false;

}

}

{

catch (Exception ex)

ex. Message. ToStringO;

return false;

}

}

public bool IsXmlValid(string filePath, string XSDfilePath, out string applicationNo, out string xmlfilename)

applicationNo = "o".

xmlfilename =

bool result = true;

try

{

using (ZipArchive archive = ZipFile.OpenRead(filePath))

{

foreach (ZipArchiveEntry entry in archive. Entries)

{

if (entry. Name. Endswith(" XML", StringComparison .OrdinalIgnoreCase))

{

xmlfilename = entry. Name;

var schemas = new XmlSchemaset();

schemas .Add("", XSDfilePath + "party.xsd");

Document doc = new Document();

Stream stream = entry. Open();

using (StreamReader sr = new StreamReader (entry. Open ()))

{

string fileContent = sr.ReadToEnd();

}

doc = XDocument. Load (stream);

doc. Validate (schemas, (sender, e) =>

{

result = false;

});

{

var getResult = from d in doc.Descendants ("party")

select new{

applicationNo = d.Element ("applicationno"). Value};

foreach (var no in getResult){

applicationNo = no.applicationNo;

}

}

}

return false;

}

}

Catch(Exception ex)

{

ex.Message.ToString();

result= false;

}

return result;

}

public void ExtractZipFile(string applNo,string filePath,string extractPath,out string extractFolderName)

{

extractFoldername = "”;

try

{

Guid obj = Guid. NewGuid();

extractFoldername = applNo + obj. ToString();

bool folderExists = Directory.Exists(extractPath + extractFoldername);

if (!folderExists)

Directory.CreateDirectory(extractPath + extractFoldername);

using (ZipArchive archive = ZipFile.OpenRead(filePath))

{

foreach (ZipArchiveEntry entry in archive.Entries)

{

if (entry. Name != "")

{

entry. ExtractToFile(Path. Combine(extractPath + extractFoldername, entry. Name));

}

}

}

}

{

catch (Exception ex)

ex Message.ToString();

}

}

public static void Email(string htmlString)

{

try

string fromMailAddress - ConfigurationManager.AppSettings!"FromMailAddress":

string tollailAddress = SonfigurationNanager.Apesettingsl"TolailAddressi

string password = Sonfigurationtlanager AppSettingsl"PasSwozd]

MailMessage message = new MailMessage);

SmtpClient smtp = new SmtpClientO;

message.From = new MailAddress (fromMailAddress);

message. To. Add(new MailAddress(toMailAddress));

message. Subject = "Zip File Processor™

message. IsBodyHtml = true; //to make message body as html

message. Body = htmlString;

smtp.Port = 587;

smtp.Host = "[smtp.gmail.com](http://smtp.gmail.com/)"; //for gmail host

smtp. EnableSsl = true;

smtp.UseDefaultCredentials = false;

smtp. Credentials = new NetworkCredential(fromMailAddress, password);

smtp.DeliveryMethod = SmtpDeliveryMethod. Network;

smtp. Send (message);

}

catch (Exception ex) {

ex. Message. ToString();

}

}

//Logerror

public void LogError (string message)

{

string LogerrorFilePath = Sonfigurationtanaget.AppSettings["Logerrageilepath"1;

string errormessage = string. Format (" {0}{1}"

DateTime .Now, message);

File.AppendAllText(logErrorFilePath, errormessage + Environment .NewLine);

}

public void Log(string message)

{

string logFilePath = ConfigurationManager-AppSettings["LoqFilePath"];

string errormessage = string. Format ("{0}{1}

DateTime.Now, message);

File.AppendAllText(LogFilePath, errormessage + Environment .NewLine);

}

4 references

public void SendNotification(string message)

{

Email (message);

}

}