متاسفانه در زبان هاي C# و VB امكاني براي زيپ كردن فايل ها به طور مستقيم وجود ندارد . با اين وجود در زبان J#راهي براي zip و unzip نمودن فايل ها وجود دارد.

در اين مقاله يک كتابخانه (Library) خواهيم نوشت كه قابليت استفاده مجدد داشته و مي توانيم از آن در تمام پروژه هاي تحت ويندوز و وب براي zip و unzip كردن فايل ها استفاده كنيم.

مقدمه:

در بسياري مواقع ، در برنامه هايي كه مي نويسيم نياز داريم كه بعضي فايل ها مانند اسناد و فايل هاي XML و غيره را به صورت فشرده ذخيره كنيم و يا فايلي كه به صورت فشرده ، ذخيره شده است را از حالت فشردگي خارج نماييم.

مثلا فرض كنيد كه شما در برنامه وب خود ، امكاني داده ايد كه كاربران اسناد و يا فايل هاي خود را در صفحه اي براي شما آپلود كنند و بديهي است كه ترجيح مي دهيد فايل ها به صورت زيپ شده در فضاي سايت شما ذخيره شوند (به دلايل محدوديت فضا و غيره.(

براي zip و unzip كردن فايل ها در زبان C# معمولا از يكي از اين ۳ طريق استفاده مي شود :

۱. استفاده از كامپوننت ها و برنامه هاي مكمل (تهيه شده توسط ديگران)
۲. استفاده از كامپوننت هاي متن باز (Source Open) موجود.
۳. پياده سازي الگوريتم زيپ سازي فايل ها ، در برنامه

گزينه اول نيازمند دريافت مجوز استفاده و خريد كامپوننت و پرداخت هزينه مي باشد .

گزينه دوم يعني استفاده از كامپوننت هاي متن باز ، بسيار جالب مي باشد ولي با اين وجود مشكلات برنامه هاي متن باز ، اينجا نيز وجود دارد . يعني مشكل باگ هاي احتمالي و بروز رساني و پشتيباني كه مي تواند موجب پديد آمدن مشكلات بزرگي شود.

روش سوم بسيار مشكل بوده و نياز مند صرف هزينه و رمان زيادي مي باشد.

خوشبختانه زبان J# كه يكي از زبان هاي پلتفرم دات نت مي باشد ، داراي امكاني براي zip و unzip كردن فايل ها به وسيله كد مي باشد.

مزاياي استفاده از J# براي zip و unzip كردن فايل ها :

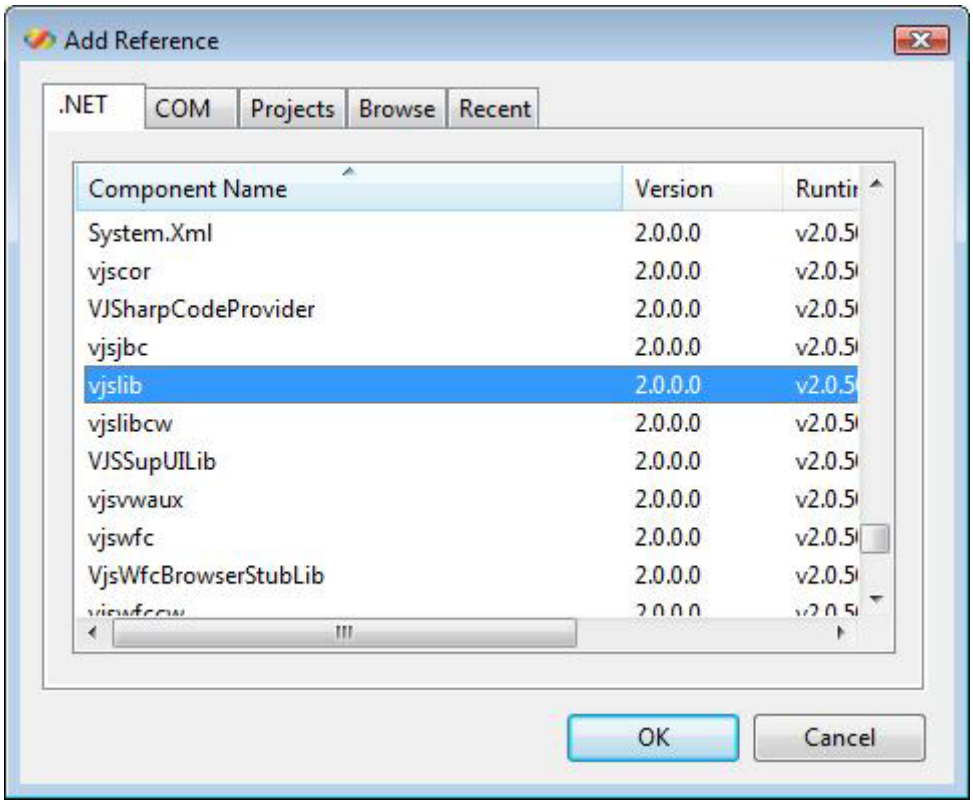
- J# يكي از زبان هاي موجود در پلتفرم دات نت بوده و بدون هزينه اضافي در دسترس ما مي باشد.
- زبان J# توسط شركت مايكروسافت پشتيباني و بروزرساني مي شود .
- ديگر نيازي به استفاده از برنامه هاي مكمل نداريم.

در اين قسمت با استفاده از كلاس هاي J# ، كامپوننتي در C# ايجاد خواهيم كرد و از آن در تمام برنامه هاي خود استفاده مي كنيم .

ايجاد يک كتابخانه از كلاس ها :

براي شروع كار يک پروژه Class Library در زبان C# ايجاد نموده و در آن با استفاده از كلاس ها و توابع J# كامپوننت را ايجاد خواهيم كرد.

يک پروژه Class Library ايجاد كنيد. در Solution Explorer روي نام پروژه كليک راست نموده و Add Reference را انتخاب كنيد. همانطور كه در شكل مي بينيد ، اسمبلي vjslib را به برنامه اضافه كنيد.



سپس فضا هاي نامي (namespace) زير را به برنامه اضافه كنيد

```
java.util;   •
java.util.zip;   •
java.io;   •
```

فضاي نامي java.util شامل كلاس هاي سودمندي براي ما مي باشد . فضاي نامي java.util.zip شامل كلاس هاي اصلي مرتبط به ايجاد فايل هاي زيپ مي باشد . و در نهايت فضاي نامي java.io شامل كلاس هاي مرتبط به IO مي باشد.

ما از كلاس هاي زير كه در فضاهاي نامي ذكر شده ، مي باشد ، استفاده مي كنيم.

- ZipFile
- ZipEntry
- InputStream
- OutputStream
- FileInputStream
- FileOutputStream
- ZipOutputStream
- Enumeration

كلاس ZipFile امكان ايجاد يک فايل زيپ از طريق برنامه نويسي مي دهد. يک ZipFile شامل صفر و يا بيشتر از اشياء ZipEntry و محتواي واقعي فايل هاي زيپ مي باشد.

كلاسهاي InputStream و OutputStream و FileInputStream و FileOutputStream ارائه دهنده استريم هاي به ترتيب اشاره گر به حافظه و مبتني برفايل مي باشند.

كلاس ZipOutputStream يک استريم قابل نوشتن (writeable) ايجاد نموده که به يک فايل زيپ اشاره مي کند . اين استريم مي تواند براي نوشتن اشياء ZipEntry و محتواي آنها در يک فايل زيپ استفاده شود.

ايجاد فايل هاي زيپ:

قبل از اينکه به نوشتن کدهاي مربوطه براي ايجاد و استخراج (Extract) فايل هاي زيپ بپردازيم ، اجازه بدهيد که به معرفي متدهايي که ما در آينده به آن نياز خواهيم داشت بپردازيم.

```
GetZippedItems()    •
CopyEntries() (two overloads)    •
CopyStream()    •
AddEntries()  •
RemoveEntries()  •
```

فشرده سازي تعدادي از عناصر، در فقط يک فايل زيپ:

متد GetZippedItems يک شي ZipFile را به عنوان پارامتر گرفته و يک ليست ژنريک از اشياء ZipEntry موجود در فايل را برمي گرداند.

متد GetZippedItems را در زير مشاهده مي کنيد.

```
private static List<ZipEntry> GetZippedItems(ZipFile file)

{

List<ZipEntry> entries = new List<ZipEntry>();

Enumeration e = file.entries();

while (true)

{

if (e.hasMoreElements())

{

ZipEntry entry = (ZipEntry)e.nextElement();

entries.Add(entry);

}

else

{

break;

}
```

```
}

return entries;

}
```

متد GetZippedItems يک شي ZipFile را به عنوان پارامتر گرفته و يک ليست ژنريک از اشياء ZipEntry را برمي گرداند.

داخل متد ، يک ليست ژنريک از نوع ZipEntry به نام entries ايجاد نموده ايم و سپس با فراخواني متد entries يک داده شمارشي (Enumeration) از اشياء ZipEntry ها پر مي کنيم و در نهايت ليست پر شده را برمي گردانده مي شود.

کپي کردن استريم ها:

در زمان اضافه کردن يا برداشتن فايل ها از يک فايل زيپ موجود ، ما نياز داريم که محتواي اصلي فايل را از استريم مبدأ به استريم مقصد کپي کنيم.

بنابراين نيازمند متدي هستيم تا اين عمل را براي ما انجام دهد .اين متد را CopyStream مي ناميم و در زير آن را مشاهده مي کنيد.

```
private static void CopyStream(InputStream source, OutputStream
destination)

{

sbyte[] buffer = new sbyte[8000];

int data;

while (true)

{

try

{

data = source.read(buffer, 0, buffer.Length);

if (data > 0)

{

destination.write(buffer, 0, data);

}

else

{

return;
```

```
}

}

catch (Exception ex)

{

string msg = ex.Message;

}

}

}
```

همانطور که مشاهده مي کنيد ، متد CopyStream دو پارامتر مي گيرد ، يکي از نوع InputStream که همان استريم منبع مي باشد و ديگري از نوع OutputStream که استريم مقصد مي باشد. با استفاده از متد read مربوط به استريم منبع ، به اندازه قطعات ۸۰۰۰ بايتي از آن خوانده و در استريم مقصد با استفاده از متد write موجود در OutputStream مي نويسيم.

کپي کردن اشياء: ZipEntry

کلاس هاي مربوط به فشرده سازي در J# به شما اجازه اضافه کردن و حذف کردن (Remove) فايل ها ، به طور مستقيم ، از يک فايل زيپ را نمي دهند. تنها راه اضافه کردن و حذف کردن فايل ها از يک فايل زيپ شده ، اين است که يک فايل زيپ جديد با عناصر مورد نياز آن ، ايجاد نموده و جايگزين (Replace) فايل زيپ اصلي نماييم.

بنابراين ما نيازمند متدي هستيم که اشياء ZipEntry را از يک فايل زيپ به فايل ديگر که مورد نظر ما مي باشد ، کپي کند. ما اين متد را CopyEntries مي ناميم و به صورت دو متد Overloads شده مي نويسيم . در زير اين متد را مشاده مي کنيد.

```
private static void CopyEntries(ZipFile source, ZipOutputStream
destination)

{

List<ZipEntry> entries = GetZippedItems(source);

foreach (ZipEntry entry in entries)

{

InputStream s = source.getInputStream(entry);

destination.putNextEntry(entry);

CopyStream(s, destination);

destination.closeEntry();

s.close();

}
```

```
}
```

```
private static void CopyEntries(ZipFile source, ZipOutputStream
destination, string[] entryNames)

{

List<ZipEntry> entries = GetZippedItems(source);

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

{

foreach (ZipEntry entry in entries)

{

if (entry.getName() == entryNames[i])

{

InputStream s = source.getInputStream(entry);

destination.putNextEntry(entry);

CopyStream(s, destination);

destination.closeEntry();

s.close();

}

}

}

}
```

اولين متد دو پارامتر دريافت مي كند. پارامتر اول ZipFile منبع مي باشد كه شامل ZipEntry هايي است كه قرار است كپي شوند. پارامتر دوم ZipOutputStream هدف مي باشد كه قرار است ZipEntry ها ، در آن نوشته شوند.

متد Overloads شده دوم ، داراي سه پارامتر مي باشد . ٢ پارامتر اول همانند متد قبلي مي باشند و پارامتر سوم حاوي نام entry هاي مشخصي مي باشد كه ما در نظر داريم كپي شوند.

فرق اين متد با متد قبلي در اين است كه در متد اول تمام entry ها به استريم مقصد كپي مي شوند ولي در متد دوم ، فقط entry هاي مشخصي كه ما در پارامتر سوم نام آن ها را ارسال مي كنيم ، به استريم مقصد كپي مي شوند (در ادامه مقاله از اين دو متد استفاده خواهيم كرد و تفاوت آنها مشاده خواهيد كرد.)

هر دوي متد هاي Overloads شده بالا ، با استفاده از متد GetZippedItems كه قبلا نوشته ايم ، ليستي از ZipEntryها را بازيابي مي كنند و در داخل ليستي به نام entries قرار مي دهند . سپس entries به ZipOutputStreamانتقال مي يابد . متد putNextEntry از كلاس ZipOutputStream ، يک ZipEntry را كه بايد به فايل زيپ اضافه شود ، را گرفته و در فايل زيپ مي نويسد.

متد getInputStream از کلاس ZipFile ، یک ZipEntry را گرفته و یک InputStream که به یک entry اشاره مي کند را برمي گرداند.(اين استريم توسط متد CopyStream که قبلا نوشته ايم براي خواندن اطلاعات از entry مربوطه استفاده مي شود)

به ياد داشته باشيد که ZipEntry به سادگي متا ديتاي مربوط به entry ، را ايجاد مي کند و ZipFile منبع ، با استفاده از متد getInputStream ، محتواي واقعي فايل را ايجاد خواهد کرد که به شکل InputStream مي باشد.

در نهايت با فراخواني متد closeEntry از کلاس ZipOutputStream نوشتن entry پايان مي يابد.

اضافه کردن entries ، به فايل زيپ موجود:

متدي به نام AddEntries مي نويسيم که اشياء ZipEntry را به يک فايل زيپ اضافه مي کند.

```
private static void AddEntries(ZipFile file, string[] newFiles)

{

string fileName = file.getName();

string tempFileName = Path.GetTempFileName();

ZipOutputStream destination = new ZipOutputStream(new
FileOutputStream(tempFileName));

try

{

CopyEntries(file, destination);

if (newFiles != null)

{

foreach (string f in newFiles)

{

ZipEntry z = new ZipEntry(f.Remove(0, Path.GetPathRoot(f).Length));

z.setMethod(ZipEntry.DEFLATED);

destination.putNextEntry(z);

try

{

FileInputStream s = new FileInputStream(f);

try

{
```

```
CopyStream(s, destination);

}

finally

{

s.close();

}

}

finally

{

destination.closeEntry();

}

}

}

}

finally

{

destination.close();

}

file.close();

System.IO.File.Copy(tempFileName, fileName, true);

System.IO.File.Delete(tempFileName);

}
```

همانطور كه مي بينيد با استفاده از متد getName مسير كامل فايل را بدست مي آوريم و با استفاده از متد GetTempFileName، از كلاس Syste.IO ، يک نام فايل موفقتي (temporary file name) ايجاد مي كنيم. ممكن است اكنون براي شما اين سوال به وجود آمده باشد كه فايل موقتي را براي چه ما ايجاد مي كنيم؟!

متد AddEntries را زماني فراخواني مي كنيم كه بخواهيم يک فايل زيپ جديد ايجاد نموده و يا عناصري را به يک فايل زيپ موجود ، اضافه كنيم.

كلاس هاي فشرده سازي در J# اجازه تغيير دادن يک فايل زيپ را به صورت مستقيم نمي دهند ، بنابراين بايد يک فايل زيپ جديد ، با عناصر مورد نيازمان را به طور موقت ايجاد نموده و جايگزين (Replace) فايل قديمي نماييم . نام اين فايل موقت را با استفاده از متد GetTempFileName بدست مي آوريم.

سپس يک شي از نوع ZipOutputStream ايجاد كرديم كه به اين فايل موقت اشاره مي كند. در اينجا متد CopyEntriesكه قبلا نوشته ايم را فراخوني مي كنيم . اين متد entry ها را از فايلي كه به عنوان پارامتر اول به آن ارسال مي شود، مي گيرد و به داخل يک (ZipOutputStream پارامتر دوم) كپي مي كند.

اگر شما در حال ايجاد يک فايل زيپ جديد مي باشيد ، متد CopyEntries ، هيچ entry را كپي نمي كند ولي اگر در حال اضافه كردن عناصري به فايل زيپي كه در حال حاضر وجود دارد ، هستيد اين متد تمام entry هاي موجود در اين فايل زيپ را به فايل زيپ موقتي جديد كپي خواهد كرد.سپس در حلقه foreach همه فايل هايي كه بايد زيپ شوند به ZipFile اضافه مي شوند.

هر فايل زيپ شده توسط كلاس ZipEntry ، قابل ارائه خواهد بود .سازنده (Constructor) كلاس ZipEntry ، نام entry مربوطه را مي گيرد.

با استفاده از تابع setMethod ، متد فشرده سازي به حالت پيشفرض ، تنظيم مي شود.

ZipEntryاي كه به تازگي ايجاد شده است ، با استفاده از متد putNextEntry به ZipOutputStream اضافه شده است.

يک ZipEntry فقط متاديتاي مربوط به يک entry را ارائه مي دهد و ما هنوز نياز داريم كه محتواي واقعي فايل را به فايل زيپ اضافه كنيم . اين عمل را با استفاده از متد CopyStream كه قبلا نوشته ايم امكان پذير مي باشد.

حذف كردن entry ها از يک فايل زيپ شده:

در مقابل متد AddEntries ، متد RemoveEntries ، اشياء ZipEntry را از يک فايل زيپ شده مشخص حذف مي كند. اين متد در زير نشان داده شده است.

```
private static void RemoveEntries(ZipFile file, string[] items)

{

string fileName = file.getName();

string tempFileName = Path.GetTempFileName();

ZipOutputStream destination = new ZipOutputStream(new
FileOutputStream(tempFileName));

try

{

List<ZipEntry> allItems = GetZippedItems(file);

List<string> filteredItems = new List<string>();

foreach (ZipEntry entry in allItems)

{

bool found = false;

foreach (string s in items)

{
```

```
if (s != entry.getName())

{

found = true;

}

}

if (found)

{

filteredItems.Add(entry.getName());

}

}

CopyEntries(file, destination, filteredItems.ToArray());

}

finally

{

destination.close();

}

file.close();

System.IO.File.Copy(tempFileName, fileName, true);

System.IO.File.Delete(tempFileName);

}
```

متد RemoveEntries یک ZipEntry را به در پارامتر اول گرفته و لیست اسامي entry هايي که بايد از اين فايل حذف شوند را در پارامتر دوم خود مي گيرد.

متد RemoveEntries بسيار شبيه به متد AddEntries مي باشد به استثناي اينکه ديگر ، entryهاي مشخص شده ، کپي نمي شوند.

به کدهايي که به صورت **bold** مي باشند توجه کنيد . در اين قسمت ليستي که حاوي کل entry ها مي باشد و ليستي که حاوي نام entry هايي که بايد حذف شوند ، مقايسه مي شوند. اگر نام entry جزو نام هايي که بايد حذف شوند نباشد ، به ليست filteredItems اضافه مي شود و در غير اين صورت اضافه نمي شود.

در نهايت ليست filteredItems ، حاوي نام entry هايي مي باشد که بايد کپي شوند . در انتها متد Overloads شده دوم CopyEntries را فراخواني مي کنيم تا فقط entry هاي مشخص شده را براي ما کپي کند.

ايجاد يک فايل زيپ جديد:

به منظور ايجاد يک فايل زيپ جديد ، يک متد استاتيک با نام CraeteZipFile در داخل کلاس مي نويسيم . متد CraeteZipFileدو پارامتر مي گيرد . پارامتر اول مسير و نام فايل زيپ مي باشد . و پارامتر دوم آرايه اي از نام هاي عناصري است که قرار است زيپ شوند

```
public static void CreateZipFile(string filename, string[] items)

{

FileOutputStream fout = new FileOutputStream(filename);

ZipOutputStream zout = new ZipOutputStream(fout);

zout.close();

ZipFile zipfile = new ZipFile(filename);

AddEntries(zipfile, items);

}
```

همانطور که ملاحظه مي کنيد يک شي از نوع کلاس FileOutputStream ايجاد نموده ايم . کلاس FileOutputStreamيک استريم قابل نوشتن روي يک فايل را ارائه مي دهد.

سازنده (Constructor) اين کلاس مسير فايلي را که ما قصد داريم استريم را در آن بنويسيم ، دريافت مي کند. سپس يک شي از نوع کلاس ZipOutputStream ايجاد نموده ايم . کلاس ZipOutputStream يک استريم که قابليت نوشتن (writeable) روي يک فايل زيپ را دارد ، ارائه مي کند.

با فراخواني متد close مربوط به کلاس ZipOutputStream يک فايل خالي زيپ به وجود مي آيد. اکنون يک شي از نوع کلاس ZipFile ايجاد کرده ايم. کلاس ZipFile يک فايل زيپ را به ما ارائه مي کند و محتواي فايل زيپ را ايجاد مي کند.

در انتهاي متد AddEntries را که قبلا نوشته ايم ، فراخواني کرده ايم.

اضافه کردن فايل ها به يک فايل زيپ موجود:

مواقعي وجود دارد که نياز داريم ، فايل هايي را به فايل زيپي که هم اکنون موجود است ، اضافه کنيم . متد AddToZipFileدقيقا اين کار را انجام مي دهد.

```
public static void AddToZipFile(string filename, string[] items)

{

ZipFile file = new ZipFile(filename);

AddEntries(file, items);

}
```

اين تابع مسير فايل زيپ شده را به عنوان پارامتر اول و آرايه اي از نامهاي فايل هايي که بايد به اين فايل زيپ اضافه شوند را به عنوان پارامتر دوم مي گيرد . وظيفه باقي کد ها نيز قبلا شرح داده شده است.

حذف کردن فايل هايي از يک فايل زيپ شده:

براي انجام اين کار متدي به نام RemoveFromZipFile نوشته ايم که در زير آن را مي بينيد.

```
public static void RemoveFromZipFile(string filename, string[] items)

{

ZipFile file = new ZipFile(filename);

RemoveEntries(file, items);

}
```

استخراج (Extract) یک فایل زیپ:

تا اینجا شما یاد گرفته اید که چگونه فایل را فشرده سازید (زیپ کنید) و به یک فایل زیپ ، عناصري را اضافه و یا حذف کنید. اکنون فرا خواهید گرفت که چگونه فایل هاي موجود در یک فایل زیپ را استخراج کنید . براي این کار متدي به نام ExtractZipFile مي نویسیم.

```
public static void ExtractZipFile(string zipfilename, string
destination)

{

ZipFile zipfile = new ZipFile(zipfilename);

List<ZipEntry> entries = GetZippedItems(zipfile);

foreach (ZipEntry entry in entries)

{

if (!entry.isDirectory())

{

InputStream s = zipfile.getInputStream(entry);

try

{

string fname = System.IO.Path.GetFileName(entry.getName());

string dir = System.IO.Path.GetDirectoryName(entry.getName());

string newpath = destination + @"\" + dir;

System.IO.Directory.CreateDirectory(newpath);

FileOutputStream dest = new
FileOutputStream(System.IO.Path.Combine(newpath, fname));

try
```

```
{

CopyStream(s, dest);

}

finally

{

dest.close();

}

}

finally

{

s.close();

}

}

}

}
```

متد ExtractZipFile ، مسير فايل زيپ را به عنوان پارامتر اول و مسير پوشه اي كه فايل هاي زيپ شده بايد در آن استخراج شوند را به عنوان پارامتر دوم مي گيرد. يک شي از نوع ZipFile به وجود آورديم كه entries را توسط متد GetZippedFileNamesكه قبلا نوشته ايم ، بدست مي آورد.

سپس در يک حلقه foreach تمام entry ها پيمايش مي شوند . در هر بار پيمايش حلقه foreach ، يک entry مشخص به درون پوشه مورد نظر استخراج مي شود. متد getInputStream از كلاس ZipFile ، يک InputStream براي entry مشخص ، برمي گرداند.

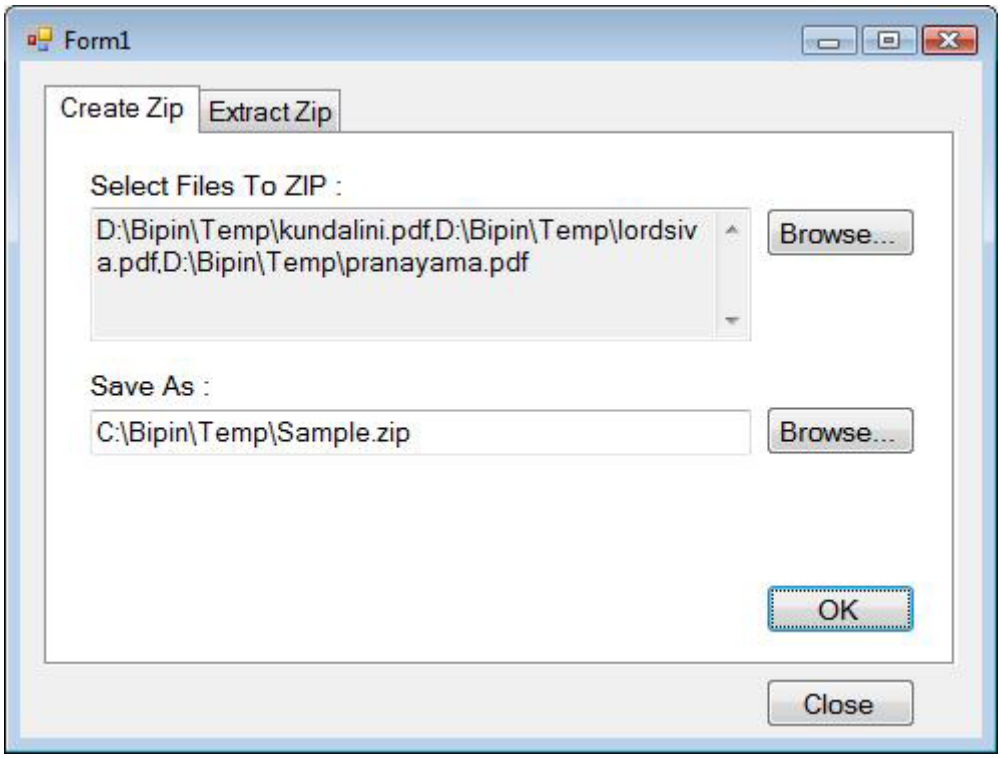
اين استريم به عنوان استريم منبع عمل مي كند. متد getName از كلاس ZipEntry ، نام كامل entry را برمي گرداند. توجه داشته باشيد كه اين نام شامل نام درايو نمي باشد. نام فايل با استفاده از نام entry و مسير پوشه مقصد محاسبه مي شود.

يک FileOutputStream ايجاد كرده ايم تا فايل استخراج شده را روي هارد ديسک بنويسيم. و در نهايت با استفاده از متد CopyStream اطلاعات را از InputStream منبع به FileOutputStream مقصد انتقال مي دهيم.

خوب . تمام شد . كتابخانه كلاس ما كامل شد.

استفاده از كتابخانه كلاس: (Class Library)

استفاده از كتابخانه كلاس نسبتا ساده مي باشد. شما نياز داريد كه متد هاي لازم را از كلاس ZipFileHelper كه امروز نوشتيم ، فراخواني كنيد. براي مثال براي ايجاد يک فايل زيپ بايد متد CreateZipFile را فراخواني كنيد و براي استخراج فايل هاي موجود در يک فايل زيپ بايد متد ExtractZipFile را فراخواني كنيد.


Form1
Create Zip
Extract Zip
Select Files To ZIP :
D:\Bipin\Temp\kundalini.pdf,D:\Bipin\Temp\lordsiva.pdf,D:\Bipin\Temp\pranayama.pdf
Browse...
Save As :
C:\Bipin\Temp\Sample.zip
Browse...
OK
Close