Compressing Streams & Using Isolated Storage

By Pakita Shamoi

Intro

- Two methods for compressing data: **GZIP** and **DEFLATE**.
- Both of these compression methods are industry-standard compression algorithms that are also free of patent protection.
- Limit is 4gb.
- Two corresponding streams are **GZipStream** and **DeflateStream**

Compressing data

- Open the file to be compressed and the file you are going to write to:

```
FileStream sourceFile = File.OpenRead("a.txt");
FileStream destFile = File.Create("azip.txt");
```

- Create GZipStream object and specify the destination stream

```
GZipStream compStream  = new GZipStream(destFile, CompressionMode.Compress);
```

- Then read data from the source stream and feed it into the compression stream

```
int theByte = sourceFile.ReadByte();
while (theByte != -1)
{
    compStream.WriteByte((byte)theByte);
    theByte = sourceFile.ReadByte();
}
```

Decompressing data

- In this case, the source file is a compressed file and the destination file is going to be written as a decompressed file

```
GZipStream decompStream = new GZipStream(sourceFile, CompressionMode.Decompress);
```

- Now you need to read from the decompression stream instead of from the source file and write out to the file directly

```
while (theByte != -1)
{
    Console.WriteLine("ss");
    destFile.WriteByte((byte)theByte);
    theByte = decompStream.ReadByte();
}
```

Isolated storage

- To bridge the needs of applications to save data and the desire of administrators and users to use more limited security settings, the .NET Framework supports the concept of *isolated storage*.
- It would be nice to have a place to store information that was safe to use without having to test whether the application has enough rights to save data to the hard drive.
- By using isolated storage to save your data, you will have access to a safe place to store information without needing to resort to having users grant access to specific files or folders in the file system.

- *The main benefit of using isolated storage is that your application will run regardless of whether it is running under partial, limited, or full-trust.*

IsolatedStorageFile

- Provides the basic functionality to create files and folders in isolated storage.
- Before you can save data in isolated storage, you must determine how to scope the data you want in your store (Assemby/Machine or Assembly/User).
- The store in this case is scoped to the specific user that is executing the assembly

```
IsolatedStorageFile userStorage = IsolatedStorageFile.GetUserStoreForAssembly();
```

IsolatedStorageFileStream

- Encapsulates a stream that is used to create files in isolated storage
- Derives from FileStream

```
IsolatedStorageFileStream userStream = new IsolatedStorageFileStream("UserSettings.set",
FileMode.Create,userStore);
```

- Once you have an instance of the *IsolatedStorageFileStream* class, working with it is identical to working with any file stream

```
StreamWriter userWriter = new StreamWriter(userStream);
userWriter.WriteLine("User Prefs");
userWriter.Close();
```

Reading from your store

```
IsolatedStorageFileStream userStream2 = new IsolatedStorageFileStream("UserSettings.set",
FileMode.Open, userStorage);
StreamReader sr = new StreamReader(userStream2);
Console.WriteLine(sr.ReadLine());
```

GetFileNames()

- Although you can't directly check the file in isolated storage for existence (using Exists method), you can do it using GetFileNames():

```
string[] files = userStorage.GetFileNames("*");
if (files.Length == 0)
{
    Console.WriteLine("No data saved for this user");
}
else
{
    Console.WriteLine(files[0]);
}
```

Using Directories

- You are not limited to storing data as just a set of files in isolated storage; instead, you are also allowed to create directories to store data within.

```
userStorage.CreateDirectory("SomeDir");
IsolatedStorageFileStream userStream = new IsolatedStorageFileStream(@"SomeDir\UserSettings.set",
FileMode.Create, userStorage);
```

- GetDirectoryNames()

Summary

- The compression stream classes (**GZipStream** and **DeflateStream**) can be used to compress or decompress any data up to 4 GB.
- The **IsolatedStorageFile** class can be used to access safe areas to store data for assemblies and users.
- The **IsolatedStorageFileStream** class can be used to read and write data into these safe stores.
- The **IsolatedStorageFileStream** class derives from the **FileStream** class, so any files the class creates can be used like any other file in the file system.