Contents

This walkthrough shows you how to save and restore the EntityManager cache in compressed, encrypted format using chained streams.

- **Language:** C#, VB

Problem

You'd like to save the entity cache to local storage, but you'd also like that file to be compressed, and encrypted for secure access.

Solution

This solution illustrates how to save the contents of an EntityManager cache to local storage, in encrypted form; and how to restore the saved CacheState to an EntityManager.

You can take advantage of Stream chaining to save and restore the *EntityManager* cache in various formats.

Here we'll show a simple example how to create and restore a file of cached entities in compressed, encrypted format.

We use three Streams,

- A FileStream for writing and reading to a file;
- A CryptoStream to provide encryption;
- A GZipStream to provide compression.

Note that the standard *GZipStream* class is not available in Silverlight, but you can instead use the GZipStream class provided with DevForce.

Sample

Here we define a simple class of extensions to the *EntityManager*. Defining extension methods is not required, it just makes calling the methods a little tidier.

The save and restore methods both accept a hash key to be used during encryption/decryption and the name of the file to be saved or restored. Both methods use the AesManaged provider, which is available to both WinClient and Silverlight applications.

The save method chains the three streams to save cache contents to a file: cache state => compressed with *GZipStream* => encrypted with *CryptoStream* => written to a file with *FileStream*.

The restore method restores the file into cache: file read with *FileStream* => decrypted with *CryptoStream* => decompressed with *GZipStream* => cache loaded.

Remember that you must use the same key during both save and restore.

```
public static class EncryptoExtensions {
 /// <summary>
 /// Save cache state to a file in compressed, encrypted format.
 /// </summary>
 public static void SaveCompressedEncryptedFile(this EntityManager em, byte[] key, string fileName) {
    var provider = new AesManaged { Key = key, IV = key };
   using (FileStream fs = new FileStream(fileName, FileMode.Create)) {
     using (CryptoStream cs = new CryptoStream(fs, provider.CreateEncryptor(), CryptoStreamMode.Write)) {
       using (GZipStream gz = new GZipStream(cs, CompressionMode.Compress)) {
         em.CacheStateManager.SaveCacheState(gz);
       }
     }
    }
  }
 /// <summary>
 /// Restore cache state from a file in compressed, encrypted format.
 /// </summary>
 public static void RestoreCompressedEncryptedFile(this EntityManager em, byte[] key, string fileName) {
    var provider = new AesManaged { Key = key, IV = key };
   using (FileStream fs = new FileStream(fileName, FileMode.Open)) {
```

```
    using (CryptoStream cs = new CryptoStream(fs, provider.CreateDecryptor(), CryptoStreamMode.Read)) {
      using (GZipStream gz = new GZipStream(cs, CompressionMode.Decompress)) {
        em.CacheStateManager.RestoreCacheState(gz, RestoreStrategy.Normal);
      }
    }
  }
 }
}
```

```
Public Module EncryptoExtensions
''' <summary>
''' Save cache state to a file in compressed, encrypted format.
''' </summary>
<System.Runtime.CompilerServices.Extension> _
Public Sub SaveCompressedEncryptedFile(ByVal em As EntityManager, ByVal key() As Byte, ByVal fileName As String)
Dim provider = New AesManaged With {.Key = key, .IV = key}
  Using fs As New FileStream(fileName, FileMode.Create)
Using cs As New CryptoStream(fs, provider.CreateEncryptor(), CryptoStreamMode.Write)
  Using gz As New GZipStream(cs, CompressionMode.Compress)
 em.CacheStateManager.SaveCacheState(gz)
  End Using
End Using
  End Using
End Sub
''' <summary>
''' Restore cache state from a file in compressed, encrypted format.
''' </summary>
<System.Runtime.CompilerServices.Extension> _
Public Sub RestoreCompressedEncryptedFile(ByVal em As EntityManager, ByVal key() As Byte, ByVal fileName As String)
Dim provider = New AesManaged With {.Key = key, .IV = key}
  Using fs As New FileStream(fileName, FileMode.Open)
Using cs As New CryptoStream(fs, provider.CreateDecryptor(), CryptoStreamMode.Read)
  Using gz As New GZipStream(cs, CompressionMode.Decompress)
 em.CacheStateManager.RestoreCacheState(gz, RestoreStrategy.Normal)
  End Using
End Using
  End Using
End Sub
End Module
```

It's simple to call these methods whenever we need to save or restore the *EntityManager* cache.

Here we save the cache contents to a file.

```
// Assume an EM loaded with customers, their orders and order details.
var em = new DomainModelEntityManager();
em.Customers.Include("Orders.OrderDetails").ToList();
string fileName = @"c:\temp\cachefile.bin";
var key = CryptoFns.AesGetHashKey("akey4me");
em.SaveCompressedEncryptedFile(key, fileName);
```

```
Dim em = New DomainModelEntityManager()
em.Customers.Include("Orders.OrderDetails").ToList()
Dim fileName As String = "c:\temp\cachefile.bin"
Dim key = CryptoFns.AesGetHashKey("akey4me")
em.SaveCompressedEncryptedFile(key, fileName)
```

And later, we'll restore the file:

```
var em = new DomainModelEntityManager();
string fileName = @"c:\temp\cachefile.bin";
var key = CryptoFns.AesGetHashKey("akey4me");

em.RestoreCompressedEncryptedFile(key, fileName);
```

```
Dim em = New DomainModelEntityManager()
Dim fileName As String = "c:\temp\cachefile.bin"
Dim key = CryptoFns.AesGetHashKey("akey4me")
em.RestoreCompressedEncryptedFile(key, fileName)
```

The *CryptoFns.AesGetHashKey* method is a helper method defined in *IdeaBlade.Core.CryptoFns*.