Contents

Add system utility classes like **SystemTime** and **SystemGuid** to your application so you can test component behaviors that are otherwise sensitive to unpredictable values coming from the system environment (e.g., the current time or a new Guid).

The problem

.NET environment functions typically return values that come from the operating system such as DateTime.Now and Guid.NewGuid(). These values are different everytime which complicates repeatable testing.

You can control these values if you replace calls to the .NET functions with calls to similar functions in system utility classes that you wrote. These utility classes behave like the .NET functions under normal conditions but you can change their behavior during testing.

SystemTime

DateTime is especially problematic for testers who want to validate logic that is time sensitive. Suppose you had business logic that expected something to happen three days from now. You certainly don't want a three-day test.

You can write a **SystemTime** class such as the one included below. *SystemTime* has *Now()* and *Today()* methods that return a *DateTime*. If you do nothing, they behave exactly like *DateTime.Now* and *DateTime.Today*.

Now everywhere in your application you always call *SystemTime.Now()* and *SystemTime.Today()* wherever you would have called *DateTime.Now* and *DateTime.Today*.

If you do nothing, the behavior of your application is unchanged.

During a test, however, you can replace the *SystemTime.NowFn* with a test-time function that returns any *DateTime* value you need.

Remember when you are done to restore *NowFn* to the proper default behavior. A "ResetNowFn" method makes that easier for testers.

```
/// <summary>
/// Replacement for <see cref="DateTime"/>
/// ** ALWAYS USE THIS INSTEAD OF CALLING DATETIME DIRECTLY
/// </summary>
/// <remarks>
/// Covers the <see cref="DateTime"/> static methods so that
/// test and development scenarios can reset the "Current" time
/// to whatever they like. See <see cref="NowFn"/>.
/// TODO: Extend to support UtcNow
/// </remarks>
public static class SystemTime {
  /// <summary>
  /// Gets a <see cref="DateTime"/> that is set to the "current" date and time
  /// in the manner of <see cref="DateTime.Now"/>.
  /// See <see cref="NowFn"/>.
  /// </summary>
  public static DateTime Now() { return NowFn(); }
  /// <summary>
  /// Gets "current" date
  /// in the manner of <see cref="DateTime.Today"/>.
  /// See <see cref="NowFn"/>.
  /// </summary>
  public static DateTime Today() {
     var now = Now();
   return now-now.TimeOfDay;
   }
  /// <summary>
  /// Function for getting the "current" date and time.
  /// Defaults to <see cref="DateTime.Now"/> but
  /// could be replaced in a test or development scenario.
  /// </summary>
  public static Func<DateTime> NowFn = () => DateTime.Now;
```

```
  /// <summary>
  /// Reset <see cref="NowFn"/> to application default
  /// </summary>
  public static void ResetNowFn() { NowFn  = () => DateTime.Now; }
 }
```

```
''' <summary>
''' Replacement for <see cref="DateTime"/>
''' ** ALWAYS USE THIS INSTEAD OF CALLING DATETIME DIRECTLY
''' </summary>
''' <remarks>
''' Covers the <see cref="DateTime"/> static methods so that
''' test and development scenarios can reset the "Current" time
''' to whatever they like. See <see cref="NowFn"/>.
''' TODO: Extend to support UtcNow
''' </remarks>
Public NotInheritable Class SystemTime
  ''' <summary>
  ''' Gets a <see cref="DateTime"/> that is set to the "current" date and time
  ''' in the manner of <see cref="DateTime.Now"/>.
  ''' See <see cref="NowFn"/>.
  ''' </summary>
  Private Sub New()
  End Sub
  Public Shared Function Now() As Date
  Return NowFn()
  End Function
  ''' <summary>
  ''' Gets "current" date
  ''' in the manner of <see cref="DateTime.Today"/>.
  ''' See <see cref="NowFn"/>.
  ''' </summary>
  Public Shared Function Today() As Date
 Dim now = SystemTime.Now()
Return now-now.TimeOfDay
  End Function
  ''' <summary>
  ''' Function for getting the "current" date and time.
  ''' Defaults to <see cref="DateTime.Now"/> but
  ''' could be replaced in a test or development scenario.
  ''' </summary>
  Public Shared NowFn As Func(Of Date) = Function() Date.Now
  ''' <summary>
  ''' Reset <see cref="NowFn"/> to application default
  ''' </summary>
  Public Shared Sub ResetNowFn()
  NowFn = Function() Date.Now
  End Sub
End Class
```

SystemGuid

Guid.NewGuid() returns a new and deliberately unpredictable value every time. You may find yourself writing a test where you would like to control the value of that new Guid.

You can write a **SystemGuid** class such as the one included below. *SystemGuid* has a *NewGuid()* method that return a *Guid*. If you do nothing, it behaves exactly like *Guid.NewGuid()*.

Now everywhere in your application you always call *SystemGuid.NewGuid()* wherever you would have called *Guid.NewGuid*.

If you do nothing, the behavior of your application is unchanged.

During a test, however, you can replace the *SystemGuid.NewGuidFn* with a test-time function that returns any *Guid* value you need.

Remember when you are done to restore *NewGuidFn* to the proper default behavior. A "ResetNewGuidFn" method makes that easier for testers.

```
 public static class SystemGuid
  {
    /// <summary>
```

```
    /// Gets a new <see cref="Guid"/>
    /// in the manner of <see cref="Guid.NewGuid"/>.
    /// See <see cref="NewGuidFn"/>.
    /// </summary>
    public static Guid NewGuid() { return NewGuidFn(); }
    /// <summary>
    /// Function for getting a new <see cref="Guid"/>.
    /// Defaults to <see cref="Guid.NewGuid"/> but
    /// could be replaced in a test or development scenario.
    /// </summary>
    public static Func<Guid> NewGuidFn = Guid.NewGuid;
    /// <summary>
    /// Reset <see cref="NewGuidFn"/> to application default
    /// </summary>
    public static void ResetNewGuidFn() { NewGuidFn = Guid.NewGuid; }
  }
```

```
Public NotInheritable Class SystemGuid
  ''' <summary>
  ''' Gets a new <see cref="Guid"/>
  ''' in the manner of <see cref="Guid.NewGuid"/>.
  ''' See <see cref="NewGuidFn"/>.
  ''' </summary>
  Private Sub New()
  End Sub
  Public Shared Function NewGuid() As Guid
  Return NewGuidFn()
  End Function
  ''' <summary>
  ''' Function for getting a new <see cref="Guid"/>.
  ''' Defaults to <see cref="Guid.NewGuid"/> but
  ''' could be replaced in a test or development scenario.
  ''' </summary>
  Public Shared NewGuidFn As Func(Of Guid) = Guid.NewGuid
  ''' <summary>
  ''' Reset <see cref="NewGuidFn"/> to application default
  ''' </summary>
  Public Shared Sub ResetNewGuidFn()
  NewGuidFn = Guid.NewGuid
  End Sub
End Class
```

Make your own system utility classes

Look for other places in your application where you call a .NET static function or property and consider following the pattern you see here.