عنوان: **آشنایی با آزمایش واحد (unit testing) در دات نت، قسمت 5**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۱۸ ۱۳:۱۰:۳۸
آدرس: www.dotnettips.info
برچسب‌ها: Unit testing

ادامه آشنایی با NUnit

حالت‌های مختلف Assert :

NUnit framework حالت‌های مختلفی از دستور Assert را پشتیبانی می‌کند که در ادامه با آنها آشنا خواهیم شد.

کلاس Assertion :
این کلاس دارای متدهای زیر است:

```
public static void Assert(bool condition)
public static void Assert(string message, bool condition)
```

تنها در حالتی این بررسی موفقیت آمیز گزارش خواهد شد که condition مساوی true باشد

```
public static void AssertEquals(string message, object expected, object actual)
public static void AssertEquals(string message, float expected, float actual, float delta)
public static void AssertEquals(string message, double expected, double actual, double delta)
public static void AssertEquals(string message, int expected, int actual)
public static void AssertEquals(int expected, int actual)
public static void AssertEquals(object expected, object actual)
public static void AssertEquals(float expected, float actual, float delta)
public static void AssertEquals(double expected, double actual, double delta)
```

تنها در صورتی این بررسی به اثبات خواهد رسید که اشیاء actual و expected یکسان باشند. (دلتا در اینجا به عنوان تلرانس آزمایش درنظر گرفته می‌شود)

```
public static void AssertNotNull(string message, object anObject)
public static void AssertNotNull(object anObject)
```

این بررسی تنها در صورتی موفقیت آمیز گزارش می‌شود که شیء مورد نظر نال نباشد.

```
public static void AssertNull(string message, object anObject)
public static void AssertNull(object anObject)
```

این بررسی تنها در صورتی موفقیت آمیز گزارش می‌شود که شیء مورد نظر نال باشد.

```
public static void AssertSame(string message, object expected, object actual)
public static void AssertSame(object expected, object actual)
```

تنها در صورتی این بررسی به اثبات خواهد رسید که اشیاء actual و expected یکسان باشند.

```
public static void Fail(string message)
public static void Fail()
```

همواره Fail خواهد شد. (در مورد کاربرد آن در قسمت بعد توضیح داده خواهد شد)

نکته:
در یک متد آزمایش واحد شما مجازید به هرتعدادی که لازم است از متدهای Assertion استفاده نمائید. در این حالت اگر تنها یکی از متدهای assertion با شکست روبرو شود، کل متد آزمایش واحد شما مردود گزارش شده و همچنین عبارات بعدی Assertion بررسی نخواهند شد. بنابراین توصیه می‌شود به ازای هر متد آزمایش واحد، تنها از یک Assertion استفاده نمائید.

مهم!
کلاس Assertion منسوخ شده است و توصیه می‌شود بجای آن از کلاس Assert استفاده گردد.

آشنایی با کلاس Assert :
این کلاس از متدهای زیر تشکیل شده است:

الف) بررسی حالت‌های تساوی

```
Assert.AreEqual( object expected, object actual );
```

جهت بررسی تساوی دو شیء مورد بررسی و شیء مورد انتظار بکار می‌رود.

```
Assert.AreNotEqual( object expected, object actual );
```

جهت بررسی عدم تساوی دو شیء مورد بررسی و شیء مورد انتظار بکار می‌رود.
برای مشاهده انواع و اقسام overload های آن‌ها می‌توانید به راهنمای NUnit که پس از نصب، در پوشه doc آن قرار می‌گیرد مراجعه نمائید.

همچنین دو متد زیر و انواع overload های آن‌ها جهت برسی اختصاصی حالت تساوی دو شیء بکار می‌روند:

```
Assert.AreSame( object expected, object actual );
Assert.AreNotSame( object expected, object actual );
```

بعلاوه اگر نیاز بود بررسی کنیم که آیا شیء مورد نظر حاوی یک آرایه یا لیست بخصوصی است می‌توان از متد زیر و oveload های آن استفاده نمود:

```
Assert.Contains( object anObject, IList collection );
```

ب) بررسی حالت‌های شرطی:

```
Assert.IsTrue( bool condition );
```

تنها در حالتی این بررسی موفقیت آمیز گزارش خواهد شد که condition مساوی true باشد

```
Assert.IsFalse( bool condition);
```

تنها در حالتی این بررسی موفقیت آمیز گزارش خواهد شد که condition مساوی false باشد

```
Assert.IsNull( object anObject );
```

این بررسی تنها در صورتی موفقیت آمیز گزارش می‌شود که شیء مورد نظر نال باشد.

```
Assert.IsNotNull( object anObject );
```

این بررسی تنها در صورتی موفقیت آمیز گزارش می‌شود که شیء مورد نظر نال نباشد.

```
Assert.IsNaN( double aDouble );
```

این بررسی تنها در صورتی موفقیت آمیز گزارش می‌شود که شیء مورد نظر عددی نباشد (اگر با جاوا اسکریپت کار کرده باشید حتما با isNan آشنا هستید، is not a numeric).

```
Assert.IsEmpty( string aString );
Assert.IsEmpty( ICollection collection );
```

جهت بررسی خالی بودن یک رشته یا لیست بکار می‌رود.

```
Assert.IsNotEmpty( string aString );
Assert.IsNotEmpty( ICollection collection );
```

جهت بررسی خالی نبودن یک رشته یا لیست بکار می‌رود.

ج) بررسی حالت‌های مقایسه‌ای

```
Assert.Greater( double arg1, double arg2 );
Assert.GreaterOrEqual( int arg1, int arg2 );
Assert.Less( int arg1, int arg2 );
Assert.LessOrEqual( int arg1, int arg2 );
```

نکته‌ای را که در اینجا باید در نظر داشت این است که همواره شیء اول با شیء دوم مقایسه می‌شود. مثلا در حالت اول، اگر شیء اول بزرگتر از شیء دوم بود، آزمایش مورد نظر با موفقیت گزارش خواهد شد.
از ذکر انواع و اقسام overload های این توابع جهت طولانی نشدن مطلب پرهیز شد.

د) بررسی نوع اشیاء

```
Assert.IsInstanceOfType( Type expected, object actual );
Assert.IsNotInstanceOfType( Type expected, object actual );
Assert.IsAssignableFrom( Type expected, object actual );
Assert.IsNotAssignableFrom( Type expected, object actual );
```

این توابع و Overload های آن‌ها امکان بررسی نوع شیء مورد نظر را میسر می‌سازند.

ه) متدهای کمکی

```
Assert.Fail();
Assert.Ignore();
```

در حالت استفاده از ignore ، آزمایش واحد شما در حین اجرا ندید گرفته خواهد شد. از متد fail برای طراحی یک متد assertion سفارشی می‌توان استفاده کرد. برای مثال:
طراحی متدی که بررسی کند آیا یک رشته مورد نظر حاوی عبارتی خاص می‌باشد یا خیر:

```
public void AssertStringContains( string expected, string actual,

string message )

{

if ( actual.IndexOf( expected )

Assert.Fail( message );

}
```

و) متدهای ویژه‌ی بررسی رشته‌ها

```
StringAssert.Contains( string expected, string actual );
StringAssert.StartsWith( string expected, string actual );
StringAssert.EndsWith( string expected, string actual );
StringAssert.AreEqualIgnoringCase( string expected, string actual );
StringAssert.IsMatch( string expected, string actual );
```

این متدها و انواع overload های آن‌ها جهت بررسی‌های ویژه رشته‌ها بکار می‌روند. برای مثال آیا رشته مورد نظر حاوی عبارتی خاص است؟ آیا با عبارتی خاص شروع می‌شود یا با عبارتی ویژه، پایان می‌پذیرد و امثال آن.

ز) بررسی فایل‌ها

```
FileAssert.AreEqual( Stream expected, Stream actual );
FileAssert.AreEqual( FileInfo expected, FileInfo actual );
FileAssert.AreEqual( string expected, string actual );

FileAssert.AreNotEqual( Stream expected, Stream actual );
```

```
FileAssert.AreNotEqual( FileInfo expected, FileInfo actual );
FileAssert.AreNotEqual( string expected, string actual );
```

این متدها جهت مقایسه دو فایل بکار می‌روند و ورودی‌های آن‌ها می‌تواند از نوع stream ، شیء FileInfo و یا مسیر فایل‌ها باشد.

ح) بررسی collections

```
CollectionAssert.AllItemsAreInstancesOfType( IEnumerable collection, Type expectedType );
CollectionAssert.AllItemsAreNotNull( IEnumerable collection );
CollectionAssert.AllItemsAreUnique( IEnumerable collection );
CollectionAssert.AreEqual( IEnumerable expected, IEnumerable actual );
CollectionAssert.AreEquivalent( IEnumerable expected, IEnumerable actual);
CollectionAssert.AreNotEqual( IEnumerable expected, IEnumerable actual );
CollectionAssert.AreNotEquivalent( IEnumerable expected,IEnumerable actual );
CollectionAssert.Contains( IEnumerable expected, object actual );
CollectionAssert.DoesNotContain( IEnumerable expected, object actual );
CollectionAssert.IsSubsetOf( IEnumerable subset, IEnumerable superset );
CollectionAssert.IsNotSubsetOf( IEnumerable subset, IEnumerable superset);
CollectionAssert.IsEmpty( IEnumerable collection );
CollectionAssert.IsNotEmpty( IEnumerable collection );
```

به صورت اختصاصی و ویژه نیز می‌توان بررسی مقایسه‌ای را بر روی اشیایی از نوع IEnumerable انجام داد. برای مثال آیا معادل هستند، آیا شیء مورد نظر نال نیست و امثال آن.

نکته: در تمامی overload های این توابع، آرگومان message نیز وجود دارد. از این آرگومان زمانیکه آزمایش با شکست مواجه شد، جهت ارائه اطلاعات بیشتری استفاده می‌گردد.

ادامه دارد...