NUnit vs xUnit.net

Nunit został wprowadzony w roku 2002 i nadal jest szeroko stosowany, bardzo dobrze udokumentowany i posiada duże community użytkowników, gdzie xUnit.net jest bardziej nowoczesny, bardziej rozszczerzalny i zyskuje na popularności w .NET Core.

Główna różnica polega na sposobie w jaki xUnit.net testuje metody.

W NUnit mamy klasę testową i zbiór metod testowych w niej. NUnit tworzy nową instancję klasy testowej, a następnie uruchamia wszystkie metody testowe w tym samym momencie.

Gdzie xUnit.net tworzy nową instancję klasy testowej dla każdej z metod testowych. W związku z tym nie można używać pól ani właściwości do udostępniania danych między metodami testowymi, co jest złą praktyką, ponieważ metody testowe byłyby od siebie zależne, co jest niedopuszczalne w TDD (Test Driven Development). Dlatego używając Xunit.net trzeba być pewnym, że wszystkie metody są kompletnie odizolowane. Jednakże, jezeli chcemy udostępnić jakieś dane miedzy naszymi metodami testowymi to xUnit nam na to zezwoli.

JUnit – W JUnit, test jednostkowy, który pozwala sprawdzic czy dana funkcjonalność działa w zamierzony sposób. Używanie JUnitów znacznie przyśpiesza pracę nad projektem. JUnity powinno się dzielić na wybrane testy metody lub funkcjonalności, idąc od najprostszych do bardziej zaawansowanych. JUnit jest przykładem architektury xUnity dla frameworków testów jednostkowych.

Junit vs xUnit.net

Porównanie:

	Junit	xUnit.net
Język programowania	Java	.NET
Strona klienta	JUnit umożliwia testowanie front-endowych kompontentów takich jak indywidualne klasy oraz funkcje, które wchodzą w skład front-endu.	Możliwość niezależnego testowania równych komponetnów front-endowych, ponieważ jest to framework do testów jednostowych.
Strona serwera	Możliwość testów klas oraz funkcji, które składają się na back-end takie jak np. połączenia bazy danych.	Możliwość niezależnych testowów różnych komponentów back-endu, ponieważ jest to framework do testów jednostkowych.
Stany danych (Fixtures)	JUnit zawiera metodę setUp(), która jest uruchamiana przed każdym wywołaniem testu, oraz metodę tearDown(), która jest uruchamiana po każdej metodzie testowej.	Posiada klasy, które są konfirurowane po przetestowaniu klasy.
Grupowanie stanów danych (Group fixtures)	Możliwość używania wbudowanych funkcji setUp() i tearDown() jako grupowania określonych stanów danych.	xUnit.net zawiera narzędzia do zbierania danych, które pozwalają na współdzielenie kontekstu między wieloma testami.

AssertJ powstał jako opensource'owa biblioteka, dzięki której możemy pisać w uproszczony sposób. AssertJ znaczenie upraszcza weryfikację wyjątków nawet w porównaniu z JUnit 5. AssertJ zdecydowanie wygrywa czytelniejszym API, które ułatwia zapoznawanie się z napisanymi testami. Obie te biblioteki czyli AssertJ i JUnit 5 możemy łaczyć, aby użyskać ciekawy rezultat. Metoda statyczna assertAll pochodzi z JUnit, natomiast dobrze znany assertThat z AssertJ. Jak widać można wyciągnąć z każdej biblioteki co najlepsze i połączyć to w czytelne testy jednostkowe.