embarcadero®

embarcadero®

Importância em criar apps com testes automatizados/unitários nos dias atuais

Juliomar Marchetti
Embarcadero MVP/Gerente ControlSoft

Embarcadero Conference

Unit Test em Delphi com DUnitX, Delphi Mocks e TestInsight

Software deve-se começar sempre pelos seus testes, se souber o resultado que precisa alcançar será mais fácil de escrever seu código para atender.

Quando falamos de software, falamos de diferentes tipos de testes possíveis de serem aplicados, em diferentes momentos do ciclo de vida do desenvolvimento. Temos os testes de integração, os testes de aceitação, testes de interface, entre outros. Um deles é o teste unitário.

Teste Unitário

Vantagens do teste unitário

- O quanto antes, melhor!
- Facilidade em encontrar falhas e eliminá-las de forma rápida.
- Cobrir todas as possibilidades do seu código (code coverage), para garantir que tudo funcione adequadamente.
- Parte da documentação do seu código-fonte.
- Economia de tempo (e dinheiro!) .
- Isolar o seu código de forma que você tenha certeza de que ele funcione corretamente.

Teste Unitário

Como escrever bons testes?

- Siga o princípio AAA – Arrange, Act e Assert.
- Apenas uma checagem por teste.
- Não utilize lógica em seus testes.
- Os testes precisam ser fáceis de serem executados.
- Faça os testes com os valores esperados primeiro, e depois os que tentam “encontrar” o problema do código.
- Se for possível, realize todos os testes possíveis. Teste todas as possibilidades do seu código.

Teste Unitário

Como escrever bons testes?

- Todos os testes precisam ser independentes.
- Todos os “exceptions” precisam ser testados.
- Cria nomes claros e elucidativos para os seus testes.
- Rode seus testes constantemente.
- Apenas crie os testes necessários.

DUnitX (Test Framework)

Com o Delphi 2010 e a evolução da RTTI do Delphi, foi possível evoluir os frameworks existentes. Assim, recomenda-se hoje a utilização do framework DUnitX ao invés do próprio DUnit, que já vinha com o Delphi de longas datas. As últimas edições do Rad Studio já vem com o DUnitX, mas para as outras versões, é necessário realizar o download das bibliotecas e realizar a instalação.

DUnitX (Test Framework)

DUnitX utiliza os recursos de atributos do Delphi inseridos na versão Delphi 2010, e os principais são:

- **[TestFixture]** – Define que a classe é uma classe de testes. O framework utiliza essa marcação para identificar o que utilizar para os testes.
- **[Setup]** – Define o método responsável pela configuração de tudo o que é necessário para o funcionamento do teste. Lembra do princípio AAA? Esse é o responsável pelo "Arrange", ou seja, preparar todas as condições necessárias para a execução do teste. É a primeira coisa a ser executada dentro do teste.
- **[TearDown]** – É o contrário do Setup, ou seja, desfaz tudo o que foi criado para os testes. É a última coisa a ser executada dentro do teste.
- **[Test]** – Define a procedure que testará a funcionalidade.
- **[TestCase('TestA','1,2')]** – Define o caso de teste.

DUnitX (Test Framework)

- Exemplo

DUnitX (Test Framework)

Veja que ele segue os princípios AAA:

- **Arrange** – preparar todas as condições necessárias para a execução do teste no método de setup.
- **Act** – executar o que será testado em “Aux := FTratamentoStrings.SomenteNumeros(‘A0-2,45%4$38&¨8689.3’);“.
- **Assert** – validar as informações depois do teste em “Assert.IsTrue(Aux = ‘024543886893’, ‘String de entrada A0-2,45%4$38&¨8689.3 deveria retornar 024543886893 mas retornou ‘ + Aux);“.

DUnitX (Test Framework)

Existem ainda diversos outro tipos de verificações (contidas na unit DunitX.Assert) que podem ser feitos:

- Pass
- Fail
- FailFmtNotImplemented
- AreEqual
- AreEqualMemory
- AreNotEqual
- AreNotEqualMemory
- AreSame
- AreNotSame
- Contains
- DoesNotContain
- Implements
- IsTrue
- IsFalse
- IsNull
- IsNotNull
- IsEmpty
- IsNotEmpty
- WillRaise
- WillRaiseWithMessage
- WillRaiseDescendant
- WillRaiseAny
- WillNotRaise
- WillNotRaiseDescendant
- WillNotRaiseAny
- StartsWith
- EndsWith
- InheritsFrom
- IsType
- IsMatch

Delphi Mock (Isolation Framework)

Delphi Mock é um isolation framework para Delphi, ou seja, uma biblioteca de funcionalidades que vão te ajudar a isolar uma unidade. Uma unidade somente poderá ser testada “unitariamente” se estiver isolada de todo o resto. E se, tendo uma classe que possui uma dependência com outro objeto do sistema, eu quisesse realizar um teste unitário? Teria que primeiramente isolá-la da própria dependência, ou seja, do próprio objeto de que ela depende.

Delphi Mock (Isolation Framework)

Existem dois desses tipos de classes:

- **Stub** – não possui funcionalidade nenhuma. Apenas existe como elemento da dependência.
- **Mock** – Possui funcionalidade assim como existe como elemento da dependência. Pode ser utilizado lógica para tratar o retorno da dependência.

Delphi Mock (Isolation Framework)

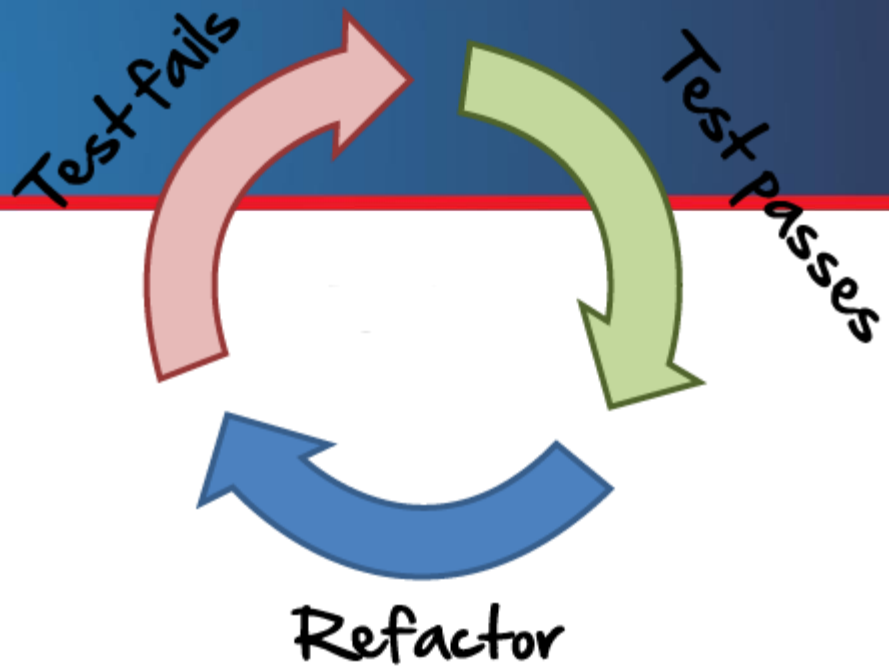
exemplo

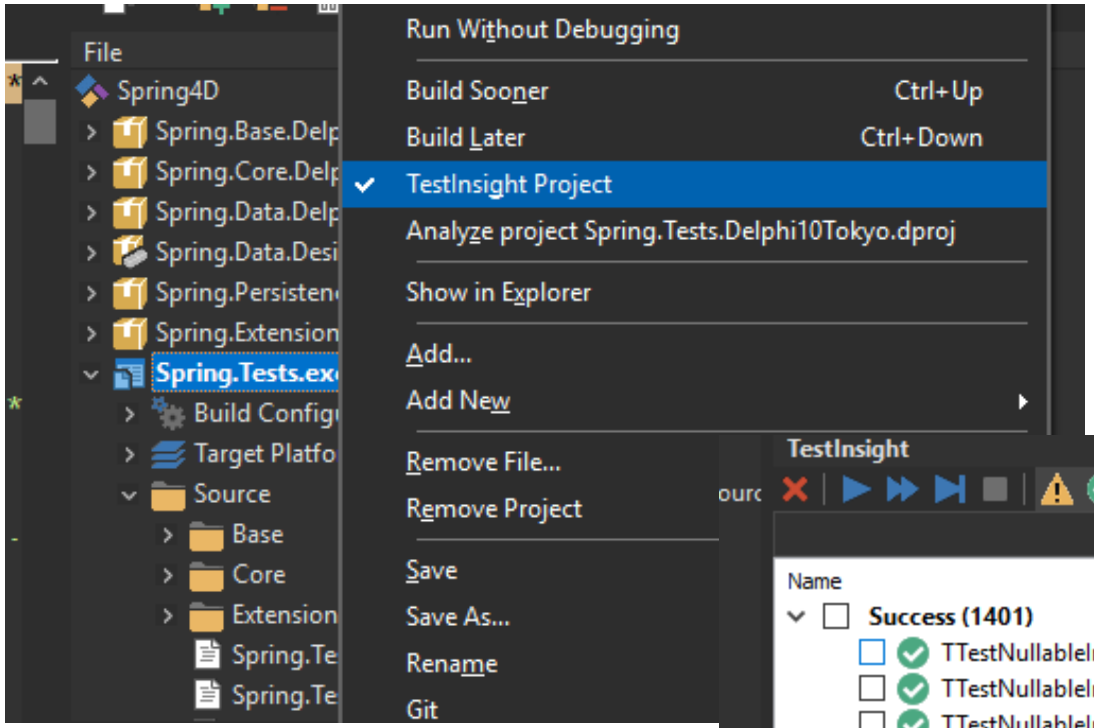
Test Insight

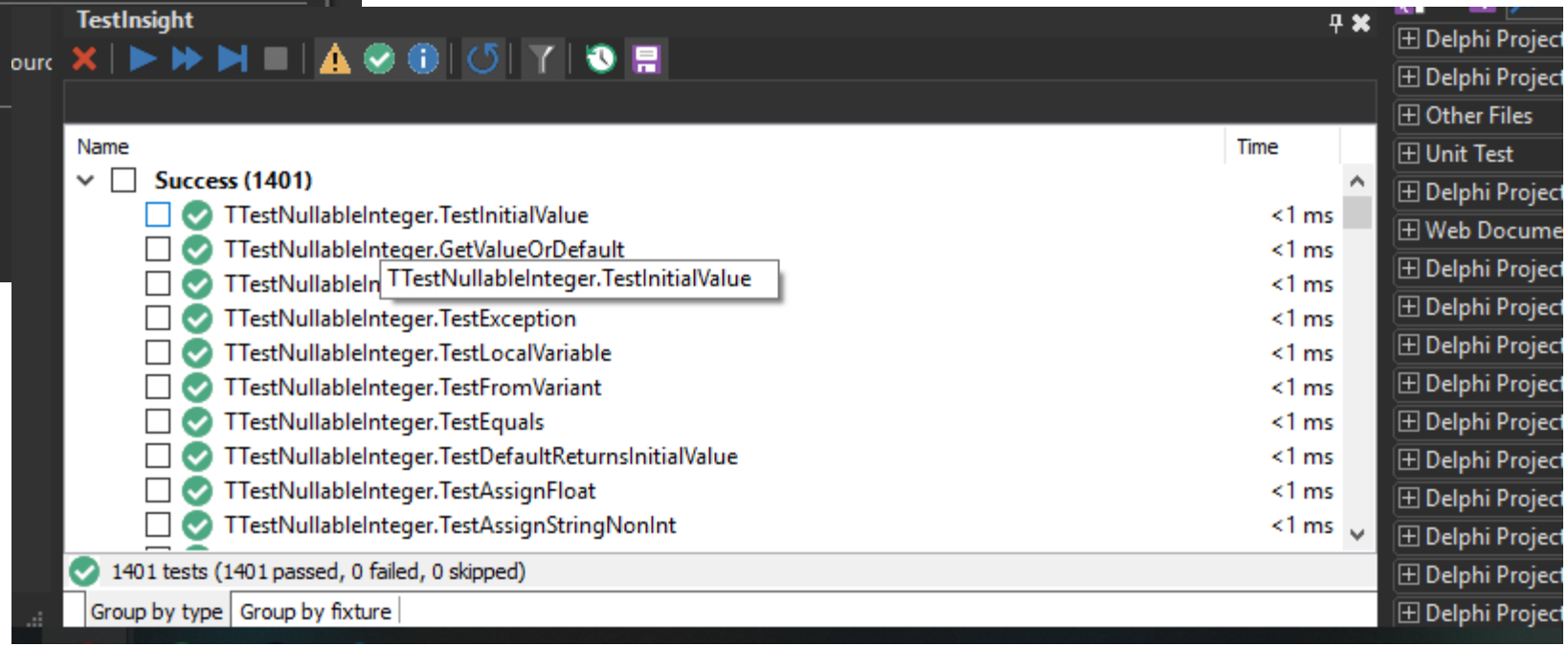
Quanto trabalhamos com o DUnitX, os resultados através do console. O antigo DUnit falta uma visualização gráfica dos resultados. A equipe do DUnitX já havia se programado para o desenvolvimento dessa ferramenta, mas optou por abandonar o projeto em virtude do TestInsight, que já realizou o trabalho muito bem.

O TestInsight é uma ferramenta visual para apuração dos resultados dos testes unitários do Delphi, realizados pelo DUnitX, DUnit e DUnit2.

Test Insight

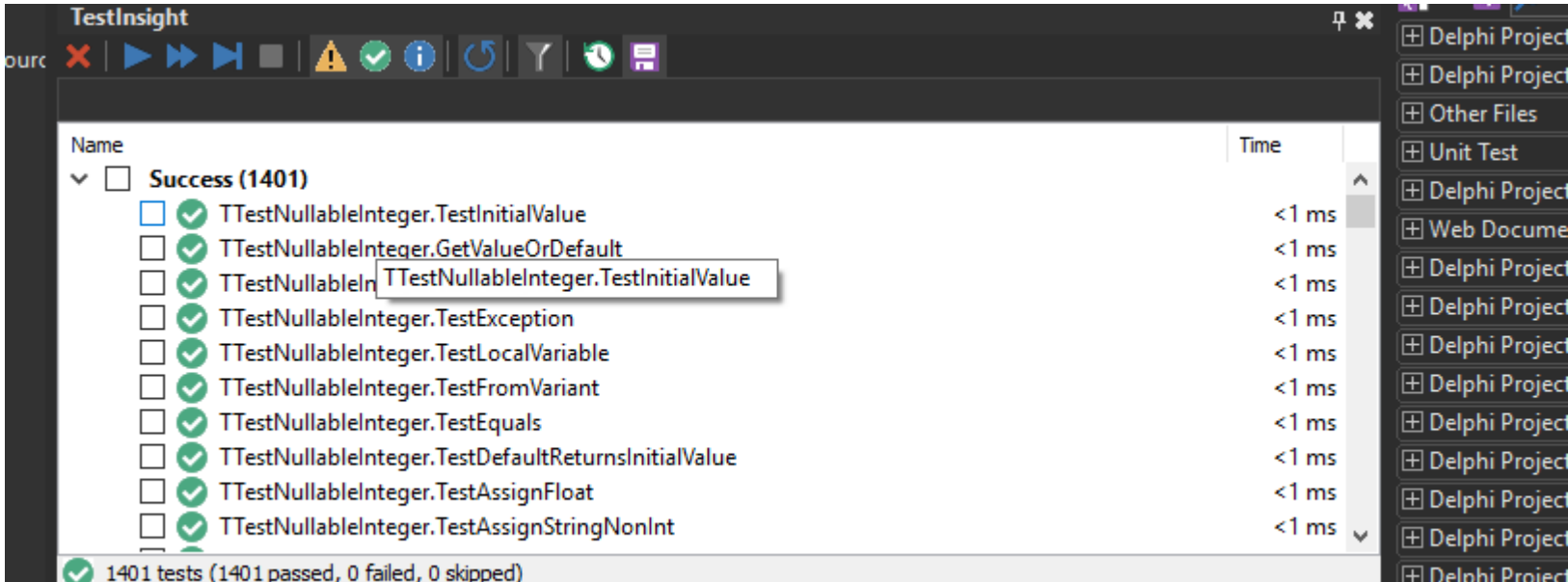






Test Insight

Além do visual, existem funcionalidades ótimas realizadas por esse plug-in. Quando você dá duplo-click em um teste, ele te redireciona automaticamente para o código-fonte em questão. Você pode visualizar os testes por tipo de resultado, conforme a imagem acima, ou por teste, conforme a imagem abaixo:



Test Insight

Na barra superior, o test Insight possui as opções de:

- Limpar os resultados
- Rodar todos os testes
- Rodar apenas os testes selecionados
- Rodas apenas os testes posteriores ao cursor no editor de texto do Delphi
- Terminar os testes em execução
- Mostrar ou ocultar os testes que geraram avisos
- Mostrar ou ocultar os testes que obtiveram sucesso
- Mostrar ou ocultar os testes que não foram executados
- Mostrar uma barra de progresso das execuções dos testes
- Filtrar os resultados dos testes
- Rodar os testes continuamente quando detectar uma ociosidade
- Rodar os testes quando o projeto for salvo.

Test Insight

Interessante ressaltar as duas últimas funcionalidades. Como visto na parte teórica sobre os testes, rodar os testes sempre que possível é altamente recomendável para detectar qualquer problema no código, e o CodeInsight nos dá a opção de fazer isso sempre que salvamos o projeto e sempre que ele detecta uma ociosidade.

Test Insight

Exemplo

Sites e Blogs com conteúdo

http://edgarpavao.com/2017/07/11/unit-test-em-delphi-com-dunitx-delphi-mocks-e-testinsight/

DUnitX (Test Framework)

https://github.com/VSoftTechnologies/DUnitX

Delphi Mock (Isolation Framework)

https://github.com/VSoftTechnologies/Delphi-Mocks

https://delphisorcery.blogspot.com/2015/02/testinsight-unit-testing-like-pro.html

Test Insight

https://bitbucket.org/sglienke/testinsight/wiki/Home

Convite – Dia do ACBr

Dia do ACBr
Automação Comercial Brasil

10/11
Sábado
Inscreva-se

Parque Tecnológico de Sorocaba/SP

OBRIGADO

Juliomar Marchetti

juliomarmarchetti@gmail.com

Telegram: @juliomar

juliomar

www.juliomarmarchetti.com.br

juliomar_m

juliomar-marchetti-9485003a

Embarcadero Conference

embarcadero®