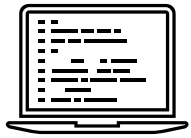


# Delphi et les tests unitaires



---

Webinaire du 3 novembre 2020



# PRÉSENTATEUR

## PATRICK PREMARTIN

MVP Embarcadero

Prestataire informatique freelance et formateur Delphi

CV et contacts :

<https://www.linkedin.com/in/patrickpremartin/>

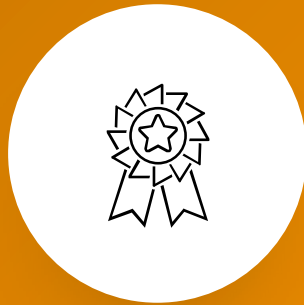
Blog Delphi/Pascal :

<https://developpeur-pascal.fr>

B A R N S T E N

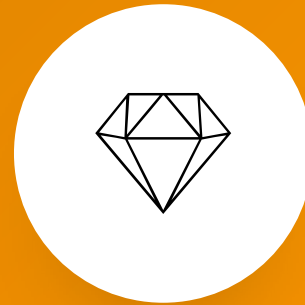
# QUI NOUS SOMMES

— ..



## Distributeur Officiel

Partenaire exclusif de Embarcadero Technologies, largement reconnu pour ses produits de programmation primés.



## Outils Embarcadero

Les meilleurs outils de développement multi-plateformes. Créez une fois et déployez des applications modernes sur chaque plateforme.



## Support Technique

Nos techniciens sont à votre service à tout moment pour répondre à vos questions et vous fournir des informations sûres.

O L F S O F T W A R E

## NOTRE PARTENAIRE



Société de prestations informatiques.  
Edition de sites Internet, logiciels, vidéos et livres.  
Centre de formation référencé sur Datadock.

Infos et contacts : <https://olfsoftware.fr>

Formations en entreprise : <https://se-former-a-delphi.fr>

Formations en ligne : <https://apprendre-delphi.fr>



# INTRODUCTION



Le cycle de vie d'un logiciel ne s'arrête pas au développement, aux tests utilisateurs et à la mise en production/vente. Il y a aussi l'idée de départ, l'analyse des besoins, la conception, puis plusieurs étapes de tests si on veut faire les choses proprement et dans les règles de l'art (dépendant beaucoup de l'impact du logiciel et son public).

# INTRODUCTION



Aujourd'hui nous allons nous concentrer sur le travail que peuvent produire les développeurs afin de limiter le travail des testeurs en aval.

# PHASES DE TEST

Différents types de tests :

- Unit testing
- Smoke testing
- Integration testing
- System testing
- Regression testing
- Performance, load et stress testing
- Acceptance testing



# TESTS UNITAIRES



Ce qui nous intéresse aujourd'hui ce sont les tests unitaires, je vous laisse chercher pour les autres si vous ne savez pas à quoi ils correspondent.

Petit tour sur Wikipédia pour la définition:

[https://fr.wikipedia.org/wiki/Test\\_unitaire](https://fr.wikipedia.org/wiki/Test_unitaire)

# TESTS UNITAIRES



*En programmation informatique, le test unitaire (ou « T.U. », ou « U.T. » en anglais) ou test de composants est une procédure permettant de vérifier le bon fonctionnement d'une partie précise d'un logiciel ou d'une portion d'un programme (appelée « unité » ou « module »).*

*Dans les applications non critiques, l'écriture des tests unitaires a longtemps été considérée comme une tâche secondaire. Cependant, les méthodes Extreme programming (XP) ou Test Driven Development (TDD) ont remis les tests unitaires, appelés « tests du programmeur », au centre de l'activité de programmation.*

# TESTS UNITAIRES



En clair les tests unitaires sont des tests développés pour s'assurer que les modules développés pour le logiciel fonctionnent et restent fonctionnels durablement malgré les modifications qui leur sont apportées.

Ils ne permettent pas de tester l'interface utilisateur. Ils ne sont là que pour le code.

# QUEL INTERET ?



On pourrait voir les tests unitaires comme une perte de temps et dans certains cas ils le sont.

En revanche ils sont vitaux dans notre monde ultra connecté où les logiciels dépendent de morceaux provenant d'un peu partout. C'est surtout vrai pour les applications en ligne, mais ça le devient pour les « vrais » logiciels.

# QUEL INTERET ?



Les tests unitaires peuvent servir dans des équipes de développeurs afin de s'assurer que ce qui est nécessaire à l'un et développé par un autre sort bien le résultat attendu histoire de savoir d'où viennent les problèmes lorsqu'il y en a une fois le code synchronisé.

# QUEL INTERET ?



Les tests unitaires permettent également de s'assurer que le code du logiciel de base est cohérent dans la durée, que les modifications apportées au code ne cassent rien ailleurs.

Les tests unitaires sont à mettre dans la boîte à outil des tests de non régression.

# QUEL INTERET ?



Grâce aux tests unitaires on peut tester notre code mais aussi sa dépendance avec les frameworks, librairies et modules développés par des tiers.

# PREMIER EXEMPLE



Un programme contient une fonction permettant d'additionner deux valeurs numériques.

```
function Ajoute(a, b: integer): integer;  
begin  
    result := a + b;  
end;
```

# PREMIER EXEMPLE



Le test unitaire de cette fonction pourrait être sous cette forme:

```
procedure TestAjoute;  
begin  
    if 10 <> Ajoute(5, 5) then  
        raise exception.Create('erreur');  
    if 0 <> Ajoute(0, 0) then  
        raise exception.Create('erreur');  
    if 20 <> Ajoute(5, 15) then  
        raise exception.Create('erreur');  
    if 65535 <> Ajoute(32768, 32767) then  
        raise exception.Create('erreur');  
end;
```

# PREMIER EXEMPLE



Dans ce cas peu de risque de se planter jusqu'au jour où la taille du type « integer » change selon le système d'exploitation ou le processeur sur lequel on travaille.

Autant être prévenu par des tests automatisés plutôt que s'en apercevoir par hasard en production quand on le voit...

# DEUXIEME EXEMPLE



Prenons un autre cas typique des logiciels développés avec Delphi avant la version 2009. On utilise le type char au lieu de byte pour gérer des zones mémoires ou tableaux d'octets.

Une fois passés à une version récente les caractères ne sont plus des octets puisque Unicode est devenu la norme.

# DEUXIEME EXEMPLE



Le programme fonctionne peut-être encore physiquement, mais les données traitées ne sont pas les bonnes.

Un test unitaire vérifiant que la zone mémoire traitée fait la taille qu'on attend d'elle pourrait éviter des soucis.

# COMMENT FAIRE SOUS DELPHI ?



L'exemple d'avant avec les exceptions permet de réaliser des tests unitaires, mais ce n'est pas hyper pratique à l'usage. On a d'autres solutions pour travailler proprement.

# COMMENT FAIRE SOUS DELPHI ?



Voici quelques solutions disponibles pour les projets Delphi :

- Les assertions
- À la main
- DUnit
- DUnitm
- DUnitX

# ASSERTIONS DANS DELPHI



Les assertions sont gérées directement par le compilateur dans le projet que vous développez.

Il suffit d'utiliser l'instruction [assert\(\)](#)

Elles sont actives par défaut dans toutes les configurations et toutes les plateformes. Vous pouvez le modifier dans les options de projet.

# ASSERTIONS DANS DELPHI



L'avantage est que ça s'insère simplement dans le code, pas besoin de faire quoi que ce soit de plus.

L'inconvénient, c'est qu'elles s'exécutent en même temps que le code, donc uniquement si on passe dessus à l'utilisation du logiciel.

# DEMONSTRATION



Exemple d'un programme utilisant les assertions pour s'autotester en cours d'utilisation.

# METHODE MANUELLE



Sur le principe des assertions vous pouvez aussi coder directement vos tests unitaires dans les unités de vos projets.

Il suffit de les activer dans certains événements ou dans l'initialisation des unités.

# METHODE MANUELLE



Évitez de les laisser s'exécuter dans la version de production de vos logiciels. Utilisez une définition conditionnelle ou directement DEBUG.

Assurez-vous de récupérer les informations liées à vos tests qui échouent sous forme de log (fiche « log », fichier, mail, FTP, CodeSite, ...)

# DEMONSTRATION



Exemple d'un programme proposant des tests unitaires manuels.

# LIBRAIRIE DUNIT



DUnit est un projet open source qui date des années 199x.

Il est considéré comme obsolète pour les projets récents mais fonctionne toujours très bien en environnement Windows.

Plus d'infos sur <http://dunit.sourceforge.net/>

# LIBRAIRIE DUNIT



Embarcadero la propose en option lors de l'installation de RAD Studio, Delphi et C++Builder.

Elle n'est plus recommandée pour les projets Delphi mais sert toujours pour les projets C++Builder.

# LIBRAIRIE DUNIT



DUnit ajoute un assistant de création de projet de test et un assistant de création de cas de test.

Le projet de test correspond à un ou plusieurs projets « réels ». C'est un projet Delphi.

Le cas de test est une unité liée à un projet de test dans laquelle vous allez coder vos tests unitaires.

# LIBRAIRIE DUNIT



Vous trouverez plus d'infos sur son usage dans sa documentation sur

[http://docwiki.embarcadero.com/RADStudio/Sydney/fr/Présentation de DUnit](http://docwiki.embarcadero.com/RADStudio/Sydney/fr/Présentation_de_DUnit) ou

<http://dunit.sourceforge.net/#Documentation>

Et à travers de nombreuses vidéos disponibles en ligne dont <https://youtu.be/1KeQf1urVQ4>

# LIBRAIRIE DUNITM



Egalement open source, DUnitm n'est pas proposée par Embarcadero.

Elle est disponible sur GitHub :

<https://github.com/glenkleidon/DelphiTips/wiki/DUnitm---Mini-Test-Framework>

# LIBRAIRIE DUNITM



L'objectif annoncé est de simplifier la création de tests unitaires sur vos projets. Ca reste dans tous les cas du codage à faire à la main.

Vous trouverez des démos sur la page de présentation et dans cette playlist YouTube :

<https://www.youtube.com/playlist?list=PL42y13vA83auEzqLTzmkwnQuC6pyOxlQE>

# LIBRAIRIE DUNITX



DUnitX est fournie par Embarcadero sous forme d'option lors de l'installation de Delphi, C++Builder et RAD Studio. C'est aussi une librairie open source dont le dépôt se trouve sur <https://github.com/VSoftTechnologies/DUnitX>

Elle propose un assistant de projet DUnitX et d'unité DUnitX.

# LIBRAIRIE DUNITX



L'automatisation des tests se fait à l'aide d'attributs dans les déclaration des unités DUnitX comme nos jeux de données de test.

Les tests sont codés une seule fois.

DUnitX s'occupe de les appeler avec les paramètres déclarés en attributs.

# LIBRAIRIE DUNITX



Le mode d'emploi de base de DUnitX est disponible sur

[http://docwiki.embarcadero.com/RADStudio/Sydney/fr/Présentation de DUnitX](http://docwiki.embarcadero.com/RADStudio/Sydney/fr/Présentation_de_DUnitX)

# LIBRAIRIE DUNITX



Il y a également un mode d'emploi pour la conversion de projets de tests DUnit en projets DUnitX sur

[http://docwiki.embarcadero.com/RADStudio/Sydney/fr/Comment convertir les tests DUnit en DUnitX](http://docwiki.embarcadero.com/RADStudio/Sydney/fr/Comment_convertir_les_tests_DUnit_en_DUnitX)

# DEMONSTRATION



Exemple de projet Delphi avec son projet de tests unitaires réalisés avec DUnitX.

# ALLER PLUS LOIN



L'avantage des tests unitaires est qu'ils permettent dans certains cas de s'assurer que les versions transmises entre développeurs ou avant compilation automatisée sont fonctionnelles.

Les outils d'intégration continue comme Jenkins peuvent gérer les résultats et agir en fonction d'eux.

# ALLER PLUS LOIN



Plusieurs extensions sont disponibles pour l'IDE de Delphi afin de gérer les tests en direct sans lancer le programme de tests. Par exemple :

- Test Grip :

<https://github.com/GDKsoftware/TestGrip>

- Test Insight :

[https://bitbucket.org/sglienke/testinsight/wiki/  
Home](https://bitbucket.org/sglienke/testinsight/wiki/Home)

# TestGrip de GDKSoftware

— ..

C'est un projet open source disponible sur <https://github.com/GDKsoftware/TestGrip>

Une vidéo de 2012 montre son fonctionnement : <https://youtu.be/cgl-VOAVGd8>

# TestInsight de Stefan Glienke



Conçu et maintenu par Stefan Glienke, MVP  
Embarcadero allemand, Test Insight est  
disponibles sur

[https://bitbucket.org/sglienke/testinsight/wiki/H  
ome](https://bitbucket.org/sglienke/testinsight/wiki/Home)

# TestInsight de Stefan Glienke



L'installation se fait très simplement et fournit un assistant à activer sur les projets de tests DUnit ou DUnitX.

Une fois activé on n'a plus à compiler le projet de test. L'assistant s'en occupe lui-même régulièrement et nous informe des résultats en temps réel.

# TestInsight de Stefan Glienke

— ..

Un très bon tuto a été rédigé par David Millington sur son blog à l'adresse

<https://parnassus.co/unit-testing-with-testinsight/>

# DEMONSTRATION



Exemple de projet de tests unitaires DUnitX avec Test Insight activé.

# CONCLUSION



Nous avons vu plusieurs solutions pour faire des tests unitaires dans les projets Delphi.

Choisissez la méthode qui vous convient le mieux et habituez-vous à en faire régulièrement au moins sur les fonctionnalités sensibles de vos projets.

# CONCLUSION



Utilisez les assertions en pensant à les désactiver sur les versions RELEASE pour ne pas perdre de temps CPU « inutile ».

C'est simple et rapide à mettre en place pour les sessions de test développeur/utilisateur.

# CONCLUSION



Utilisez DUnitX pour les « vrais » tests unitaires exécutables en masse régulièrement avant de fournir des versions de vos logiciels à d'autres.

Activez les dans votre système d'intégration continue si vous en avez un.

# CONCLUSION



Et n'oubliez pas que le travail nécessaire à la mise en place de tests unitaires peut permettre de gagner du temps et de la fiabilité sur les projets maintenus à plusieurs sur des années.

# PLUS SUR CE SUJET



Pour des ressources complémentaires,  
télécharger cette présentation et accéder aux  
exemples, rendez-vous sur :

[https://developpeur-pascal.fr/p/00l-  
webinaire-du-3-novembre-2020-delphi-et-les-  
tests-unitaires.html](https://developpeur-pascal.fr/p/00l-webinaire-du-3-novembre-2020-delphi-et-les-tests-unitaires.html)

# (RE)VOIR CETTE SESSION



La rediffusion de ce webinaire devrait être disponible sur la chaîne YouTube de Barnsten :  
<https://www.youtube.com/c/BarnstenFrance/videos>

Vous y trouverez également d'autres vidéos dont une formation complète aux bases de la programmation avec Delphi VCL et FMX.

# PROCHAINS RENDEZ-VOUS



Prochains webinaires :

- 24 novembre 2020 : [Diffuser nos logiciels et gérer leurs mises à jour](#)
- 29 décembre 2020 : [Utilisation de Git dans Delphi](#)

Rediffusions, détails et inscriptions depuis

<https://developpeur-pascal.fr/p/6007-webinaires.html>

# LIVRES SUR DELPHI



Des livres sur le développement avec Delphi sortent régulièrement. Retrouvez en la liste sur <https://delphi-books.com/>

Le dernier en date est le très attendu « [Delphi GUI programming with FireMonkey](#) » d'Andrea Magni en vente sur <https://amzn.to/3oOloYD> et sur commande en librairie.



# QUESTIONS REPONSES





**CONTACTEZ NOUS**  
**EQUIPE@BARNSTEN.COM**

