LinuxFocus article number 265
http://linuxfocus.org

Gorm et ProjectCenter, les outils RAD de GNUstep


par Georges Tarbouriech
<gt(at)linuxfocus.org>

L´auteur:

Georges est un vieil utilisateur d'Unix. Il adore GNUstep et les outils proposés par cet excellent environnement de développement.

Résumé:

RAD signifie Rapid Application Development (développement rapide d'application). A la fin des années 80, lors de l'apparition de NeXTstep, celui-ci proposait un incroyable outil nommé InterfaceBuilder. Utilisé conjointement à un autre outil appelé ProjectBuilder, il permettait de développer des applications graphiques en un clin d'oeil. GNUstep propose une version libre de ces outils, Gorm.app et ProjectCenter.app.

Traduit en Français par:
Georges Tarbouriech
<gt(at)linuxfocus.org>

Au commencement...

Dès la préhistoire des ordinateurs, le développement logiciel a été un énorme défi. Les ordinateurs étaient très volumineux malgré leur faible puissance. Ils étaient très chers, pas vraiment nombreux et les développeurs ne pouvaient pas les utiliser aussi souvent qu'ils l'auraient voulu puisqu'ils devaient les partager avec d'autres. Les chercheurs ont alors essayé de trouver un moyen de faire accomplir à ces ordinateurs plus d'une tâche à la fois afin d'améliorer leur efficacité. De toute évidence, ils devaient partir de zéro pour concevoir des langages de programmation, prenant en considération les faibles ressources des machines disponibles.
Ainsi, pendant les années 60 de nombreux langages de programmation sont apparus : LISP, FORTRAN, BASIC, Algol68, BCPL, etc. Vint ensuite le langage B dérivé du BCPL mentionné ci-dessus, qui devait bientôt devenir le langage C. Ce dernier a changé le monde de la programmation.
Les langages orientés Objet (SmallTalk, Objective C, C++, etc) sont arrivés plus tard avec "l'ère graphique".
Dans les années 80, quelques machines disposaient d'OS graphiques (Apple Macintosh, Commodore Amiga, Atari ST, etc) et le X Window System était en plein développement. A cette époque, une société travaillait sur une interface graphique pour l'OS2 d'IBM appelée Presentation Manager. Avant de finir son travail, cette société a "sorti" sa "propre" interface graphique pour son DOS, appelée... Windos. Les

deux premières versions étaient pratiquement inutilisables, mais... la troisième a démarré le processus. L'ItAM (Intelligence très Artificielle de Microsoft) était née !
En clair, chaque utilisateur devint un informaticien. A partir de là nous avons vu apparaître des applications "géniales" écrites en utilsant Excel ou Word et VisualBasic :-(
N'en parlons plus !
Heureusement, bien avant d'atteindre cette situation, NeXTstep était né et avec lui apparaissait InterfaceBuilder. Cet outil permettait de construire une interface graphique pour votre application dans un temps très bref et avec une grande facilité. A partir de là, ce genre d'outil s'est multiplié. Parmi beaucoup d'autres, nous pouvons citer Omnis, 4D, Delphi, Kylix, etc. Quelques uns sont multi-plateformes alors que la grande majorité est dédiée à Windos. Mentionnons également qu'il existe des outils libres suivant la même philosophie, Gtk (Gimp Tool Kit) par exemple. Les Unixes propriétaires proposent aussi ce genre d'outils.
La caractéristique principale de ces outils c'est que vous n'avez pas besoin d'écrire le code des 200 fenêtres de votre application mais seulement celui servant à gérer les données.
Que vous aimiez ou non ce type d'outil n'est pas le problème. Le temps de développement est court : c'est un fait (d'où le nom, "Rapid Application Development").
GNUstep nous propose des outils RAD. Il se nomment Gorm et ProjectCenter. Certes, ces outils sont très "jeunes" mais ils fonctionnent. Regardons-les de plus près.

GNUstep

Pour pouvoir utiliser Gorm et ProjectCenter, vous devez installer GNUstep. Comment le faire n'entre pas dans le cadre de cet article. Vous trouverez tout le nécessaire sur le site de GNUstep. Ceci inclut le code source, des HOWTOs, des tutoriels, etc.
Vous pouvez aussi jeter un oeil sur ces articles : GNUstep, l'OpenStep open source et GNUMail.app, la preuve de portabilité.
Les tests pour cet article ont été réalisés sous FreeBSD 4.7 avec Window Maker 0.80.1 en utilisant gnustep-make-1.5.0, gnustep-base-1.5.0, gnustep-gui-0.8.2 et gnustep-back-0.8.2. Ces derniers sont les versions instables de GNUstep les plus récentes. Vous pouvez bien sûr utiliser les versions stables si vous préférez. Enfin, nous avons utilisé le compilateur gcc 3.0.4.

Gorm.app

Gorm signifie Graphic Object Relationship Modeler (ou peut-être GNUstep Object Relationship Modeler, comme indiqué dans le fichier README). C'est un clone de l'InterfaceBuilder de NeXTstep (ou aujourd'hui MacOS X) mentionné plus haut.
Gorm est l'oeuvre de Richard Frith-Macdonald qui a démarré le projet. Aujourd'hui, Gregory Casamento est le nouveau mainteneur et il fait le plus gros du travail avec Pierre-Yves Rivaille. La version actuelle est 0.1.9. Vous pouvez obtenir les dernières améliorations par CVS sur http://savannah.gnu.org/projects/gnustep.
Vous pouvez télécharger la dernière version stable à partir du site de GNUstep.
La philosophie de Gorm (et d'InterfaceBuilder) consiste à proposer à l'utilisateur des objets disposés sur des palettes et à les glisser sur des fenêtres vides afin de concevoir la partie graphique d'une application.

Les objets peuvent être des boutons, des champs, des cases à cocher, des panneaux, etc. Autrement dit, tout ce que vous pouvez ajouter à une fenêtre pour la rendre conviviale. Ensuite vous modifiez ces objets grâce aux inspecteurs. Les inspecteurs permettent de changer les attributs, de définir des connexions, des tailles, une aide, et de manipuler des classes pour les objets sélectionnés. Après avoir créé une classe, vous pouvez ajouter des "outlets" ("canaux") et des actions aux objets.
Ensuite vous devez instancier la classe, ce qui crée un nouvel objet (l'instance) dans la fenêtre principale de Gorm, et vous connectez les "outlets" et les actions aux composants correspondants. Vous faites ceci en glissant, tout en maintenant la touche Ctrl enfoncée, depuis l'instance vers l'objet pour connecter les "outlets" et depuis l'objet vers l'instance pour connecter les actions. Enfin, vous créez le squelette des fichiers source de la classe et c'est tout. Nous y reviendrons.

ProjectCenter.app

ProjectCenter, comme le nom l'indique, est le "coeur" d'un projet. C'est un clone de Project Builder trouvé sous NeXTstep et Mac OS X.
ProjectCenter est l'oeuvre de Philippe C.D.Robert et la version actuelle est 0.3.0. Comme Gorm, vous pouvez le télécharger depuis le site de GNUstep dans la section Developper apps. Bien sûr, vous pouvez obtenir le dernier CVS : nous l'utilisons pour cet article et il s'agit de la version 0.3.1.
A partir de ProjectCenter vous pouvez créer un projet, son interface (en utilisant Gorm), écrire son code source; vous pouvez compiler le projet et l'exécuter (le débogage n'est pas encore disponible). En bref, vous pouvez gérer toutes les ressources réclamées par le projet : code source, documentation, bibliothèques, sous-projets, interfaces, etc.
Lorsque vous créez un nouveau projet, vous pouvez choisir son type. Vous avez le choix entre application, "bundle" (paquet), outil, bibliothèque et application Gorm.
Entre autres choses, ProjectCenter vous propose un éditeur dans lequel vous pourrez compléter le squelette du code généré par Gorm.
Comment Gorm et ProjectCenter fonctionnent-ils ensemble ? Très bien, merci !
Plus sérieusement, nous prendrons deux exemples pour l'illustrer.

Quelques notes

Cet article n'est PAS un tutoriel. L'idée est de montrer la facilité d'utilisation de ces outils tout en insistant sur le fait que vous pourrez vous servir du même code pour GNUstep (c'est-à-dire un tas de plateformes Unix... et si vous aimez "vous battre", sous Windos aussi) et sous MacOS X. La seule chose que vous aurez à faire c'est de créer l'interface pour les différents systèmes puisque les fichiers nib (InterfaceBuilder ou Gorm) ne sont pas portables (au moins pour l'instant).
L'article sur GNUMail.app mentionné précédemment montrait la portabilité du point de vue de l'utilisateur. Celui-ci insiste sur le point de vue du développeur, toujours avec la portabilité présente à l'esprit. Autrement dit, dans GNUMail.app nous nous servons du travail de Ludovic et de ses amis et ici nous concevons une application graphique pour GNUstep et MacOS X.
De nombreux tutoriels sont disponibles, que ce soit pour GNUstep ou MacOS X. Vous pouvez accéder à la plupart de ceux pour GNUstep depuis le site ou depuis http://www.gnustep.net, mais citons-en quelques uns.

- Une application avec Gorm et ProjectCenter par Pierre-Yves Rivaille.
- La page de tutoriels deNicola Pero
- Un tutoriel plus ancien sur la création d'un éditeur HTML :
http://stepwise.com/Articles/Technical/HTMLEditor/
Pour en apprendre davantage, vous pouvez aussi "éplucher" le code source, les fichiers nib, etc, des applications GNUstep existantes (Gorm, ProjectCenter, GNUMail, GWorkspace, etc) et bien évidemment les exemples "gnustep-examples".

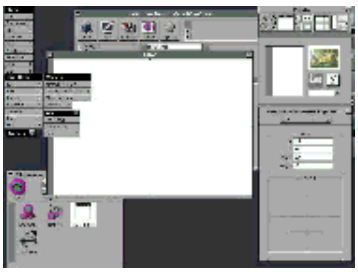
Appelons-le "EditeurTrèsSimple"

Parmi les nombreux tutoriels pour MacOS X et InterfaceBuilder disponibles sur Internet, nous utiliserons celui qui suit comme premier modèle :
http://www.macdevcenter.com/pub/a/mac/2001/05/18/cocoa.html. L'auteur, Mike Beam a écrit bien d'autres tutoriels disponibles sur http://www.macdevcenter.com/pub/ct/37.
Pourquoi celui-ci ? Parce qu'il vous permet d'obtenir un éditeur de texte fonctionnel sans écrire la moindre ligne de code. Ceci montre la puissance de ces outils de développement que ce soit sous MacOS X ou sous GNUstep.
Grâce à ProjectCenter et à Gorm, nous créons un très simple éditeur de texte capable de couper, copier, et coller sous GNUstep. Evidemment, nous ne pourrons pas sauvegarder notre travail: rappelez-vous, nous n'écrirons pas une seule ligne de code. Avec ProjectBuilder et InterfaceBuilder sous MacOS X, nous ferons la même chose. Certes, il reste beaucoup à faire pour améliorer cet éditeur mais nous réserverons cela comme exercice pour le lecteur. Encore une fois, cet article n'est pas un tutoriel ! Allons-y.

Sous GNUstep

Ouvrez ProjectCenter.app et créez un nouveau projet appelé Editor. Choisissez un projet "Gorm Application" en bas de la fenêtre avant de sauvegarder le nom. Ceci vous donne un élément "Interfaces" dans la colonne gauche de ProjectCenter.
Cliquer sur *Interfaces* affiche *Editor.gorm*. Double-cliquez *Editor.gorm* et Gorm.app s'ouvre.
Sélectionnez la fenêtre par défaut (MyWindow) et en utilisant l'inspecteur changez le nom en *Editor* dans Attributes.
Depuis la palette, glissez une "TextView" sur la fenêtre Editor. La VueTexte (TextView) est l'objet le plus gros dans la palette que vous choisissez en cliquant sur l'icône la plus à droite en haut de la fenêtre Palettes. Redimensionnez l'objet de manière à ce qu'il remplisse la fenêtre et c'est tout.
Dans l'inspecteur GormInternalViewEditor (avec le TextView sélectionné), choisissez *Size* et modifiez les valeurs de façon à ce qu'elles correspondent à celles de la fenêtre Editor. Celles-ci sont obtenues de la même manière, en sélectionnant la fenêtre et en vérifiant les tailles dans l'inspecteur GormNSWindow. Si vous ne modifiez pas les valeurs X et Y, par exemple, vous ne pourrez pas écrire sur toute la largeur de l'éditeur, que vous redimensionniez la fenêtre ou pas.
Sauvegardez tout depuis le menu Document de Gorm et quittez pour revenir à ProjectCenter.
Sélectionnez l'icône "Build" et cliquez sur la nouvelle icône du même nom dans la partie inférieure de la fenêtre. Tout devrait bien se passer si vous avez défini les bonnes préférences pour le compilateur, le

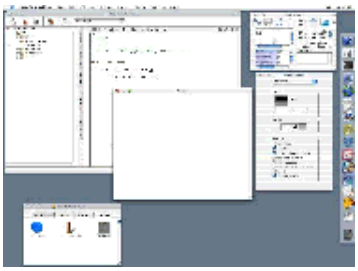
débogueur, etc. Par exemple, sous FreeBSD, vous devez remplacer make par gmake (chemin inclus) en cliquant dans l'icône Settings de ProjectCenter. Vérifiez aussi les chemins dans le menu Preferences de ProjectCenter.
Si la compilation a réussi (ce devrait être le cas !), faites la même chose avec *Run* et vous verrez apparaître l'application Editor. Amusez-vous avec en écrivant, en coupant, en collant, etc. Bien sûr vous pourrez la relancer plus tard en utilisant la commande openapp.
Combien de temps a duré l'opération ? Je dirais quelques minutes.
Voilà à quoi ça devrait ressembler pendant la phase de création :



et l'application en service :

Sous MacOS X

Pas grand chose à ajouter puisque vous devez faire la même chose que ci-dessus. Voici à quoi ça ressemble pendant la création de l'interface :



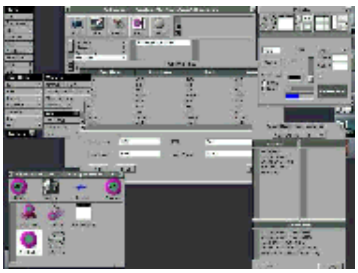
Et l'éditeur "au travail" :

Et maintenant... une application vraiment fonctionnelle

Choisissons un autre exemple de Mike Beam. Cette fois il s'agit d'une application réellement fonctionnelle, capable de gérer des données : un carnet d'adresses. Il est recommandé de lire le tutoriel de Mike pour comprendre comment fonctionne la "chose". De même, vérifiez la liste des tutoriels de Mike puisqu'il propose différentes étapes du processus de développement pour une même application afin de l'améliorer.
De nouveau nous créons et exécutons l'application sous GNUstep et MacOS X.

Sous GNUstep

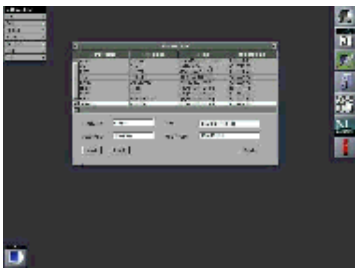
Comme vous l'avez fait pour l'éditeur, démarrez ProjectCenter. Sélectionnez une application Gorm et appelez-la AddressBook. Depuis ProjectCenter lancez Gorm en double-cliquant dans Interfaces -> AddressBook.gorm. Glissez une VueTable (TableView) de la palette vers la fenêtre par défaut. En d'autres termes suivez les indications de Mike comme vous le feriez sous MacOS X. Vous devrez adapter quelques petites choses parce qu'elles fonctionnent différemment dans Gorm et dans InterfaceBuilder.
Par exemple, le nombre de colonnes de la TableView ne peut pas être défini dans l'inspecteur des attributs de Gorm. Pour faire simple, copiez une colonne et collez-la à côté jusqu'à obtenir le nombre de colonnes requis (4 dans notre cas). Vous devriez vous retrouver avec quelque chose de ce genre :



Lorsque vous avez terminé, sauvegardez l'ensemble et revenez à ProjectCenter pour taper ou modifier le code. Si vous avez fait des erreurs, Mike fournit les sources complètes de l'application. Si vous les téléchargez, il vous suffit alors de copier et de coller le code dans vos propres fichiers Controller.m et Controller.h générés par Gorm. N'incluez pas (import dans la terminologie Objective C) *Cocoa.h*

puisque c’est réservé à MacOS X. Vous pouvez d’ailleurs conserver le squelette "écrit" par Gorm et modifier quelques petites choses. Par exemple, remplacez *void* par IBAction dans Controller.h et Controller.m. Ajoutez IBOutlet avant l’id des outlets dans Controller.h.
De toute manière, si vous préférez, vous pouvez conserver la totalité du code proposé : remplacez seulement l’inclusion de cocoa par *#import <AppKit/AppKit.h>*. Maintenant vous pouvez compiler et exécuter l’application.
Voilà : vous pouvez commencer à jouer avec votre nouveau carnet d’adresses.

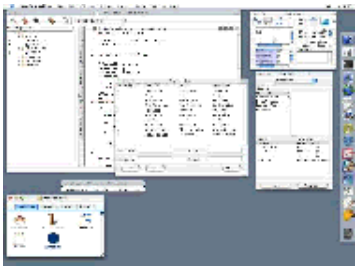
Voici à quoi il ressemble :



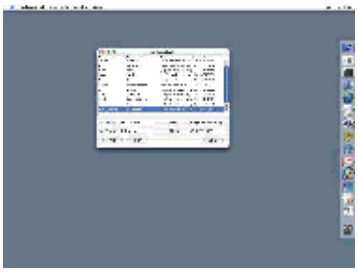
Sous MacOS X

Mike Beam a fait tout le travail: que pourrais-je ajouter ?

Voici une capture lors du processus de développement sous MacOS X:



Voilà l’application :



Alors quoi ? Eh bien, la création de l’interface a duré quelques minutes et le code complet représente quelques 2000 octets. Pas mal, non ?
Alors, bien sûr, beaucoup d’améliorations sont possibles mais l’ensemble fonctionne ! Maintenant vous

pouvez jouer avec le code pour obtenir un meilleur carnet d'adresses...

GNUstep et MacOS X

Il est évident que les outils de développement de GNUstep ne peuvent pas être aussi avancés que ceux d'Apple. Apple et NeXT représentent 15 ans d'expérience et des centaines de développeurs. GNUstep est le travail (gratuit) de quelques individualités qui doivent faire autre chose pour vivre. Par conséquent, ne soyez pas surpris de trouver beaucoup plus de classes disponibles dans InterfaceBuilder que dans Gorm. Rappelez-vous, Gorm est à la version 0.1.9 (ou 0.2.0).
De plus, nous avons fait les tests de la manière la moins favorable. C'est-à-dire, nous avons "porté" d'OS X vers GNUstep. Il aurait été plus facile de le faire dans l'autre sens à cause des différences entre les outils évoquées ci-dessus.
Par exemple, porter des applications développées sous MacOS X 10.2 serait plus difficile puisque les outils de développement d'Apple ont beaucoup évolué. Comme déjà dit, il y a beaucoup de nouvelles classes ou des classes plus élaborées.
Toutefois, les outils reposent sur la même philosophie qu'ils fonctionnent sous GNUstep ou MacOS X... et GNUstep s'améliore tous les jours. Il y a une chose que j'apprécie beaucoup : les gens de GNUstep travaillent vraiment ensemble. Ils s'aident mutuellement pour ce qui concerne les projets individuels mais ils contribuent également à l'amélioration de GNUstep proprement dit. C'est la façon de travailler dans le logiciel libre que j'aime. Félicitations pour ce comportement Mr.Fedor et vos amis.

Imaginez...

Le but de cet article était de montrer la puissance des outils "RAD" de GNUstep, Gorm.app et ProjectCenter.app. Malgré leur "jeunesse" ils sont parfaitement capables de vous permettre de développer facilement des applications sympathiques.
De plus, ces outils offrent une façon de travailler très plaisante tout en restant efficace. L'Objective C est un langage très compact, et à mon avis, plus facile à apprendre que le C++ pour quelqu'un connaissant le C (je sais, je l'ai déjà dit !). Ceci permet de développer des applications esthétiques (certes, c'est une histoire de goût, mais personnellement j'aime beaucoup) tout en conservant une taille plutôt réduite.
Je dois reconnaître que je ne me suis jamais vraiment remis du choc causé par ma première rencontre avec une machine NeXT. Le fait qu'Apple sorte une version moderne de NeXTstep me ravit. C'est aussi pourquoi je suis un inconditionnel de projets tels que GNUstep ou Window Maker. Toutefois, même si j'adore le logiciel libre, je ne suis pas un "intégriste" et par conséquent je n'ai rien contre le logiciel propriétaire (bon, peut-être un peu envers un certain éditeur... mais vraiment un peu !).
GNUstep peut bénéficier du travail d'Apple... mais Apple peut bénéficier de celui de GNUstep aussi. GNUstep n'est pas un concurrent d'Apple, c'est du logiciel libre. Que je sache, le logiciel libre est largement utilisé dans MacOS X. Ceci pour dire qu'amener encore plus de logiciel libre à Apple ne peut en aucun cas être une mauvaise chose. Ce qu'ont fait Ludovic et ses amis avec GNUMail.app est un très bon exemple de ce qui pourrait se produire.
"J'ai fait un rêve"... Apple fournissait la plupart du code source de ses outils de développement à GNUstep. Les développeurs de GNUstep et d'Apple travaillaient ensemble afin de fournir de superbes applications aux utilisateurs d'Unix. Et petit à petit, les gens réalisaient qu'ils pouvaient vivre sans

Windos...
Malheureusement, ce n’était qu’un rêve ;-)
Quoi qu’il en soit, si vous ne connaissez pas GNUstep et ses applications, n’hésitez pas à les essayer. GNUstep est un environnement de développement (à ne pas confondre avec un environnement de bureau) et des outils tels que Gorm et ProjectCenter vous fournissent tout le nécessaire à la création, à l’invention. En d’autres termes, avec un peu d’imagination, vous pouvez développer des "produits" très différents de ce que nous avons coutume de voir aujourd’hui : des clones d’applications Windos !
Nous vivons une époque formidable !

Merci...

Aux personnes de GNUstep: A.Fedor, N.Pero, G.Casamento, P.Y.Rivaille, N.Roard, L.Marcotte, R.Frith-Macdonald, P. C.D.Robert, E.Sersale, A.Froloff, F.Kiefer, M.Viviani, M.Guesdon et tous ceux que j’ai oublié pour le beau travail, qu’il s’agisse du "framework" ou des applications.
Aux personnes de Window Maker : A.Kojima, D.Pascu et leurs compères de nous avoir offert une interface NeXTstep libre pour X.
A J.M.Hullot et B.Serlet pour avoir inventé InterfaceBuilder.
A "Steve Jobs INC." pour NeXT, NeXTstep et MacOS X.
A tous ceux non cités ici qui ont contribué à rendre notre vie professionnelle beaucoup moins triste.

Translation information:
en --> -- : Georges Tarbouriech <gt(at)linuxfocus.org>
en --> fr: Georges Tarbouriech <gt(at)linuxfocus.org>

2005-01-14, generated by lfparser_pdf version 2.51