

Client riche : vers les interfaces graphiques agiles

Publié dans le magazine *Programmez!*
d'XXX 2006

Clairement avantagées par des coûts de déploiement extrêmement bas, les applications Web se sont imposées dans l'entreprise, supplantant pratiquement les applications graphiques classiques. Avec le recul, le prix à payer est lourd : ergonomie rudimentaire, temps de réponse pas toujours très bons, coûts de développement élevés. Face à ce constat, il faut bien admettre que les applications graphiques d'autant n'avaient pas ces défauts là. Pourtant, on ne regrettera pas forcément leurs coûts de déploiement élevés, et leur maintenabilité médiocre, résultant de l'absence d'une séparation minimale entre la présentation, la logique métier et l'accès aux données. Pourtant, n'est on pas légitimement en droit d'espérer le meilleur des deux mondes, avec en bonus, une vraie séparation en couche. Cette nouvelle approche, c'est le client riche. Combiné avec un SOA qui fournit les services métiers, le client riche est l'un des fondements d'un système d'information agile.

Ainsi de nombreuses technologies sont en train d'émerger (ou de ré-émerger) pour nous permettre d'implémenter des clients riches. Parmi ces technologies on compte : AJAX, Laszlo, XUL, Windows Presentation Foundation (WPF), Java Desktop Network Components (JDNC) ... Pour fonctionner, le client riche nécessite un environnement d'exécution. En conséquence, un critère de choix important est la disponibilité en standard de l'environnement d'exécution ou encore sa facilité de déploiement. Ainsi, JDNC nécessite Java, et WPF requiert .NET, AJAX fonctionne dans un navigateur et utilise JavaScript, Laszlo nécessite le plugin Flash, très répandu.

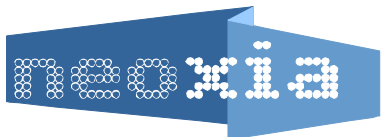
Les interfaces graphiques peuvent être décrites selon 3 axes de préoccupation: l'aspect visuel pur, le binding (relier les composants aux données affichées), et le comportement (réponse aux événements). Les deux premiers aspects peuvent bénéficier d'une approche déclarative. En utilisant un langage spécifique (comme avec WPF, XUL, Laszlo ...), on

peut décrire l'aspect graphique et le binding, sans avoir recours à du code, et surtout en se faisant assister d'outils totalement visuels. Seule l'aspect comportement reste implémenté par du code, dont la taille est réduite significativement. Le binding permet également de séparer la partie présentation, tout en bénéficiant d'une approche RAD, mais enfin structurée, et donc plus maintenable.

Les possibilités du sous-système visuel varient notablement selon la technologie de client riche. De ce point de vue, Laszlo et XUL sont bien lotis, WPF sera très richement doté, la galaxie AJAX reste elle très inhomogène, quant à JDNC, il est clairement en devenir.

Selon la technologie choisie, le sous-système de binding permet de relier l'interface graphique à divers types de données tels que : des documents XML (par exemple retournés par un service Web), des objets classiques (les fameux POJO pour Java), des données issues de la base de données, des données issues de fichiers texte tabulaires. Il peut également gérer le rafraichissement de l'interface, lorsque les données bindées sont modifiées. Éventuellement, il permet aussi de gérer la synchronisation inverse, à savoir la mise à jour des données bindées lorsqu'elles sont modifiées via l'interface. Bien-sûr, les possibilités offertes par le sous-système de binding diffèrent largement selon la technologie de client riche choisie. De ce point de vue, Laszlo n'est pas très bien loti, quant à AJAX, la situation est très différentes selon le framework. Pour l'heure, il semble que c'est bien WPF qui offrira, à sa sortie, le plus de possibilités dans ce domaine. Quant à JDNC, il semble prometteur, mais il doit vraiment aboutir et faire ses preuves.

Dans le choix d'une technologie de client riche, les possibilités offertes par le sous-système visuel, et le sous-système de binding sont un critère fondamental, au moins aussi important que la facilité de déploiement.



De ce dernier point de vue, AJAX représente un choix de transition pertinent, mais les implémentations sont très variables et l'offre très éclatée. Les autres technologies nécessitent un environnement d'exécution, qui peut être installé via les infrastructures de déploiement automatisées de l'entreprise. Au final, si les technologies les plus prometteuses sont à

venir, on peut faire dès maintenant du client riche avec AJAX, Laszlo, ou XUL. Une autre approche, souvent écarté d'office, consiste à développer des applications Windows Form, Swing ou SWT, et à rendre le déploiement transparent en utilisant ClickOnce ou Java WebStart.

NDX

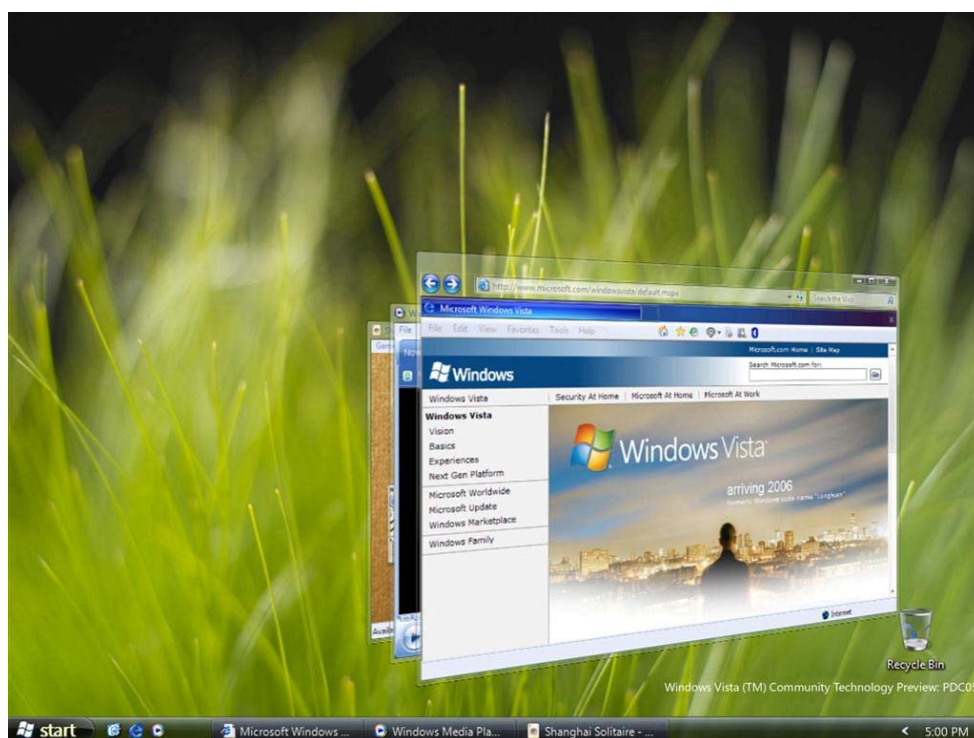


Figure 1 : Vista-Xaml

Washington Plaza
40, rue de Washington
75008 Paris
Fax: +33 1 58 36 40 69

Neoxia
www.neoxia.com
contact@neoxia.com
Tél.: +33 1 58 36 40 60

Neoxia

Neoxia est un cabinet de conseil spécialisé dans les nouvelles technologies de l'information. Depuis sa création en 2000, il offre à ses clients un large éventail de services allant du conseil, à l'ingénierie en passant par le moniorat et la formation.

Neoxia a été le premier à mettre en place une offre spécifique de tutorat des équipes techniques projet, baptisée TechCoaching®. Cette offre consiste en la mise à disposition d'un consultant Coach, expert technique reconnu, pour assurer des sessions de formation in-situ visant à transmettre des connaissances techniques et le vécu de l'intervenant sur des projets.