

Choisir une solution RIA

Comme l'indique un article apparu sur *Ajaxian*⁽¹⁾ et datant déjà de plus de six mois, le panel de solutions permettant de réaliser des applications AJAX/RIA est pléthorique : à l'époque, plus de 210 outils étaient proposés, soit 80 de plus que l'année précédente. Et le phénomène n'est pas prêt de s'arrêter.

Confronté à un tel bouillon de culture, le chef de projet, le directeur de projet, le responsable informatique sont désemparés : comment choisir LA solution qui convient le mieux, quels sont les critères qui doivent guider leur choix, quels sont les risques qu'ils prennent.

Se reconnaître dans le foisonnement RIA

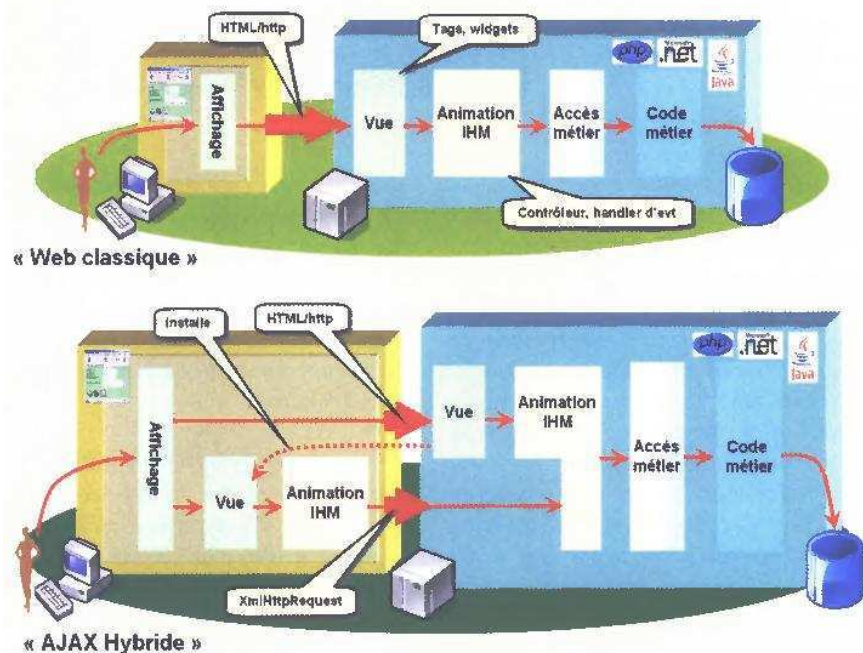
Où en est le RIA aujourd'hui ? Pour simplifier, nous pouvons considérer qu'il existe deux branches principales : AJAX et les technologies "applet-like" :

- AJAX est la branche la plus prospère, elle repose sur les capacités intrinsèques des navigateurs du marché : langage JavaScript, communication http asynchrone. Ce principe fondateur se décline sur le marché en une myriade de solutions et d'architectures. Nous y reviendrons.
- Les technologies "applet-like" sont celles qui utilisent le navigateur comme hôte d'un système qui vient s'y incruste, mais qui lui est en fait exogène. Il s'agit donc d'une solution de type "client lourd" qui avance masquée : les capacités du navigateur sont peu ou pas utilisées.

Comprendre AJAX

AJAX n'est pas un produit, c'est une idée. Une idée qui s'incarne aujourd'hui de façon extrêmement variée. On peut néanmoins dégager deux grandes stratégies :

- Une stratégie de type essentiellement "client/serveur". La partie cliente est implémentée en JavaScript. La partie serveur utilise une des plates-formes les plus courantes du marché : Java, .NET, PHP, etc. Toute l'IHM (présentation, logique) est hébergée par le navigateur. Le serveur n'exécute que les points de service dont le client a besoin. Il s'agit donc d'une stratégie "AJAX Total". L'utilisation directe de frame-



works JavaScript (Dojo, Ext, YahooUI, Script.aculo.us/prototype) peut entrer dans cette stratégie. Le Google Web Toolkit présente une solution originale – et efficace – pour construire des applications de ce type.

- L'autre stratégie consiste à confier une partie importante de la logique IHM au serveur. La partie cliente, généralement constituée d'une armature HTML enrichie de nombreux réflexes JavaScript, n'assure que des fonctions d'animation avancée (menus, arbres, glisser-déposer). Elle contient peu ou pas d'intelligence fonctionnelle. Cette stratégie marie généralement AJAX avec un framework Web classique comme ASP.NET, Struts, JSF, PHP, RubyOnRails. C'est pourquoi j'ai appelé cette stratégie : "AJAX hybride" dans la suite de cet article.

Il n'existe pas de meilleure stratégie AJAX : l'AJAX total n'est pas intrinsèquement supérieur ou inférieur à l'AJAX hybride, tout dépend de critères spécifiques à l'application à déve-

lopper. C'est ce que nous verrons dans la deuxième partie de cet article.

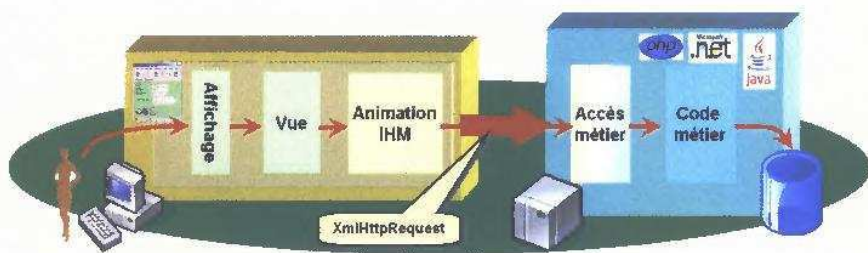
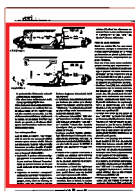
Les technologies "applet-like"

Historiquement, le premier système de ce type est l'applet Java. Flash en représente le meilleur aboutissement (et aujourd'hui le plus utilisé). Microsoft avec Silverlight complète l'offre. JavaFX - comme cela est expliqué dans un autre article de ce numéro - n'est qu'une manière simplifiée de produire des applets Java.

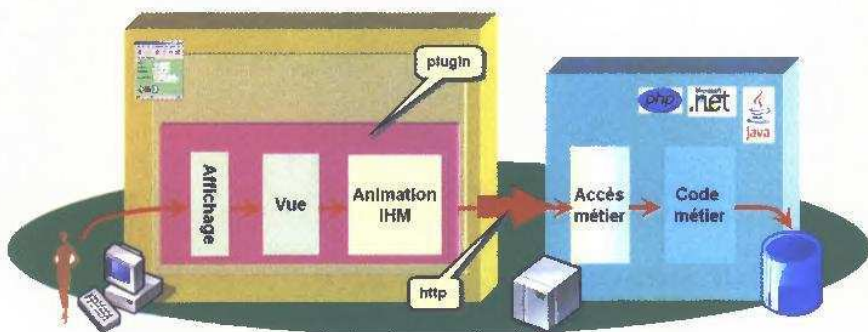
Bien que les plates-formes techniques soient différentes (Java, .NET, flash), les offres "applet-like" présentent beaucoup de similitudes :

- Elles utilisent toutes un système déclaratif de construction d'interface (mxml pour flex/flash, xaml pour Silverlight, JSON-like pour JavaFX).
- Elles permettent toutes de lier des propriétés d'objets avec les propriétés des widgets,

(1) <http://ajaxian.com/archives/210-ajax-frameworks-and-counting>



« AJAX total »



« applet-like »

la synchronisation étant ensuite automatiquement assurée par le système.

- Elles offrent toutes l'intégration des appels à un serveur via des Web Services.

Malgré l'hétérogénéité des plates-formes techniques sous-jacentes, ces similitudes font que les réflexes acquis sur une plate-forme peuvent être reconduits sur une autre. Les cas d'utilisation et les architectures adaptées sont les mêmes. Un développeur Flex/Flash peut devenir, par exemple, un développeur Silverlight ou JavaFX compétent après une phase d'apprentissage réduite.

Les vraies questions

Avant de choisir une solution, il est indispensable de savoir de quoi nous avons vraiment besoin et quelles sont nos priorités. Selon le type d'application, l'audience visée, le modèle économique, ces critères changent. Or ce sont eux qui conditionnent la validité d'un choix. Faisons un tour rapide de ces critères :

- **La pérennité.** Si quelques solutions se dégagent aujourd'hui du lot, il est encore bien trop tôt pour savoir qui finalement l'emportera. Même les produits soutenus par les plus grands ne sont pas à l'abri des caprices du marché.
- **L'ergonomie.** Les solutions " AJAX hybride " permettent de fluidifier les manipulations utilisateur courantes (menus, onglets saisie assistée, arbres, tables de données), les solutions " AJAX Total " offrent en plus des

fonctions d'ergonomie très avancées ou/et très originales.

- **La productivité.** Le développement RIA n'est pas forcément plus coûteux qu'un développement Web classique : certaines solutions " AJAX hybride " en particulier peuvent profiter d'un outillage ou de frameworks remarquables (Visual Studio, Seam).
- **La performance.** Les solutions " AJAX hybride " ont tendance à multiplier les échanges avec le serveur. Les solutions " AJAX total " ou " applet-like " exigent, elles, davantage du navigateur client, qui n'est pas toujours capable de faire face.
- **La prise en compte de l'existant.** La technologie déjà utilisée peut induire quelle extension AJAX doit être utilisée. Autre sujet : la réutilisation des compétences disponibles. Il est plus facile de passer de ASP.NET à ASP.NET AJAX, que d'utiliser le Google Web Toolkit.
- **L'ubiquité.** C'est la capacité d'une solution à s'adapter aux différents environnements clients : les principaux navigateurs du marché ou les divers systèmes d'exploitation. Les surprises sont fréquentes.
- **La robustesse.** La fiabilité d'un logiciel repose surtout sur la qualité de son code. Mais cette qualité dépend aussi des facilités offertes par la plate-forme RIA utilisée.
- **La sécurité.** Sur Internet, c'est une question essentielle. Dans un intranet, la question est généralement moins sensible. Certaines

solutions RIA offrent des mécanismes permettant d'éviter les deux problèmes majeurs de la sécurité RIA : le " XSS " et le " SQL Injection ". D'autres n'offrent rien.

Faire un choix pertinent

Choisir une solution RIA c'est donc autant connaître les avantages et inconvénients des diverses solutions proposées par le marché que de connaître les priorités qu'il faut considérer et qui sont elles, très spécifiques au projet ou à l'entreprise.

Dans cet article nous allons considérer quelques solutions, parmi les plus connues :

- **Les frameworks JavaScript** ont comme avantage de proposer une liberté maximale au développeur qui veut virtuellement faire ce qu'il veut. Cette liberté se paie au prix fort : l'outillage reste rustique, ubiquité, robustesse et sécurité sont entièrement à sa charge.
- **JavaServer Faces (JSF)** avec une extension AJAX (comme a4jsf ou ICEFaces) est la solution reine de type " AJAX hybride " dans le monde Java. Elle possède tous les avantages et les inconvénients des solutions hybrides : productivité (avec Seam), ubiquité, sécurité et robustesse. Les aspects ergonomie et performances sont moins favorisés. La prise en compte de l'existant n'a de sens que s'il est JSF.
- **Le Google Web Toolkit** est une solution de type " AJAX total " qui favorise les aspects performances, ubiquité, robustesse et sécurité grâce à un ingénieux système de développement en Java (et non en JavaScript), puis de compilation en JavaScript. La principale limite de cette solution est la relative complexité de l'intégration client/serveur. La pérennité paraît garantie.
- **ASP.NET AJAX** présente des caractéristiques proches de JSF avec une extension AJAX : C'est l'extension naturelle d'ASP.NET WebForms (dont il n'est en fait qu'une extension). C'est une solution remarquablement outillée par Visual Studio. La prise en compte d'un existant ASP.NET est très aisée.
- **Flex** est la solution de type " applet like " la plus populaire. Elle en a tous les avantages, qui sont les mêmes que ceux d'une architecture " AJAX total ". Le seul point faible est l'intégration client/serveur, nettement moins pratique que ce que propose JSF/Seam ou ASP.NET AJAX.
- **Silverlight** est la solution " applet like " du monde .NET. Elle reprend le principe de construction déclarative de l'IHM popularisé par



Flex. L'ubiquité est encore assez limitée. L'outillage, avec Visual Studio, reste le point fort.

- **JavaFX** est la solution " applet like " du monde Java. Les avantages et limites sont ceux rencontrés pour Silverlight, avec moins de crédibilité et de maturité.

La correspondance entre ces différentes solutions et les critères décrits plus haut est donnée dans le tableau n°1.

Définissons maintenant quelques catégories d'applications types, parmi celles qui sont le plus souvent rencontrées :

- L'extension d'une **application Web existante de manipulation de données** (saisie, restitution, statistiques). Il s'agit d'une application classique que l'on veut faire évoluer (nouvelles fonctionnalités) ou dont on veut améliorer partiellement l'ergonomie (quelques écrans critiques). La plate-forme technique existe et cela contraint forcément le choix de la solution RIA qui doit être greffée (ASP.NET AJAX pour une application .NET par exemple). Une réponse " AJAX hybride " est adaptée.
- Le développement complet d'une **application RIA de manipulation de données** pour un environnement particulier (une entreprise par exemple). Les besoins ergonomiques sont généralement très classiques, supportés par n'importe quelle solution RIA. Un niveau acceptable de performance suffit : le nombre d'utilisateurs est limité. La plate-forme cliente étant connue, l'ubiquité n'est pas indispensable. L'aspect le plus crucial est souvent la productivité car l'application est constituée d'un nombre important d'écrans. Ici aussi, une solution " AJAX hybride " a plus de chance de donner satisfaction.
- Le développement complet d'un **progiciel de manipulation de données**. Il s'agit donc d'un besoin spécifique aux éditeurs. Ce type d'application est une variante du précédent avec des contraintes particulières : la plate-forme cliente n'étant pas connue, l'ubiquité redevient une priorité. Le nombre d'utilisateurs pouvant être extrêmement variable, l'aspect performance doit être considéré avec soin. Dans ce cas, tous types de solution RIA doivent être étudiés.
- Le rajout sur un **portail Web classique** de quelques mécanismes RIA. Il s'agit donc de faire un lifting d'un portail existant en rendant sa manipulation plus fluide. La prise en compte de l'existant est essentielle. Le problème des performances et l'ubiquité sont très aigüés : la population visée est celle

Tableau 1	Frameworks JavaScript	JSF AJAX avec Seam	ASP.NET AJAX	Google Web Toolkit	Flex	Silverlight	JavaFx
Pérennité	★	★★	★★★★★	★★★★	★★★★	★★★★	★★★★
Ergonomie	★★★★★	★★	★★	★★★★	★★★★	★★★★	★★★★
Productivité		★★★★★	★★★★★	★★	★★★	★★★	★★★★
Performance	★★	★	★	★★★★★	★★★★★	★★★★★	★★★★★
Prise en compte de l'existant	★★	★★★★★	★★★★★	★	★	★	★
Ubiquité	★	★★★★★	★★★★★	★★★	★★★	★	★★★★
Robustesse		★★★★★	★★★★★	★★★	★★★	★★★	★★★★
Sécurité		★★★★	★★★★	★★★★★	★★★★	★★★★	★★★★

Tableau 2	Extension appli MD existante	Appli MD dev. Spécifique	Appli MD progiciel	Portail classique	Portail Web 2.0
Pérennité	★	★★	★★★★★	★★★★	★★★★
Ergonomie	★★	★★	★★★★	★★	★★★★
Productivité	★★	★★★★★	★★★★★	★★	★
Performance	★★	★★★★	★★★★	★★★★	★★★★
Prise en compte de l'existant	★★★★★			★★	
Ubiquité		★	★★★★★	★★★★	★★★★
Robustesse	★★	★★	★★★★★	★★★★	★★★★
Sécurité	★★	★★	★★★★	★★★★	★★★★

des internautes, dont on ne maîtrise ni le nombre, ni les plates-formes (forcément variées). Nous sommes sur Internet : la sécurité doit être traitée avec soin. Une solution " applet-like " est indiquée. Nous pouvons aussi envisager une solution de type " AJAX Total ".

- La création d'un **portail Web 2.0**, c'est-à-dire un site offrant des outils dont l'ergonomie est très avancée, originale, utilisant fréquemment des images, des graphismes et des mécanismes comme le glisser-déposer. L'aspect ergonomique devient primordial. Ubiquité, performances et sécurité gardent leur caractère essentiel pour les raisons décrites dans le cas d'un portail Web classique. Une solution " AJAX Total " - ou pourquoi pas " Applet like " paraît la plus pertinente.

La correspondance entre les besoins de ces différents types d'applications et les critères décrits plus haut est donnée dans le tableau n°2.

Conclusion

Une étude du marché des solutions RIA doit être mise en perspective par rapport à des besoins pour être pertinente. On découvre alors que LA solution se hissant au-dessus de toutes les autres n'existe pas (encore ?). Certains outils gagnent aujourd'hui en popularité et rien n'interdit de penser que les limites qu'ils présentent encore ne seront pas dépassées demain. Mais nous n'en sommes pas là aujourd'hui.

■ **Henri DARMET**

Directeur Technique - Objet Direct / Homsys Group