3

Comment fonctionne Ajax ?

Ajax, un amalgame de technologies

Des ingrédients déjà opérationnels

Contrairement à ce que l'on pourrait croire, Ajax n'est pas une technologie spécifique et innovante mais une conjonction de plusieurs technologies anciennes. Ainsi, les applications Ajax utilisent en général tout ou partie des technologies suivantes :

- Les feuilles de styles CSS qui permettent d'appliquer une mise forme au contenu d'une page XHTML.
- Le DOM qui représente la hiérarchie des élément d'une page XHTML.
- L'objet XMLHttpRequest de JavaScript qui permet d'assurer des transferts asynchrones (ou quelquefois synchrones) entre le client et le serveur.
- Les formats de données XML ou JSON utilisés pour les transferts entre le serveur et le client.
- Le langage de script client JavaScript qui permet l'interaction de ces différentes technologies.

L'intérêt pour Ajax d'utiliser ces différentes technologies est qu'elles sont déjà intégrées dans la plupart des navigateurs actuels. Elles sont donc immédiatement exploitables – même si quelques différences d'implémentation subsistent d'un navigateur à l'autre.

Ceci représente une véritable aubaine pour les développeurs lorsqu'on connaît les atouts d'Ajax ; et on comprend mieux pourquoi toujours plus de développeurs se rallient à cette technologie.

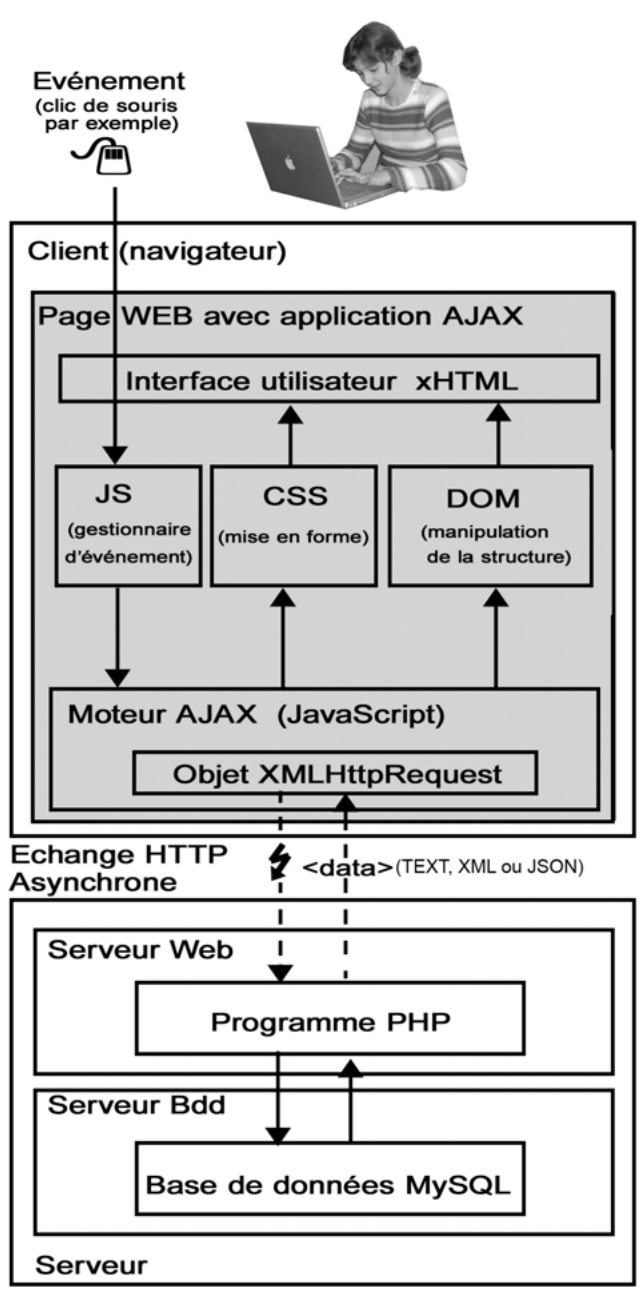
JavaScript, le ciment des fondations d'Ajax

Pour que ces différentes technologies sous-jacentes puissent être exploitées, il faut disposer d'un langage de script capable de les manipuler. Évidemment, dans ce contexte client, JavaScript est la technologie idéale pour remplir cette mission et faire interagir toutes ces technologies entre elles. Ainsi, dans chaque application Ajax, nous retrouverons un

programme JavaScript qui constituera le « moteur » du système, orchestrant à la fois les transferts de données avec l'aide de l'objet XMLHttpRequest et l'exploitation des réponses du serveur en agissant sur les CSS (pour modifier la mise en forme de la page XHTML) et sur le DOM (pour modifier le contenu ou la structure de la page XHTML) (voir figure 3-1).

Figure 3-1

Organisation des principaux composants d'Ajax



En ce qui concerne les données échangées, plusieurs formats peuvent être utilisés selon l'organisation et la complexité des flux d'informations. Les applications les plus simples

pourront se contenter de données au format texte (simples couples variable/valeur) alors que les systèmes plus complexes devront choisir de structurer leurs données en XML (le DOM assurant ensuite l'insertion des données XML dans la page XHTML) ou encore dans un format issu de la structure des objets JavaScript, le JSON. À noter que la plupart des requêtes envoyées vers le serveur utilisent le format texte (les couples variable/valeur suffisent dans la majorité des cas), mais sachez qu'elles peuvent éventuellement aussi exploiter les formats XML ou JSON, de la même manière que les résultats retournés par le serveur au navigateur.

Comparatif avec les applications Web traditionnelles

Pour bien comprendre le fonctionnement et connaître les avantages d'un nouveau système, une bonne méthode consiste à le comparer avec l'existant que l'on connaît déjà. Dans cette partie, nous allons utiliser cette méthode en comparant le fonctionnement d'une application Ajax avec celui d'un site Web statique et celui d'un site Web dynamique.

Fonctionnement d'une application Web statique

Avec un site Web statique, la seule interactivité dont dispose l'internaute est de pouvoir passer d'une page HTML à l'autre par un simple clic sur les liens hypertextes présents sur une page. À chaque fois que l'internaute clique sur un lien, une requête HTTP est envoyée, établissant du même coup une communication avec le serveur. Cette communication est de type synchrone, c'est-à-dire que dès l'émission de la requête, la communication reste en place jusqu'à la réception de la réponse du serveur. Pendant le temps de traitement de la requête, le navigateur reste figé, bloquant ainsi toute action possible de l'internaute.

À chaque requête, le serveur retournera une réponse sous la forme d'une page HTML complète. S'il s'agit d'une simple requête, suite à la saisie par l'internaute de l'URL spécifique d'une page dans la barre d'adresse du navigateur ou, plus couramment, lorsque l'internaute clique sur un lien hypertexte, le serveur se contentera de renvoyer la page HTML demandée, ce qui clôturera le traitement côté serveur et débloquera ainsi le navigateur.

Fonctionnement d'une application Web dynamique

Nous avons vu précédemment le traitement d'une simple requête par le serveur mais d'autre cas peuvent se produire, notamment lors de l'envoi d'un formulaire. Dans ce cas, la requête est constituée d'une ligne de requête (précisant la méthode utilisée et le protocole HTTP), d'un corps (qui contient les données envoyées au serveur dans le cas d'une requête émise avec la méthode POST) et d'une série d'en-têtes qui définissent les spécificités de la requête (nature du navigateur utilisé, type d'encodage…) qui permettront au serveur de traiter correctement les informations. En général, lors de l'envoi d'un formulaire, le traitement côté serveur est réalisé par une page contenant un programme (en PHP par exemple). Les données réceptionnées pouvant être traitées directement par le programme ou entraîner un échange avec un serveur de base de données afin de les mémoriser ou d'émettre une requête SQL. À l'issue de ce traitement, une nouvelle page HTML sera construite à la volée et renvoyée au navigateur, ce qui clôturera le processus, débloquant le navigateur de la même manière qu'avec un site statique.

Fonctionnement d'une application Ajax

Dans le cas d'une application Ajax, si la page contenant la structure XHTML et ses scripts client (moteur Ajax, gestionnaire d'événement…) est chargée de la même manière que pour un site statique, il n'en est pas de même pour les interactions qui suivent entre le navigateur et le serveur. Le moteur Ajax une fois chargé dans le navigateur restera en attente de l'événement pour lequel il a été programmé. Pour cela, un gestionnaire d'événement JavaScript est configuré pour appeler le moteur dès l'apparition de l'événement concerné. Lors de l'appel du moteur, un objet XMLHttpRequest est instancié puis configuré, une requête asynchrone est ensuite envoyée au serveur. À la réception de celle-ci, le serveur démarrera son traitement et retournera la réponse HTTP correspondante. Cette dernière sera prise en charge par la fonction de rappel du moteur Ajax qui exploitera les données pour les afficher à un endroit précis de l'écran.

Chronogrammes des échanges client-serveur

Une des grandes différences entre une application Web traditionnelle et une application Ajax est liée à l'échange asynchrone de données entre le navigateur et le serveur. Pour vous permettre de bien appréhender la différence entre ces deux applications, nous vous proposons de les comparer maintenant à l'aide de leur chronogramme.

Chronogramme d'une application Web dynamique traditionnelle

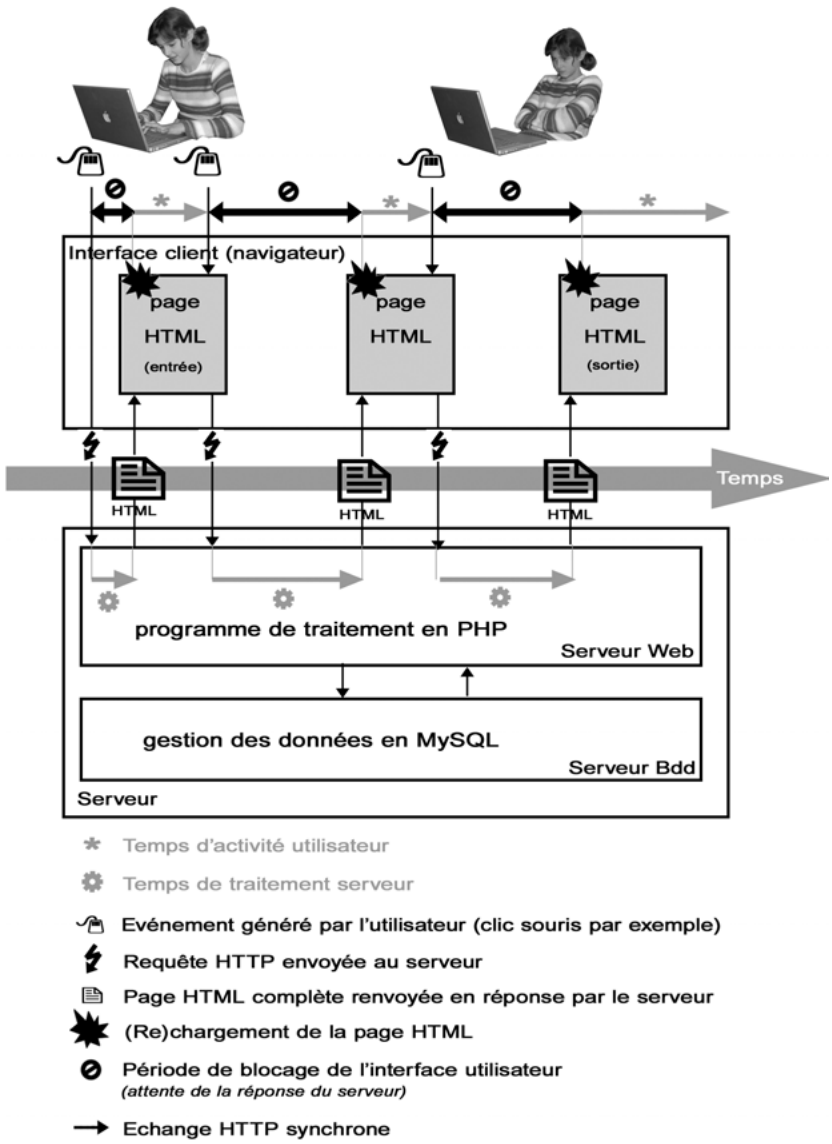
Lorsqu'un utilisateur sollicite le serveur dans une application Web dynamique traditionnelle (en envoyant un formulaire ou en cliquant sur une URL dynamique), il déclenche une requête HTTP dans laquelle sont imbriqués les paramètres de la demande. À partir de ce moment, le navigateur se fige jusqu'à la réception de la réponse HTTP du serveur, interdisant ainsi à l'utilisateur toute action pendant le temps de traitement de la requête. Dès la réception de la requête, le serveur Web analysera les paramètres et traitera la demande selon son programme. Il pourra interroger un serveur de base de données pour recueillir des données complémentaires si nécessaire. Une fois le traitement terminé, une page HTML complète sera construite à la volée, incluant les résultats du traitement après leur mise en forme. Cette page sera alors retournée au navigateur après son intégration dans le corps de la réponse HTTP. À la réception de la réponse HTTP, le navigateur interprétera la page HTML, comme lors de l'appel d'une page Web dans un site statique, et l'affichera à l'écran, entraînant le rechargement complet de la page. À la fin du chargement de la page, le navigateur est débloqué et l'utilisateur reprend la main sur l'application. Il pourra ainsi éventuellement réitérer une nouvelle demande serveur qui suivra le même cycle de traitement que celui que nous venons de décrire (voir figure 3.2).

Chronogramme d'une application Ajax en mode asynchrone

Dans le cas d'une application Ajax en mode asynchrone, le déroulement du traitement est différent. À noter que l'objet XMLHttpRequest peut aussi envoyer des requêtes synchrones, mais dans ce cas le fonctionnement serait semblable à celui d'une application Web dynamique traditionnelle comme celle que nous avons décrite précédemment.

Figure 3-2

Chronogramme des échanges client-serveur d'une application traditionnelle

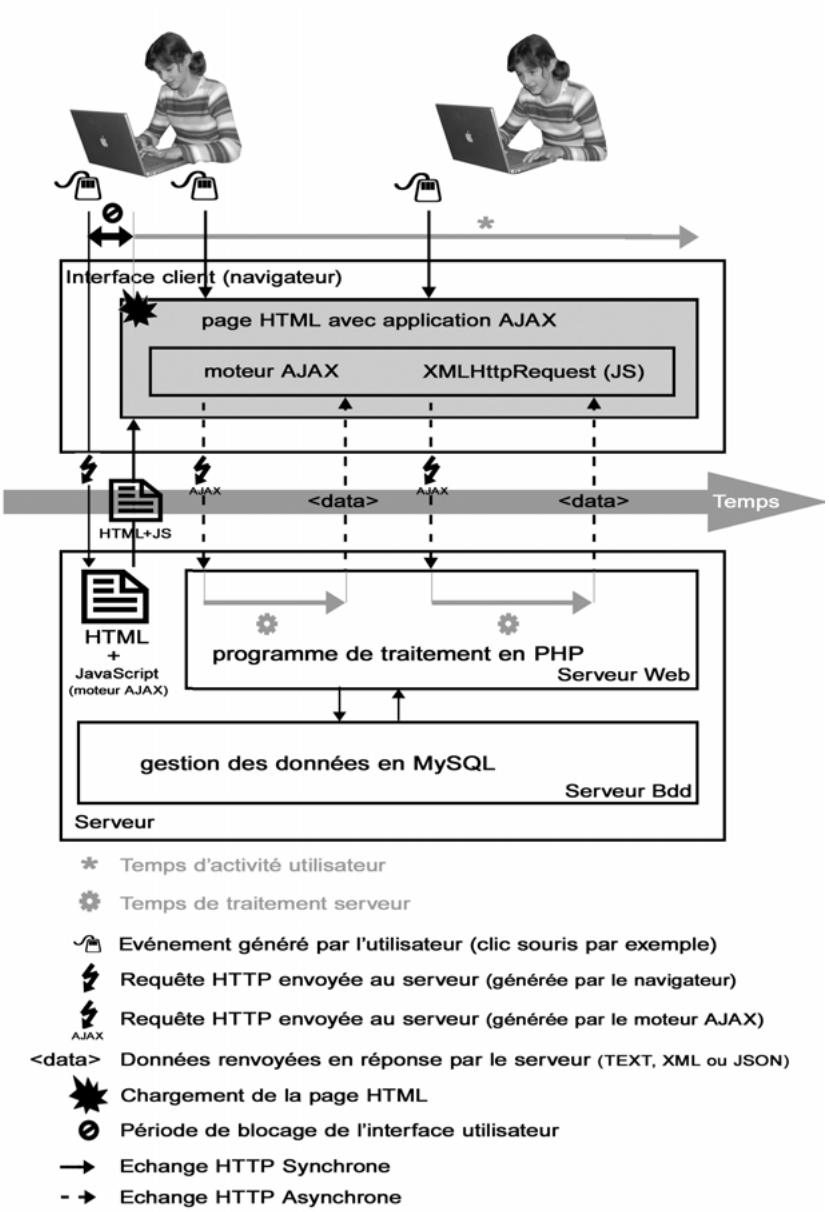


Dans une application Ajax, l'utilisateur doit commencer par appeler la page HTML contenant le moteur Ajax. Une fois la page chargée dans le navigateur, les échanges avec le serveur seront contrôlés par l'application Ajax (voir figure 3-3). L'envoi d'une requête est souvent déclenché par un gestionnaire d'événement JavaScript, mais il peut aussi être généré par un script de temporisation pour actualiser des informations à intervalles réguliers. Quel que soit le mode de déclenchement, le moteur Ajax est appelé par le biais d'une fonction JavaScript. La première action du moteur est la création d'un objet XMLHttpRequest immédiatement suivi de sa configuration (choix de la méthode de transfert GET ou POST, choix du fichier serveur sollicité, activation du mode asynchrone, désignation d'une fonction de rappel, intégration des paramètres…). Une fois l'objet configuré, l'envoi de la requête est déclenché, générant une requête HTTP semblable à celle créée avec une application dynamique traditionnelle. Toutefois, dans le

cas de l'envoi d'une requête Ajax, le navigateur n'est pas bloqué et l'utilisateur peut continuer à utiliser son interface comme bon lui semble ; le transfert est asynchrone. Côté serveur, les paramètres seront analysés et le programme pourra aussi solliciter un serveur de base de données si besoin. Mais, contrairement à une application dynamique traditionnelle, le corps de la réponse HTTP retournée au navigateur ne sera pas composé de la page HTML complète : il contiendra seulement les données réclamées par le client. Lorsque le navigateur reçoit la réponse, une fonction de rappel, programmée lors de l'envoi de la requête, se chargera de récupérer les données placées dans le corps de la réponse HTTP, de les mettre en forme et de les insérer dans une zone particulière de la page Web et cela sans nécessiter le rechargement de la page (voir figure 3-3).

Figure 3-3

Chronogramme des échanges client-serveur d'une application Ajax



Les avantages d'Ajax

Économie de la bande passante

Avec Ajax, il n'est plus nécessaire de renvoyer le contenu entier de la page HTML à chaque requête, car l'objet XMLHttpRequest assure la récupération et l'insertion dans la page en cours des seules données à modifier. Ce système permet d'éliminer le transfert de nombreuses informations redondantes, allégeant ainsi fortement le trafic réseau entre le serveur Web et le client (navigateur).

Empêche le rechargement de la page à chaque requête

Le traitement traditionnel d'une requête HTTP entraîne à chaque retour de la réponse du serveur un rechargement complet de la page en cours. Hormis le désagréable « trou blanc » que cela engendre, ce phénomène allonge le temps de traitement d'une requête aux dépens de la réactivité de l'application.

Évite le blocage de l'application pendant le traitement de la requête

Contrairement au simple échange HTTP d'une application traditionnelle, dans laquelle l'application cliente est bloquée pendant tout le temps d'attente de la réponse du serveur, l'échange XMLHttpRequest asynchrone d'une application Ajax permet à l'internaute de continuer à travailler pendant le temps de traitement de la requête. Cela ouvre des possibilités nouvelles pour le développement Web, permettant ainsi aux développeurs de créer des applications dont le mode de fonctionnement se rapproche de celui des applications disponibles jusqu'alors sur des ordinateurs de bureau.

Augmente la réactivité de l'application

Les données renvoyées par le serveur étant plus légères (le serveur retournant uniquement les données demandées et non la page HTML entière) et le rechargement de la page complète n'ayant plus lieu à chaque requête, cela améliore considérablement la réactivité du système. De plus, le chargement progressif des données couplé à une méthode prédictive permet de disposer de fonctionnalités graphiques avancées (déplacement d'une carte à l'aide de la souris dans une application de cartographie en ligne par exemple) jusqu'alors réservées aux logiciels autonomes de bureau.

Améliore l'ergonomie de l'interface

Une interface Ajax peut être composée de multiples zones ayant une gestion du contenu indépendante l'une de l'autre. Chaque zone pouvant déclencher ses propres requêtes, il est désormais possible d'avoir une mise à jour ciblée des contenus. Ainsi, grâce aux technologies DHTML associées à Ajax, l'utilisateur peut aménager librement ses différentes zones par un simple glisser-déposer et améliorer l'ergonomie de son interface Web.

Les inconvénients d'Ajax

Pas de mémorisation des actions dans l'historique

Le principal inconvénient d'une application Ajax est lié au fait que les actions de l'utilisateur ne sont pas mémorisées dans l'historique du navigateur. En effet, les différents

contenus d'une application Ajax s'affichant toujours dans la même page, ils ne peuvent pas être enregistrés dans l'historique du navigateur comme le seraient les différentes pages HTML d'une application Web traditionnelle.

Par voie de conséquence, les boutons Suivant et Précédent ne sont plus utilisables car ils s'appuient sur l'historique du navigateur pour trouver la page suivante ou précédente. Ceci est évidemment très handicapant pour les internautes qui ont l'habitude d'utiliser ces boutons pour naviguer d'une page à l'autre.

Il existe néanmoins des solutions pour remédier à ce problème en couplant l'application Ajax avec un système d'iframe comme le fait Google dans plusieurs de ses applications Ajax mais cela nécessite un traitement supplémentaire qui complexifie le développement.

Problème d'indexation des contenus

Les différents contenus d'une application Ajax s'affichant dans une seule et même page, les moteurs de recherche pourront indexer uniquement le premier contenu par défaut de la page et non tous les contenus proposés par l'application.

D'autre part, le rappel des différents contenus d'une application Ajax par le biais des favoris sera confronté au même problème. Seul le contenu de la première page pourra être mémorisé dans les signets du navigateur.

Dépendance de l'activation de JavaScript sur le navigateur

Les applications Ajax utilisant JavaScript pour interagir entre les différentes technologies exploitées côté client (CSS, DOM, XML…) sont donc dépendantes de l'activation de JavaScript sur le navigateur, au même titre que tous les autres programmes clients utilisant cette technologie.

Même si les internautes qui désactivent JavaScript se raréfient, il faut toutefois prévoir une version dégradée de l'application en prévision des navigateurs qui ne supporteraient pas ce langage de script.

Les cadres cachés, une solution alternative à Ajax

Dans le chapitre précédent, nous avons cité d'autres technologies estampillées Web 2.0 (Flash + Flex, application Java) permettant la mise en œuvre d'une application Internet riche (RIA). Nous avions cependant écarté ces solutions car elles ne pouvaient pas fonctionner sur un navigateur sans l'installation d'un plug-in.

Il existe néanmoins une technique nommée « cadre caché » (frameset HTML ou iframe) utilisée bien avant celle de l'objet XMLHttpRequest qui permet d'établir des communications en arrière plan avec le serveur et qui, comme Ajax, ne nécessite pas l'ajout d'un plug-in.

La technique du cadre caché

Cette technique exploite la structure des jeux de cadres HTML dont l'un d'entre eux est invisible et sert de pont pour établir une communication avec le serveur. Le cadre caché est rendu invisible en configurant sa largeur et sa hauteur à zéro pixel. Avec cette technique,

il est alors possible d'envoyer des requêtes serveur par le biais du cadre caché sans perturber l'écran de l'utilisateur.

Pour illustrer le fonctionnement de cette technique, nous allons détailler le cycle d'une communication complète entre le navigateur et le serveur. Pour commencer, l'utilisateur déclenche une fonction JavaScript depuis le cadre visible. Cette fonction appellera un script serveur dont le retour sera assigné au cadre caché. Le script serveur analyse alors les paramètres communiqués et traite la demande. Il renvoie ensuite en réponse au cadre caché une page HTML complète contenant le résultat dans une balise `<div>`. Dans cette même page HTML se trouve une fonction JavaScript qui sera invoquée dès que la page sera complètement chargée dans le cadre caché (avec le gestionnaire d'événement `window.onload` par exemple). Enfin, lorsque la fonction JavaScript s'exécute dans le cadre caché, elle récupère le résultat inséré préalablement dans la balise `<div>` de la même page et l'affecte à une zone définie du cadre visible. L'utilisateur peut alors voir la réponse apparaître dans la page visible du navigateur et cela tout en continuant d'utiliser l'interface pendant le traitement serveur évitant ainsi que la page ne soit rechargée.

Depuis l'apparition des iframes (introduites dans la version 4.0 du HTML), il est possible d'exploiter la même technique mais sans avoir à utiliser la structure contraignante des framesets. En effet, l'iframe peut être placé dans une page HTML traditionnelle et permet de créer ainsi un cadre dans n'importe quelle page existante. Il est même possible de créer des iframes à l'aide d'un programme JavaScript, ce qui permet de mieux contrôler la création et la suppression des flux de communication entre le serveur et le navigateur.

Avantages des cadres cachés

Fonctionne sur les anciens navigateurs

Cette technique étant pratiquée depuis longtemps, elle peut être utilisée sur des navigateurs plus anciens qui ne supportaient pas encore les objets XMLHttpRequest. Il est donc possible d'utiliser la technique des cadres cachés en tant que solution alternative à Ajax si l'on désire que l'application fonctionne sur une plus grande variété de navigateurs.

Conserve l'historique du navigateur

La technique des cadres cachés permet de conserver l'historique du navigateur. Cette caractéristique permet aux internautes de continuer à utiliser les boutons Suivant et Précédent du navigateur contrairement aux applications Ajax. À noter que certaines applications couplent Ajax à la technique des cadres cachés pour remédier au problème des boutons Suivant et Précédent inactifs (comme Gmail et Google Maps par exemple).

Inconvénients des cadres cachés

Manque d'informations sur le traitement de la requête

Le principal inconvénient de la technique des cadres cachés est lié au manque d'informations concernant le traitement de la requête HTTP en arrière-plan. Cela pose de gros problèmes dans le cas où la page du cadre caché n'est pas chargée, car l'internaute peut attendre indéfiniment la réponse sans être informé de l'incident. Même s'il est possible de programmer une temporisation pour interrompre le traitement et informer l'utilisateur au bout d'un temps déterminé, il est préférable désormais d'utiliser un objet XMLHttpRequest qui nous permet de garder le contrôle de toutes les étapes du traitement de la requête HTTP.