8

Applications Ajax-PHP synchrones

Pour commencer simplement, je vous propose une série d'ateliers qui vous permettra de créer progressivement une première application synchrone.

Pour illustrer son fonctionnement nous réaliserons une petite application qui simulera le fonctionnement d'une machine à sous en ligne. Côté client, l'application sera constituée d'une page HTML dans laquelle nous construirons progressivement un moteur Ajax dont l'objectif sera de simuler le fonctionnement d'une machine à sous. Pour cela l'utilisateur devra appuyer sur un bouton pour déclencher une requête depuis son navigateur et relancer ainsi la machine à sous. Côté serveur, un fichier PHP réceptionnera et traitera la requête client puis renverra le montant du gain en retour qui s'affichera ensuite dans la page Web.

Atelier 8-1 : requête synchrone sur un fichier texte sans feuille de styles

Composition du système

Nous allons commencer par mettre en place une structure minimaliste pour tester le fonctionnement d'une première requête synchrone sur un simple fichier texte (voir figure 8-1). Cette première structure est composée :

- d'une page HTML (`index.html`) dans laquelle nous allons intégrer un bouton de formulaire pour déclencher le jeu, une zone d'affichage du résultat et le JavaScript du moteur Ajax ;
- d'un simple fichier texte (`gainMax.txt`) dans lequel nous allons saisir la valeur du gain maximum, soit 100.

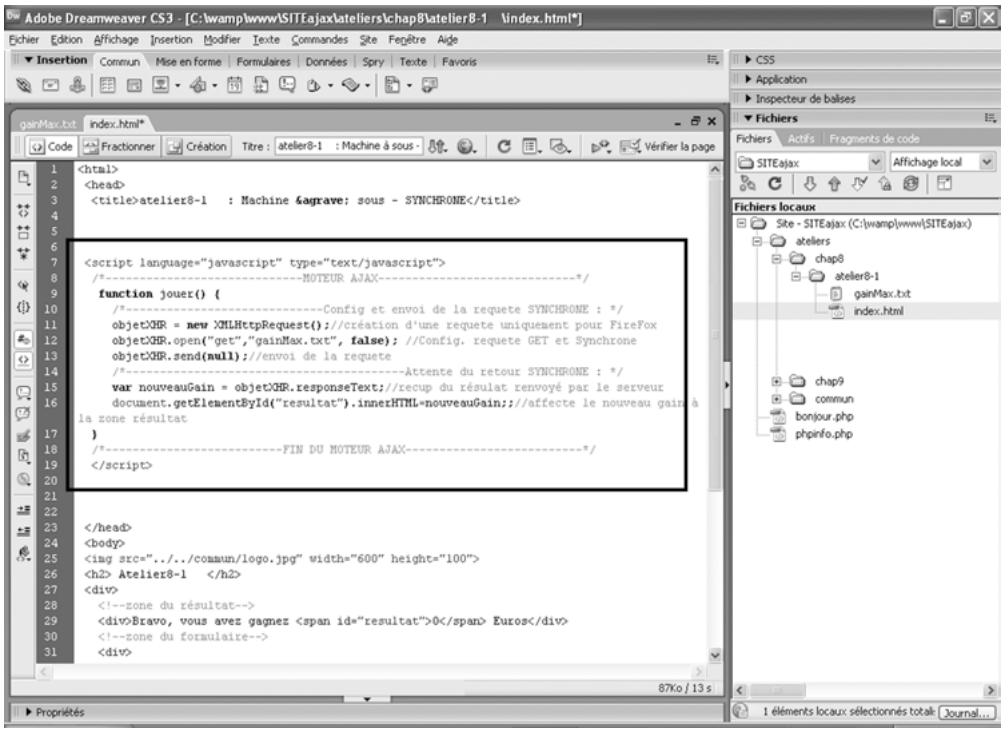

Figure 8-1

Saisie du moteur Ajax dans la balise <head> de la page

> **Prérequis concernant l'objet XMLHttpRequest (XHR)**
>
> Comme vous avez déjà découvert le fonctionnement de l'objet `XMLHttpRequest` dans le chapitre 4 de ce même ouvrage, nous nous appuierons sur ce prérequis dans la rédaction des ateliers de ce chapitre. Si toutefois certaines méthodes ou propriétés de cet objet vous semblent inconnues, n'hésitez pas à vous reporter au chapitre 4 pour revoir son concept.
>
> À noter aussi que nous aurons souvent l'occasion de faire référence à l'objet `XMLHttpRequest` dans cet ouvrage et pour alléger l'écriture, nous utiliserons fréquemment son appellation abrégée XHR.

Fonctionnement du système

Le bouton JOUER de la page permettra à l'utilisateur d'afficher la valeur maximum des gains. Ce bouton sera relié à un gestionnaire d'événement `onclick` qui appellera une fonction `jouer()` (voir code 8-1, à noter que pour cette première application le gestionnaire sera intégré dans la balise HTML `<input>` mais nous verrons par la suite qu'il est préférable de déclarer les gestionnaires d'événements directement dans le code JavaScript afin de bien séparer le programme de la structure de la page).

Cette fonction déclenchera le moteur Ajax (voir code 8-2) qui créera un objet XMLHttpRequest (par la suite, nous utiliserons son appellation abrégée XHR) puis le configurera (à l'aide de la méthode `open()` de l'objet : choix de la méthode `GET` et ciblage du fichier `gainMax.txt` en mode Synchrone) avant d'envoyer la requête (à l'aide de la méthode `send()` de l'objet).

L'information contenue dans le fichier texte `gainMax.txt` (soit la valeur 100) sera ensuite retournée au navigateur dans le corps de la réponse. Cette valeur sera enregistrée dans la variable `nouveauGain` par le biais de la propriété `responseText` de l'objet puis affectée à la zone de résultat à l'aide de la propriété `innerHTML` de l'élément résultat.

Conception du système

Téléchargement des codes sources des ateliers

Le moment est venu de passer à l'action. Les explications des différents ateliers vous permettront de créer vos différents scripts à partir de zéro. Cependant, vous avez la possibilité de télécharger tous les fichiers des exemples de cet ouvrage à votre disposition dans l'extension du livre sur le site *www.editions-eyrolles.com* (utilisez le nom de l'auteur comme mot clé pour accéder à l'extension de cet ouvrage).

Vous pourrez ainsi vous reporter à ces fichiers pour les comparer avec les vôtres en cas de problème, ou encore tester directement le fonctionnement de tous les ateliers directement sur ces ressources si vous ne désirez pas saisir vous-même les codes.

Ouvrez Dreamweaver (ou l'éditeur de votre choix) et sélectionnez le site Ajax que nous avons créé lors de l'installation de l'environnement de développement. Créez un nouveau fichier HTML et enregistrez-le tout de suite dans le répertoire `/ateliers/chap8/atelier8-1/` en le nommant `index.html`.

Organisation de vos ateliers

Nous vous suggérons de créer chaque atelier dans un répertoire différent portant le nom de l'atelier afin de bien isoler les fichiers utilisés dans le cadre de nos essais. Les deux fichiers (`index.html` et `gainMax.txt`) de ce premier atelier seront donc enregistrés dans un répertoire nommé « atelier8-1 ». Ainsi chaque atelier sera indépendant de l'autre, le seul élément qui ne sera pas dans le répertoire est l'image `logo.jpg` placée en haut de chaque page `index.html`, cette dernière étant commune à tous les ateliers, nous l'avons placé dans un répertoire nommé `/commun/`.

Dans ce premier exemple, nous avons choisi de ne pas lier la page HTML à une feuille de styles par souci de simplicité pour la mise en œuvre de votre première application. Néanmoins, nous allons structurer les différentes zones de la page HTML avec des balises `<div>` en prévision de la future feuille de styles que nous allons appliquer ensuite à cette page. Les deux éléments de la page (la zone `<span>` du résultat et le formulaire contenant le bouton JOUER) sont tous les deux insérés dans des balises `<div>` différentes et l'ensemble est regroupé dans une troisième balise `<div>` qui servira de conteneur pour la page (voir code 8-1).

Code 8-1 :

```
<div>
  <!--zone du résultat-->
  <div>
  Bravo, vous avez gagné <span id="resultat">0</span> euros
  </div>
  <!--zone du formulaire-->
  <div>
  <form>
   <input name="button" type="button" onclick="jouer();" value="JOUER" />
  </form>
  </div>
 </div>
```

Ressources sur les technologies associées

Nous avons regroupé dans la partie 4 de cet ouvrage plusieurs chapitres sur chacune des technologies utilisées dans les applications des ateliers. Nous vous invitons à vous y reporter pour obtenir des compléments d'informations si les explications qui accompagnent ces ateliers ne vous suffisent pas.

Placez-vous ensuite dans la balise `<head>` de la page et saisissez le code 8-2 ci-dessous.

Code 8-2 :

```
<script language="javascript" type="text/javascript">
/*-----------------------------MOTEUR AJAX-----------------------*/
  function jouer() {
 /*-------------------Config et envoi de la requête SYNCHRONE : */
  //création d'une requête uniquement pour Firefox
  objetXHR = new XMLHttpRequest();
  //Config. requête GET et Synchrone
  objetXHR.open("get","gainMax.txt", false);
  //envoi de la requête
  objetXHR.send(null);
/*--------------------------------Attente du retour SYNCHRONE : */
  //récupération du résultat renvoyé par le serveur
  var nouveauGain = objetXHR.responseText;
  //Affecte le nouveau gain à la zone résultat
  document.getElementById("resultat").innerHTML=nouveauGain;
 }
/*--------------------------FIN DU MOTEUR AJAX-------------------*/
 </script>
```

Enregistrez le fichier `index.html` après avoir terminé la saisie puis ouvrez un fichier texte (Depuis le menu de Dreamweaver : Fichier>Nouveau cliquez sur le bouton Autres à gauche de la fenêtre puis sélectionnez le type Texte dans la liste). Saisissez la valeur 100 dans le contenu du fichier et enregistrez-le sous le nom `gainMax.txt` (voir figure 8-2).

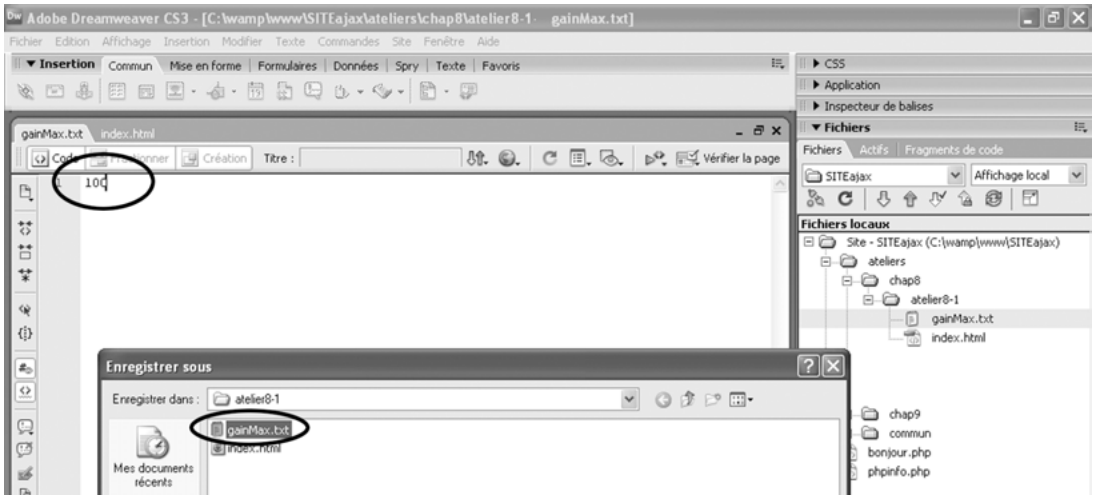

Figure 8-2

Création du fichier texte gainMax.txt

Pour ce premier script, il est intéressant d'expliquer d'une façon détaillée le code de cette page HTML afin de bien comprendre le mécanisme de cette application Ajax.

Dans la partie visible de page HTML (balise `<body>`, voir le code 8-1), la zone dans laquelle s'affichera le résultat est délimitée par une balise `<span>` à laquelle nous avons ajouté un identifiant `resultat` qui permettra ensuite d'en modifier le contenu à l'aide de la propriété `innerHTML` de l'élément ainsi constitué. Lors de son premier affichage, le contenu de cette balise est initialisé avec la valeur 0. Cette valeur sera ensuite remplacée par la valeur contenue dans le fichier texte (100).

```
<span id="resultat">0</span>
```

Un peu plus bas, un formulaire a été ajouté afin d'insérer un bouton de commande JOUER dans la page HTML. L'élément `<input>` est lié à un gestionnaire d'événement qui permettra d'appeler la fonction contenant le moteur Ajax (`onclick="jouer()"`). L'utilisateur pourra ainsi afficher le contenu retourné par le fichier texte par un simple clic sur ce bouton. À noter que la valeur du fichier texte étant toujours la même, il est nécessaire d'appuyer sur le bouton d'actualisation du navigateur (ou d'utiliser le raccourci clavier F5) pour revenir à l'état initial de la page avant d'appuyer de nouveau sur le bouton JOUER.

```
<form>
 <input type="button" onclick="jouer();" value="JOUER" />
</form>
```

Passons maintenant à la fonction `jouer()` contenant le moteur Ajax. La première instruction de cette fonction permet d'instancier la classe XMLHttpRequest et de créer un objet nommé `objetXHR`.

```
objetXHR = new XMLHttpRequest();
```

Une fois l'objet créé, il faut ensuite le configurer avec sa méthode `open()`. Trois paramètres seront nécessaires à sa configuration. Le premier permet d'indiquer que nous désirons utiliser la méthode `GET` pour émettre la requête HTTP. Le second précise le fichier ciblé par la requête, soit le fichier texte `gainMax.txt` pour ce premier exemple. Enfin le troisième paramètre est initialisé avec la valeur `false` afin d'indiquer que la requête devra être en mode synchrone.

```
objetXHR.open("get","gainMax.txt", false);
```

Maintenant que l'objet est créé et configuré, il ne reste plus qu'à l'envoyer. Pour cela, nous utiliserons la méthode `send()` de l'objet. À noter que l'argument de cette méthode sera utilisé lorsque nous aurons une méthode `POST` avec des paramètres à communiquer au serveur. Comme ce n'est pas le cas de notre exemple, ce paramètre sera configuré avec la valeur `null`.

```
objetXHR.send(null);
```

La requête étant synchrone, la communication restera ouverte dans l'attente de la réponse comme dans le cas d'une requête HTTP traditionnelle. Nous pouvons donc placer les instructions de traitement de la réponse immédiatement après l'envoi de la requête. La première instruction de ce traitement permet d'affecter la réponse texte à une variable nommée `nouveauGain`. Nous utilisons pour cela la propriété `responseText` de l'objet XHR.

```
var nouveauGain = objetXHR.responseText;
```

Nous arrivons maintenant au terme de la fonction du moteur Ajax. En effet, nous disposons de la valeur de la réponse côté client, il ne nous reste plus qu'à l'affecter à la zone

résultat afin qu'elle remplace la valeur 0 configurée par défaut. Pour cela, nous utiliserons la méthode `getElementById()` qui permet de référencer l'élément de la balise `<span>` par son identifiant `resultat`. Puis nous exploiterons la propriété `innerHTML` qui permettra de remplacer le contenu de l'élément par la valeur 100 sauvegardée précédemment dans la variable `nouveauGain`.

```
document.getElementById("resultat").innerHTML=nouveauGain;
```

Test du système

Pour tester le système, vous devez commencer par ouvrir la page `index.html` dans le Web Local avec le navigateur Firefox. Pour cela, plusieurs solutions s'offrent à vous. La première est la plus rapide mais nécessite d'avoir configuré le serveur d'évaluation dans la définition initiale du site Ajax (revoir si besoin le chapitre 7). Assurez-vous avant tout que Wamp5 est bien démarré (un icône en forme de demi cercle doit apparaître dans la zone d'état en bas à droite de l'écran de votre ordinateur).

Ouvrez la page à tester (`index.html` dans notre cas) dans Dreamweaver puis cliquez sur l'icône Aperçu/débogage (bouton ayant la forme d'une planète bleue) situé dans le menu de l'éditeur de fichier puis sélectionnez Firefox dans la liste des navigateurs (par la suite, nous vous conseillons d'utiliser le raccourci clavier F12). Le navigateur Firefox doit alors s'ouvrir et afficher la page concernée.

L'autre solution est plus longue mais pourra être utilisée dans tous les cas, même si vous n'utilisez pas Dreamweaver ou si le serveur d'évaluation n'a pas encore été configuré. Déroulez le menu du manager Wamp et sélectionnez l'option `localhost`. La page d'accueil du Web Local de Wamp doit alors s'ouvrir dans le navigateur Firefox (préalablement configuré comme le navigateur par défaut dans la procédure d'installation de Wamp5). Dans la zone Vos projets cliquez sur le petit répertoire nommé `SITEajax`. La racine de notre site Ajax n'ayant pas de fichier d'index, la liste des fichiers et répertoires s'affiche dans la nouvelle page. Cliquez successivement sur les répertoires `atelier`, `chap8` puis `atelier8-1`. La page `index.html` doit alors s'ouvrir comme dans le cas de la première solution.

Fonctionne uniquement sur Firefox

L'exemple de cet atelier fonctionne uniquement sur le navigateur Firefox (ou les autres navigateurs compatibles avec l'objet XMLHttpRequest). Si toutefois vous désirez le faire fonctionner dès maintenant avec Internet Explorer (version supérieure à 5.0), il suffit de remplacer la syntaxe d'instanciation de l'objet XHR, soit actuellement `new XMLHttpRequest()`, par celle-ci : `new ActiveXObject("MSXML2.XMLHttp")`. Soyez rassuré, nous présenterons plus loin le script à utiliser pour que vos applications puissent fonctionner avec tous les types de navigateur.

Nous pouvons remarquer dans la page `index.html` qui est maintenant affichée dans le navigateur que la valeur du gain est égale à 0. Si vous cliquez sur le bouton JOUER, vous déclenchez alors la requête synchrone et la valeur du gain doit être immédiatement remplacée par celle du gain maximum stockée dans le fichier `gainMax.txt` (soit 100).

L'intérêt de ces ateliers est surtout d'observer le fonctionnement du système afin de mieux comprendre les rouages du mécanisme d'une application Ajax et pour vous permettre par la suite de diagnostiquer un éventuel problème et dépanner une application plus complexe. Pour ces raisons nous allons activer l'extension Firebug à chaque test en cliquant sur l'icône placé en bas et à droite du navigateur.

Comme nous l'avons déjà vu dans le chapitre consacré à Firefox et à ses extensions, Firebug permet d'effectuer de multiples opérations lors du test d'une application Ajax. Par exemple, si vous cliquez sur l'onglet HTML (voir repère 3 de la figure 8-3), vous pouvez afficher le contenu des éléments en mémoire en déroulant successivement les différents éléments de la page. Attention, il s'agit des contenus en mémoire (représentation du DOM sous forme de balises) et non d'un simple affichage du code initial de la page. Ainsi, dans notre cas, après l'envoi de la requête, la valeur dans la zone de résultat est égale à 100 (voir repère 2 de la figure 8-3) et non à 0 (valeur qui serait visible dans cette même fenêtre avant l'action sur le bouton JOUER ou dans le cas de l'affichage du code source traditionnel d'une page). De plus si vous survolez avec votre souris l'un de ces éléments dans la fenêtre de Firebug, la zone correspondante dans l'écran du navigateur est alors mise en évidence (voir repère 1 de la figure 8-3).

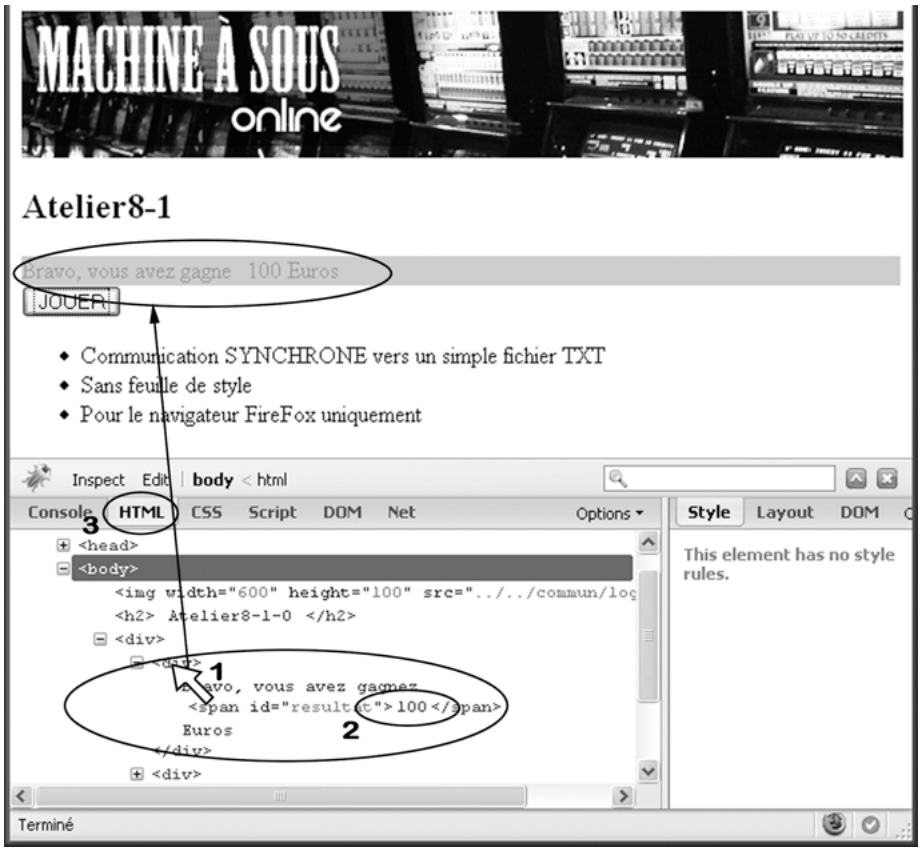

Figure 8-3

Affichage du contenu des éléments de la page HTML en mémoire à l'aide Firebug

Une autre fonctionnalité très intéressante de Firebug est de pouvoir observer les informations échangées entre le navigateur et le serveur. Vous pouvez ainsi faire apparaître tous les objets externes qui constituent votre page HTML (structure brute de la page HTML, images, feuille CSS, fichier JS externe…) lors de son appel initial et connaître le temps de chargement de chacun des objets individuellement. Mais cela est encore plus intéressant avec une application Ajax lors de l'envoi d'une requête car, si vous pouvez aussi

connaître le temps de traitement de la requête, vous pouvez surtout afficher le contenu de la requête HTTP (et de tous ses en-têtes) ainsi que le contenu de la réponse HTTP correspondante. Pour cela, il faut cliquer sur l'onglet Net de Firebug (voir repère 1 de la figure 8-4) puis sur le bouton Clear (voir repère 2 de la figure 8-4) pour nettoyer les chargements précédents. Pour simuler le chargement initial de la page, nous allons cliquer sur le bouton de réactualisation de la page (voir repère 3 de la figure 8-4, ou plus simplement à l'aide du raccourci F5). Vous devriez voir apparaître les deux objets constituant la page Web de notre exemple avec leurs temps de chargement respectifs, soient la structure brute de la page HTML et le chargement de l'image placée en tête de notre page (voir repère 4 de la figure 8-4).

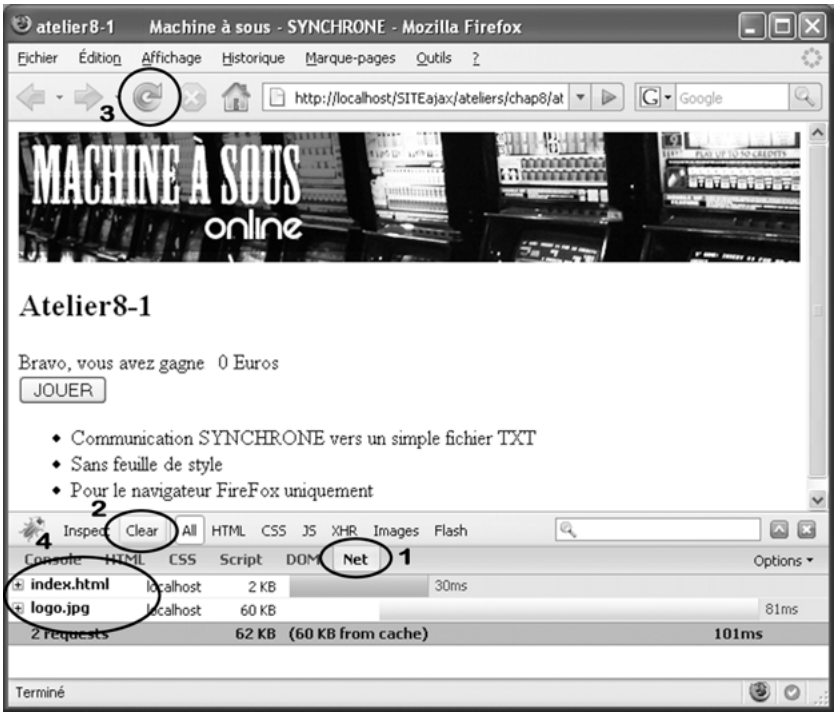

Figure 8-4

Affichage des temps de chargement des objets constituant une page Web à l'aide de Firebug

Pour l'instant, nous n'avons pas encore déclenché la requête Ajax. Si vous cliquez maintenant sur le bouton JOUER (voir repère 1 de la figure 8-5), le fichier texte doit alors être chargé dans le navigateur et apparaître à la suite de la liste des deux précédents objets de la page Web (voir repère 3 de la figure 8-5) et le résultat affiché dans la page doit prendre la valeur 100 (voir repère 2 de la figure 8-5).

Comme nous vous l'avons annoncé précédemment, Firebug permet aussi d'afficher le contenu de tous les en-têtes et le corps d'une réponse HTTP. Pour cela, cliquez sur le nom du fichier texte chargé lors de la requête HTTP (`gainMax.txt`, voir repère 1 de la figure 8-6). Une nouvelle zone doit alors apparaître en dessous du nom du fichier (utilisez votre souris pour redimensionner la fenêtre si besoin). Par défaut, c'est l'onglet des en-têtes qui doit être actif (`Headers`, voir repère 2 de la figure 8-6) mais si vous cliquez sur le second