Exercice3 : AJAX

AJAX n'est ni une technologie ni un langage de programmation ; AJAX est un concept de programmation Web reposant sur plusieurs technologies comme le JavaScript et le XML – d'où le nom AJAX. Créer un projet Exercice3 dans le même Workspace.

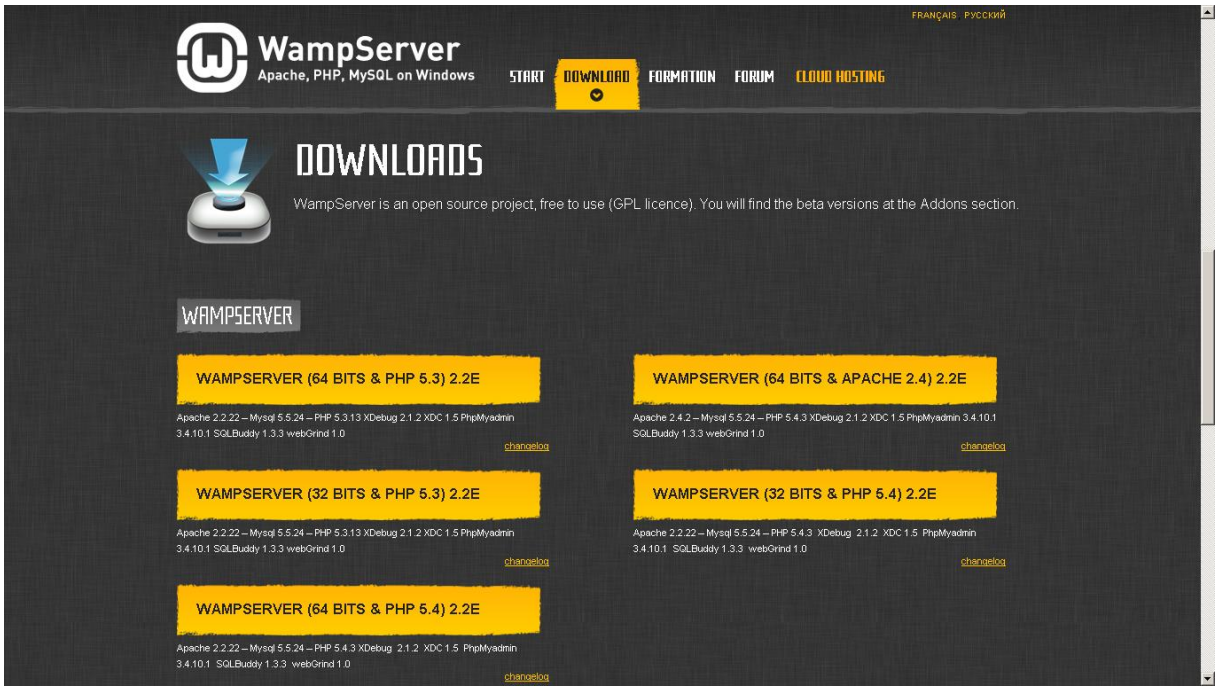
L'idée même d'AJAX est de faire communiquer une page Web avec un serveur Web sans occasionner le rechargement de la page. C'est la raison pour laquelle JavaScript est utilisé, car c'est lui qui va se charger d'établir la connexion entre la page Web et le serveur.

Etape1 : installation

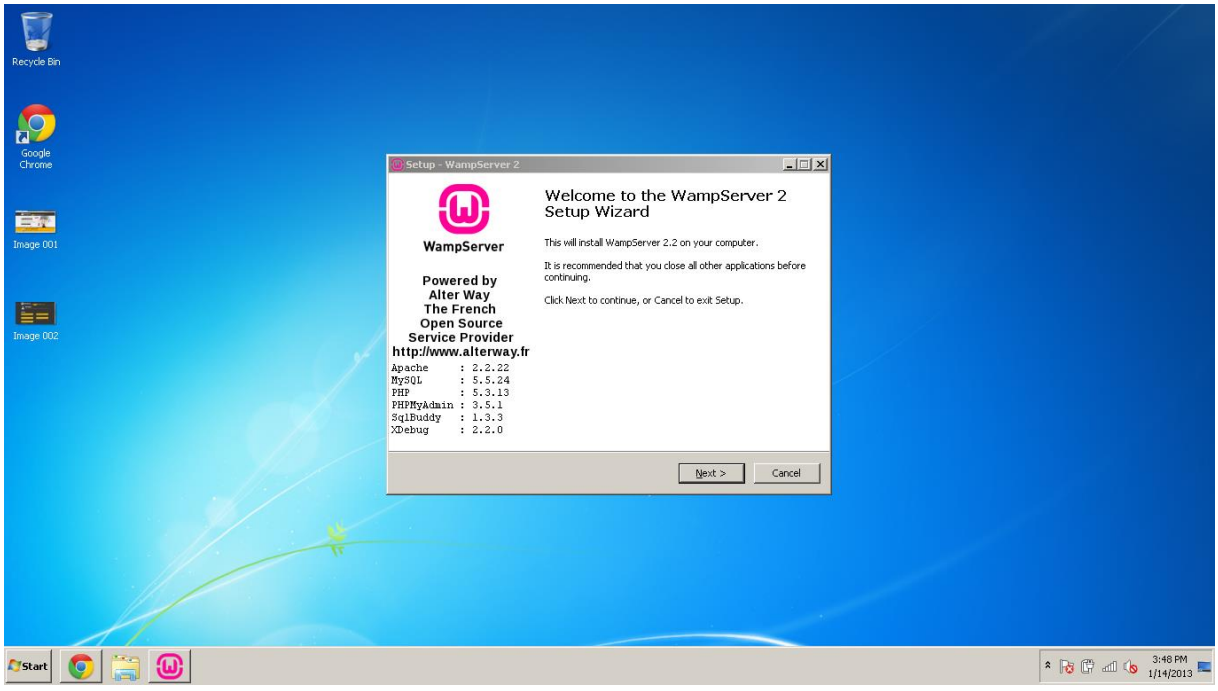
PHP est un langage qui fournit une large gamme de fonctionnalités à des sites Web. Malheureusement, PHP n'est pas comme le HTML et Javascript et vous ne pouvez pas tester facilement dans un navigateur.

Le serveur que nous utilisons est appelé serveur WAMP. Il est disponible pour Windows. Pour le télécharger, rendez-vous à: www.wampserver.com/en/.

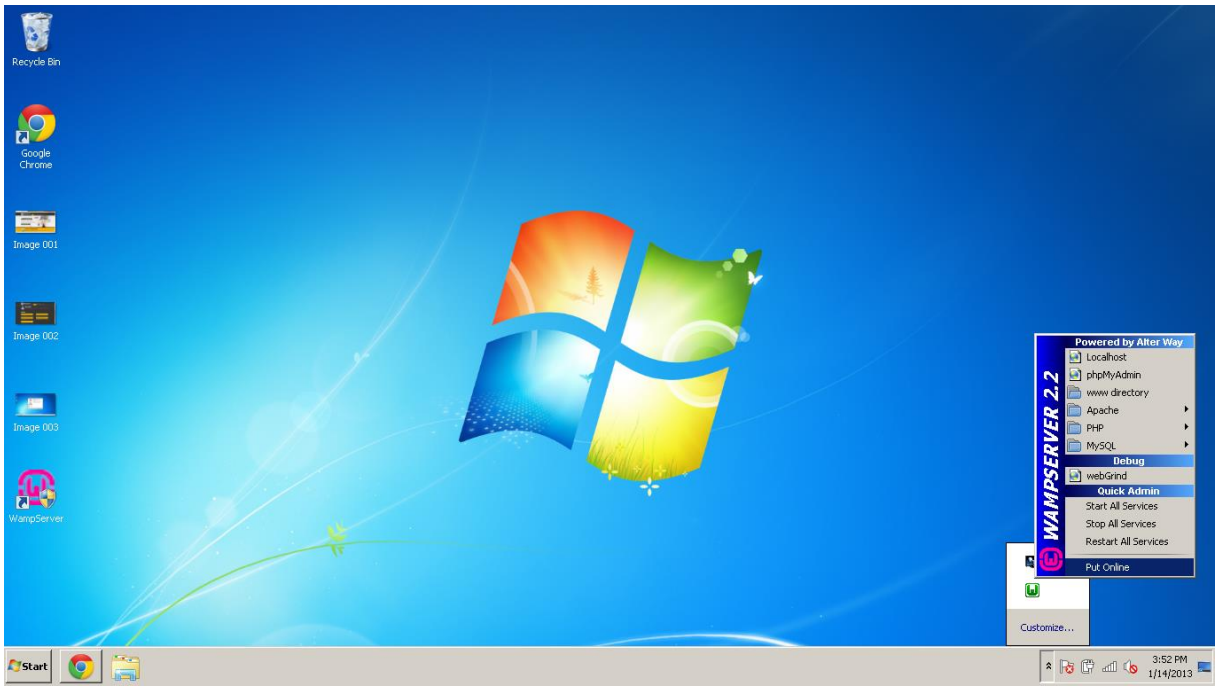
Une fois sur ce site, faites défiler vers le bas et sélectionnez l'installation correcte pour votre machine Windows.



Une fois le logiciel téléchargé, double-cliquez dessus pour l'exécuter. Passer par le processus d'installation entier.

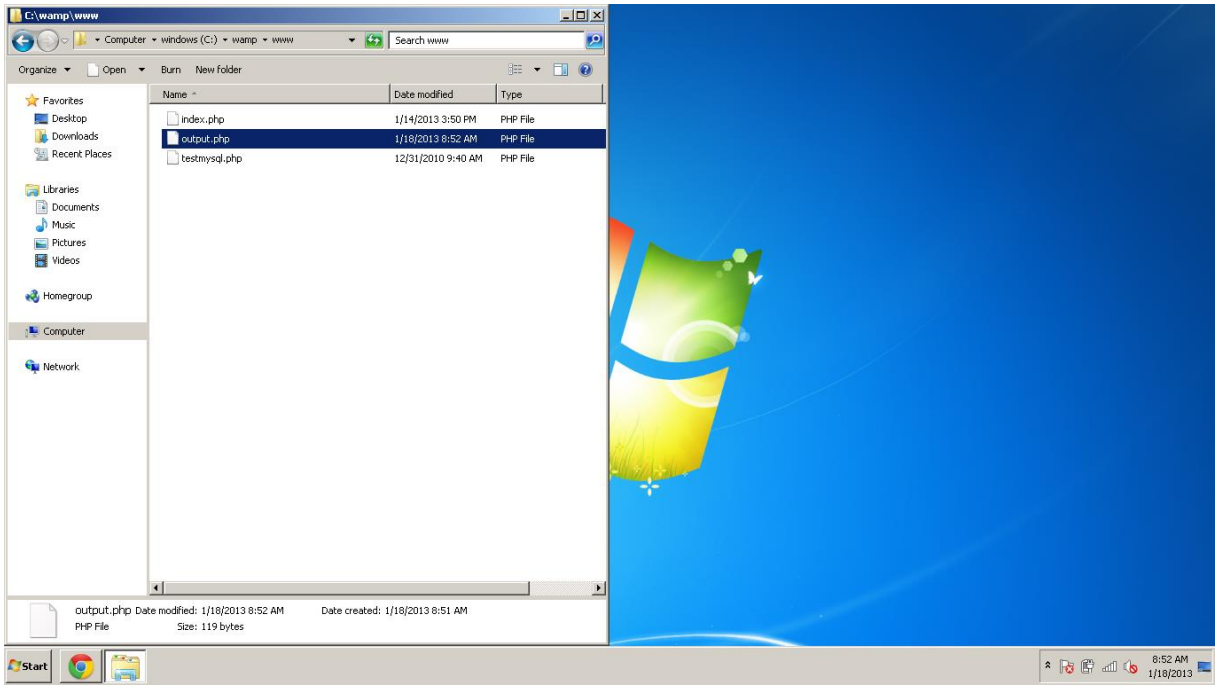


Une fois installé, lancez le logiciel. Il devrait apparaître dans le coin en bas à droite dans la barre d'état système.



Mettez le serveur en marche et assurez-vous que tous les services sont en cours d'exécution. Si vous sélectionnez l'icône de WampServer dans le bac, il vous permettra de savoir si elle est en cours d'exécution.

Voici la partie la plus délicate. Une fois que vous avez créé un document qui contient PHP, vous devez l'enregistrer dans le répertoire c :/wamp/www. Une fois que vous avez enregistré dans ce dossier, vous serez en mesure de tester votre document.

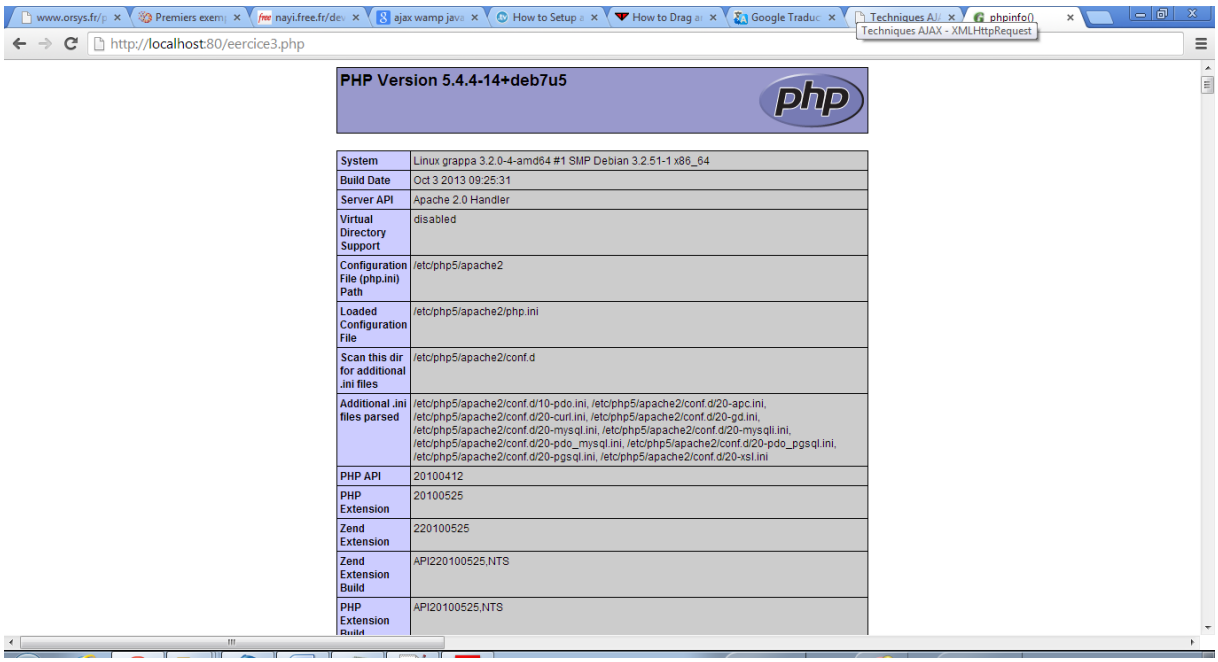


Une fois que votre serveur est chargé et votre document est dans le dossier www, chargez votre navigateur et tapez http://localhost/ suivie par le nom du fichier que vous testez. Dans ce cas, faire un fichier nommé exercice3.php.

```
<HTML><BODY>
<?php
  phpinfo();
?>
</BODY></HTML>
```

il apparaît dans le navigateur : http://localhost:80/exercice3.php

Vous devez le faire pour tous les fichiers enregistrés dans le répertoire c :/wamp/www.



System	Linux grappa 3.2.0-4-amd64 #1 SMP Debian 3.2.51-1 x86_64
Build Date	Oct 3 2013 09:25:31
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d/20-apc.ini, /etc/php5/apache2/conf.d/20-curl.ini, /etc/php5/apache2/conf.d/20-gd.ini, /etc/php5/apache2/conf.d/20-mysql.ini, /etc/php5/apache2/conf.d/20-mysqli.ini, /etc/php5/apache2/conf.d/20-pdo_mysql.ini, /etc/php5/apache2/conf.d/20-pdo_pgsql.ini, /etc/php5/apache2/conf.d/20-pgsql.ini, /etc/php5/apache2/conf.d/20-xsl.ini
PHP API	20100412
PHP Extension	20100525
Zend Extension	220100525
Zend Extension Build	API220100525,NTS
PHP Extension Build	API20100525,NTS

Etape2 : accès à une ressource

On va simplement lire le contenu d'un fichier XML, nommé XMLHttpRequest_getXML.xml dans le répertoire www de wamp:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- XMLHttpRequest_getXML.xml -->
<root>
    <soft name="Adobe Dreamweaver" />
    <soft name="Microsoft Expression Web" />
    <soft name="Notepad++" />
    <soft name="gedit" />
    <soft name="Emacs" />
</root>
```

C'est un fichier tout simple, contenant une liste d'éditeurs pouvant servir à écrire du Javascript.

Pour instancier (créer, déclarer) un objet XHR, on procède de la même façon que pour n'importe quel objet JavaScript à savoir avec le mot-clé new :

1var xhr = new XMLHttpRequest();

Les versions d'Internet Explorer inférieures à la version 7 requièrent toujours une instanciation via un contrôle ActiveX. Il y a deux façons d'instancier un objet XHR avec un contrôle ActiveX et elles dépendent de la version d'XMLHTTP utilisée. Pour faire simple, on va utiliser un try...catch , l'instanciation indiquée dans le try étant la plus récente ?

Il est demandé de construire un fichier Javascript nommé oXHR.js contenant l'appelle JavaScript dans le répertoire www de wamp.

```
function getXMLHttpRequest() {
	var xhr = null;

...

	return xhr;
}
```

Faire une page html nommée exercice3.html dans le répertoire www de wamp où en cliquant sur le bouton, on envoie la requête qui va se charger de récupérer les données, au format XML (avec responseXML donc). Ensuite, dans la fonction readData, on lit ces données et on crée une liste à puces que l'on va ajouter à l'élément appelé output.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Techniques AJAX - XMLHttpRequest</title>
	<script type="text/javascript" src="oXHR.js"></script>
<script type="text/javascript">
<!--
function request(callback) {
...
}

function readData(oData) {
...
}
//-->
</script>
</head>
<body>
<p>
    <button onclick="request(readData);">Afficher le fichier XML</button>
    <div id="output"></div>
</p>
</body>
</html>
```

Etape3 : envoi - traitement - réception

On demande à l'utilisateur d'entrer son pseudonyme ainsi que son prénom. Ces deux données seront transmises à une page PHP qui renverra un texte avec les données envoyées. Ça n'a aucun intérêt comme ça, mais c'est pour vous donner une base simple. Soit la page etape3.html


```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Techniques AJAX - XMLHttpRequest</title>
<script type="text/javascript" src="oXHR.js"></script>
<script type="text/javascript">
<!--
function request(callback) {
...
}

function readData(sData) {
    alert(sData);
}
//-->
</script>
</head>
<body>
<form>
    <p>
        <label for="nick">Pseudo :</label>
        <input type="text" id="nick" /><br />
        <label for="name">Prénom :</label>
        <input type="text" id="name" />
    </p>
    <p>
        <input type="button" onclick="request(readData);" value="Exécuter" />
    </p>
</form>
</body>
</html>
```

Et la page de script PHP etape3.php qui va traiter la requête et renvoyer la petite phrase :

```
<?php

header("Content-Type: text/plain");

$nick = (isset($_GET["Nick"])) ? $_GET["Nick"] : NULL;
```

```
$name = (isset($_GET["Name"])) ? $_GET["Name"] : NULL;

if ($nick && $name) {
    echo "Bonjour " . $name . " ! Je vois que votre pseudo est " . $nick;
} else {
    echo "FAIL";
}

?>
```

Notez bien l'utilisation d'encodeURIComponent pour protéger les données transmises. Si tout se passe bien, un alert s'affichera avec la petite phrase renvoyée par le script PHP.

Etape4 : AJAX et DOM

L'object XMLHttpRequest est pratique pour récupérer un peu n'importe quoi comme type de données (text, XML, HTML...), mais est généralement un peu lourd à utiliser. Si vous désirez juste récupérer une source de données au format XML, vous pouvez le faire via DOMImplementation.

L'interface DOMImplementation fait partie du DOM Level 2 Core et est supportée par la plupart des navigateurs. Internet Explorer supporte le DOMImplementation, mais par le biais d'un contrôle ActiveX (comme les versions 5 et 6 d'IE avec XMLHttpRequest).

Nous allons créer une fonction générique qui ira récupérer une source XML à l'adresse sUrl, et qui renverra le contenu dans une fonction de callback fCallback. Ainsi nous disposerons d'une fonction qui ne s'occupera que de la gestion de DOMImplementation : oDOMImplementation.js

```
function getDOMImplementation(sUrl, fCallback) {
    var dom;

    if (window.ActiveXObject) {
...
    }
    else if (document.implementation && document.implementation.createDocument) {
...
    }
    else {
...
    }

    dom.load(sUrl);
}
```

N.B. Attention que Webkit, le moteur de rendu utilisé notamment dans Safari et Google Chrome ne gère pas la méthode load de DOMImplementation.

Soit le fichier de données suivant nommé country.xml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<topic>
    <author>John</author>
    <country>Belgium</country>
    <lang>French</lang>
```

```
</topic>
```

Faire une page html nommé etape4.html pour demander cette information


```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<link href="mep.css" rel="stylesheet" type="text/css" media="all" />
<title>Techniques AJAX - DOMImplementation - XML Statique</title>

<script type="text/javascript" src="oDOMImplementation.js"></script>
<script type="text/javascript">
<!--
function getData(oData) {
...
}
//-->
</script>
</head>
<body>
<h1>Techniques AJAX - DOMImplementation - XML Statique </h1>
<p>
        <input type="button" value="Récupérer les données"
onclick="getDOMImplementation('country.xml', getData);" />
</p>
</body>
</html>
```

Vous pouvez utiliser une page PHP pour générer un fichier XML. Dans ce cas, des variables GET peuvent être transmis à la page PHP via l'url. Voici un exemple qui charge du XML créé dynamiquement au moyen d'une page PHP.

La page HTML est rigoureusement la même que dans l'exemple ci-dessus, à l'exception de l'appel de la fonction :

```
<input type="button" value="Récupérer les données"
onclick="getDOMImplementation('country.php?Pseudo=John', getData);" />
```

La page PHP (country.php) contient ce code :

```
<?php

header("Content-Type: text/xml");
```

```
echo "<?xml version=\"1.0\" encoding=\"iso-8859-1\"?>";
echo "<topic>";

if (isset($_GET['Pseudo'])) {
        $pseudo = $_GET['Pseudo'];

        if ($pseudo == 'John') {
                echo '<author>John</author>';
                echo '<country>Belgium</country>';
                echo '<lang>French</lang>';
        } else if ($pseudo == 'Jessica') {
                echo '<author>Jessica</author>';
                echo '<country>Belgium</country>';
                echo '<lang>French</lang>';
        }
}

echo "</topic>";
?>
```

Il ne faut surtout pas oublier de bien définir le type de contenu comme étant du text/xml. De plus, votre XML généré doit être valide, sinon vous aurez des problèmes lors du chargement du fichier par l'objet DOMImplementation.