jQuery 2.0

write less, do more.

Table des matières

CHAPITRE 1 – GESTION AJAX

1.1 - Petits rappels sur le protocole HTTP

Qu'est-ce qu'un protocole ?

Qu'est-ce que le protocole HTTP ?

Quelles sont les méthodes standards de base du protocole HTTP ?

Quels sont les éléments HTML qui permettent de requêter vers un serveur HTTPd ?

Quels sont les éléments HTML qui permettent de requêter en mode POST vers un serveur HTTPd ?

Quelles sont les différences fondamentales entre une requête POST et une requête GET ? Sémantiquement parlant et techniquement parlant ?

Que signifie U R L ?

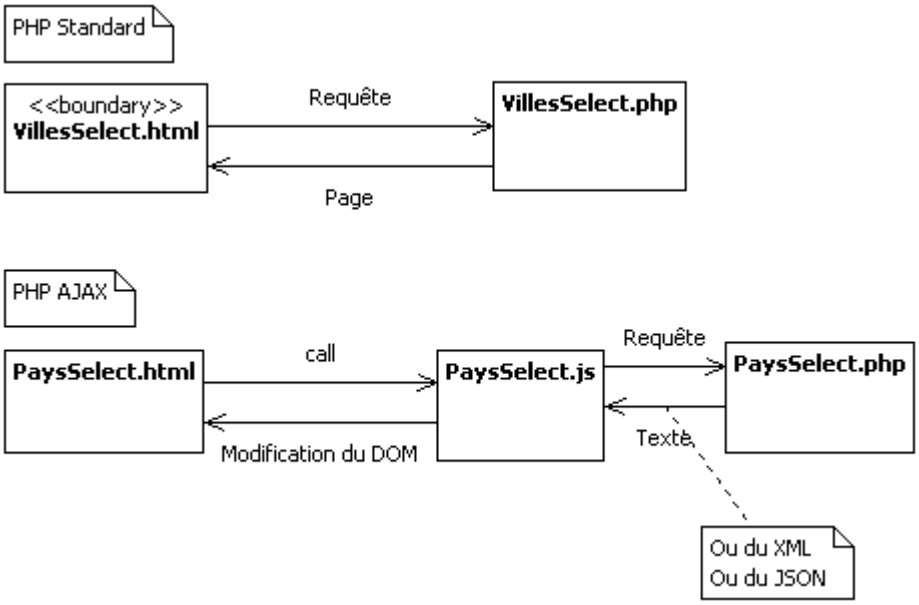
Quelle est la syntaxe d'un URL ?

Qu'est-ce qu'une ressource WEB ? Citez 5 types de ressources WEB.

Si vous ne pouvez pas répondre à ces questions - essentielles pour des développeurs WEB - , consultez votre cours HTML ou votre ami Google ou la fiche ProtocoleHTTP.odt ou ProtocoleHTTP.pdf !

Et faites un schéma, 2 niveaux ou mieux 3 niveaux, du fonctionnement du web et donc du protocole HTTP.

AJAX expliqué aux parents !



Quelques éléments de réponse de base !

Qu'est-ce qu'un protocole ?

Qu'est-ce que le protocole HTTP ?
Un protocole de communication Client/Serveur entre un client (un navigateur) et un serveur WEB (Apache, IIS, …).
Le client émet des requêtes (des demandes) et le serveur renvoie des réponses (un ressource web, un message d'erreur).

Quels sont des méthodes standards de base du protocole HTTP ?
GET, POST, PUT, DELETE, HEAD.

Quels sont les éléments HTML qui permettent de requêter vers un serveur HTTPd ?
La balise <meta> : <meta http-equiv="refresh" content="0; URL=url">
Les ancres : <a ref="url">
Les formulaires : <form action="ressource" method="methode">

Quels sont les éléments HTML qui permettent de requêter en mode POST vers un serveur HTTPd ?
Les formulaires : <form action="ressource" method="POST">

Quelles sont les différences fondamentales entre une requête POST et une requête GET ?
Sémantiquement parlant et techniquement parlant ?

	GET	POST
Sémantique	Obtenir	Envoyer
CRUD	R	CUD
Sécurité	Peu sûr	Plus sûr
Type de données	Texte	Texte et binaire
Taille des données	1024	No limit !
Moyen de requête	Ancre, meta, formulaire	Formulaire
AJAX	Oui	Oui

Notes complémentaires :

En JavaScript :
document.location.ref = "url";

En PHP :
header("location: url");

En Java :
response.sendRedirect("url");
response.setHeader("refresh", "url");
getServletContext().getRequestDispatcher("url").forward(request, response);

En JSP :
<jsp:forward page="url" />

En plus :

une requête AJAX permet elle aussi d'exécuter une requête HTTP vers un serveur WEB. C'est l'objet de ce document.

1.2 - PRÉSENTATION

AJAX est l'acronyme de **Asynchronous JavaScript and XML** et fédère un certain nombre de techniques existantes.
La classe JavaScript XMLHttpRequest est apparue avec IE 4.0 en 1997.
AJAX s'est imposé avec l'article de J.J. Garrett (2005).

Une architecture Client/Serveur doit équilibrer les charges sur les postes clients, le réseau et le serveur.
AJAX réalise des traitements client (avec JavaScript) à partir d'informations provenant du serveur.
La création de pages web dynamiques s'effectue principalement coté serveur avec des langages comme PHP, JSP et ASP. Les requêtes envoyées au serveur sont exécutées par le moteur dynamique et le serveur HTTP renvoie la réponse au navigateur au travers du réseau. AJAX permet de limiter les allers et retours client/serveur.
AJAX permet de **changer partiellement le contenu d'une page**.
Et AJAX est principalement **Asynchrone**.

Les techniques utilisées par AJAX sont les suivantes :

Le langage **HTML** pour la structure du document.
Le langage **CSS** (Cascading Style-Sheet) pour la présentation.
Le langage **JavaScript** (EcmaScript) pour les traitements locaux.
DOM (Document Object Model) pour accéder aux éléments de la page ou du formulaire ou aux éléments d'un fichier XML chargés à partir du serveur.
La classe **XMLHttpRequest** pour lire des données TEXT, SQL via un langage serveur, JSON ou XML serveur en mode asynchrone ou ~~synchrone~~.
Les langages **PHP, ASP, JSP, ...** pour les scripts coté serveur.

Les méthodes de jQuery sont les suivantes :

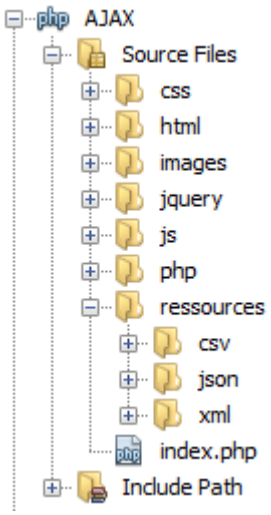
- ✓ $.get(),
- ✓ $.post(),
- ✓ $.ajax(),
- ✓ $.getJSON(),
- ✓ $.getScript().

Pour des raisons de sécurité les requêtes AJAX ne peuvent s'effectuer que sur le même domaine ou sous-domaine sauf pour les requêtes JSONP et les scripts JS et l'autorisation donnée de faire du « cross domain » (cf les annexes).

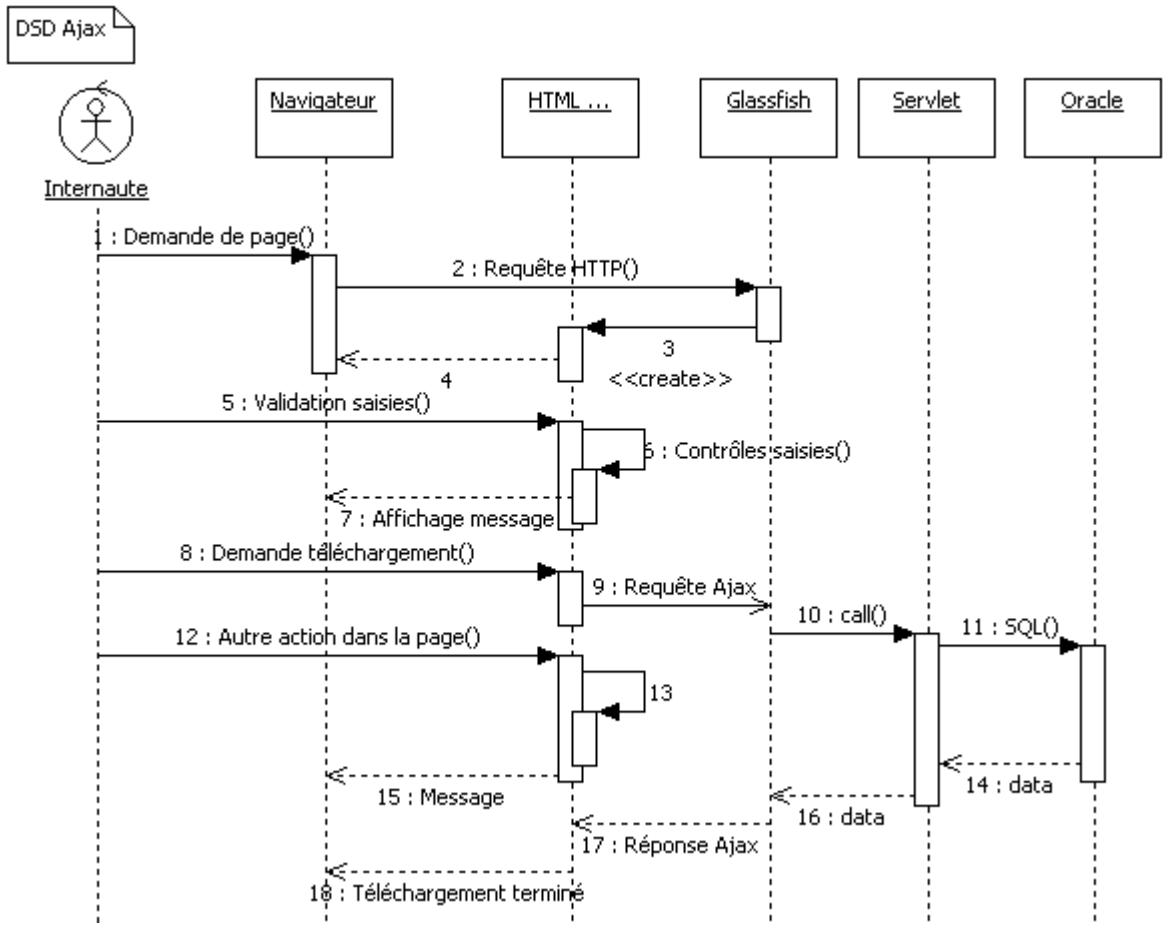
En résumé une requête AJAX est Asynchrone (non bloquante)
et ne nécessite pas le rechargement de toute la page puisque qu'elle permet de ne modifier qu'une partie de la page via la modification d'une partie du DOM.

Les corrigés sont dans le document jquery_exercices_corriges.odt

1.3 - L'ARBORESCENCE DU PROJET



Le schéma ...



Notez bien la flèche n° 9.

1.4 - Utiliser $.get()

1.4.1 - Objectif

Obtenir des données.

1.4.2 - Syntaxe de base

https://api.jquery.com/jquery.get/

```
jqXHR = $.get(
"ressource"
[, paramètres
[, function(data[, statut[, jqXHR]]) { manipulation de data }
, [type] ]
);
```

Argument	Description
Ressource	Une ressource serveur : un servlet Java, un script PHP, un fichier texte, un fichier HTML, un fichier XML, un fichier JSON, …, un URL en somme.
Paramètres	Facultatif : attribut(s) de requête Sous la forme : {attribut1:valeur1, attribut2:valeur2, …}
Fonction	La fonction anonyme qui récupère les données du serveur suite à la requête HTTP GET en cas de succès. C'est à vous de codez à l'intérieur de cette fonction (Affectation, …). Le statut : success, jqXHR : Objet jQuery mappant XHR.
Type	Facultatif : le type de données renvoyé par la requête. XML, HTML, JSON, script, text entre double quotes. Par défaut la fonction reconnaît le type des données (Intelligent Guess).

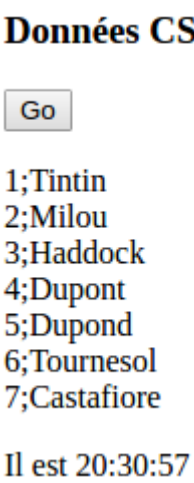
$.get() renvoie un objet jqXHR (jQuery XML HTTP Request object).

$.get() est un raccourci de la méthode $.ajax().

1.4.3 - Premier exemple

Afficher des données CSV (Comma Separated Values : les valeurs sont séparées par des ,) stockées dans un fichier nommé tintin.txt.

L'intérêt d'afficher l'heure est de démontrer que seule la liste est modifiée quand on clique à nouveau sur le bouton « Go » et que l'on a modifié la source.



tintin.txt

1;Tintin
2;Milou
3;Haddock
4;Casta
5;Castafiore
6;Dupont
7;Dupond
8;Tournesol
9;Rastapopoulos

Note : pour créer un fichier .txt avec NetBeans : Folder/Clic droit/New/Other/Other/Empy File.

```
<!DOCTYPE html>
<!--
JQAjaxGetCSV.html
-->
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
        <title>JQAjaxGetCSV.html</title>
    </head>

    <body>
        <div>
            <h3>Données CSV</h3>
            <input type="button" id="btGo" value="Go" />
            <p id="pMessage"></p>
            <p id="pHeure"></p>
        </div>

        <script src="../jquery/jquery.js"></script>
        <script src="../js/JQAjaxGetCSV.js"></script>
    </body>
</html>
```

```
/*
 * JQAjaxGetCSV.js
 */

/**
 *
 * @returns {undefined}
 */
function afficherCSV() {
    $.get(
            "../ressources/csv/tintin.txt",
            function(data) {
                console.log(data);
                // Pour transformer les RC en <br>
                let regex = /\n/g;
                data = data.replace(regex, "<br>");
                console.log(data);
                $("#pMessage").html(data);
            }
    ); /// $.get
} /// afficherCSV

/**
 *
 * @returns {undefined}
 */
function getHeure() {
    let maintenant = new Date();
    return maintenant.getHours() + ":" + maintenant.getMinutes() + ":" +
maintenant.getSeconds();
} /// getHeure


// -------------------------
$(document).ready(function() {
    $("#pHeure").html("Il est " + getHeure());
    $("#btGo").on("click", afficherCSV);
});
```

La même chose sans fonction anonyme ! (Merki Olivier !!!)

```
/*
 * JQAjaxGetCSVBis.js
 */

/**
 *
 * @returns {undefined}
 */
function afficherCSV() {
    $.get(
            "../ressources/csv/tintin.txt",
            getData
            ); /// $.get
} /// afficherCSV

/**
 *
 * @param {type} data
 * @returns {undefined}
 */
function getData(data) {
    console.log(data);
    // Pour transformer les RC en <br>
    let regex = /\n/g;
    data = data.replace(regex, "<br>");
    console.log(data);
    $("#pMessage").html(data);
} /// getData

/**
 *
 * @returns {String}
 */
function getHeure() {
    let maintenant = new Date();
    return maintenant.getHours() + ":" + maintenant.getMinutes() + ":" +
maintenant.getSeconds();
} /// getHeure

// -------------------------
$(document).ready(function() {
    $("#pHeure").html("Il est " + getHeure());
    $("#btGo").on("click", afficherCSV);
});
```

Le même fichier .csv dans une liste déroulante.

Créez un nouveau fichier HTML avec un élément <select> identifié par « liste ».

```
<select id="liste" size="3"></select>
```

Créez un nouveau fichier .js, inspirez-vous du précédent et codez ainsi la fonction « afficherCSV ».

```
function afficherCSV() {

    $.get(
            "../ressources/csv/tintin.txt",
            function (data) {
                console.log(data);
                let tRecords = data.split("\n");
                for (let i = 0; i < tRecords.length; i++) {
                    console.log(tRecords[i]);
                    if (tRecords[i].trim().length > 0) {
                        let tFields = tRecords[i].split(";");
                        let opt = $("<option>");
                        opt.val(tFields[0]);
                        opt.html(tFields[1]);
                        $("#liste").append(opt);
                    } /// fi
                } /// rof
            } /// function
    ); /// $.get

} /// afficherCSV
```

1.4.4 - Exercice : le même fichier .csv dans une liste.

Données CSV

- Tintin
- Milou
- Haddock
- Casta
- Castafiore
- Dupont
- Dupond
- Tournesol
- Rastapopoulos

1.4.5 - Autre syntaxe : get, done et fail

Requête, récupération des résultats, gestion des erreurs.

```
let jqXHR = $.get(
"ressource"
[, paramètres
, [type] ]
);


jqXHR.done(function(data [,statut[, xhr]]) {
    // code
});


jqXHR.fail(function(xhr, statut, erreur) {
    // code
});
```

Les paramètres de done() sont les données, le statut de la requête, un objet XHR.

Objet/String	Description
data	Les données reçues
statut	"success".
xhr	xhr.status : 200, 404, ... xhr.statusText : OK, Not Found, ...

Les paramètres de fail(xhr, statut, erreur).

Objet/String	Description
xhr	xhr.status : 200, 404, ... xhr.statusText : Not Found, ...
statut	"timeout", "error", "abort", and "parsererror".
erreur	HTTP Status (Not Found, Internal Server Error, ...)

```
<!DOCTYPE html>
<!--
JQAjaxGetCSVErreur.html
-->
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
        <title>JQAjaxGetCSVErreur.html</title>
    </head>

    <body>
        <div>
            <h3>Données CSV</h3>
            <input type="button" id="btGo" value="Go" />
            <p id="pResultat"></p>
        </div>

        <script src="../jquery/jquery.js"></script>
        <script src="../js/JQAjaxGetCSVErreur.js"></script>
    </body>
</html>
```

```
/*
 * JQAjaxGetCSVErreur.js
 */

/**
 *
 * @returns {undefined}
 */
function init() {
    $("#btGo").on("click", afficherCSV);
} /// init

/**
 *
 * @returns {undefined}
 */
function afficherCSV() {
    console.log("afficherCSV");

    let jqXHR = $.get(
            "../ressources/csv/tintin.xxx"
            );

    jqXHR.done(function(data) {
        console.log("done");
        console.log(data);
        // Pour transformer les RC en <br>
        let regex = /\n/g;
        data = data.replace(regex, "<br>");
        console.log(data);
        $("#pResultat").html("Done<br>" + data);
    });

    jqXHR.fail(function(xhr, statut, erreur) {
// xhr.status : 404
// xhr.statusText : Not Found
// statut : error
// erreur : Not Found par exemple
        $("#pResultat").html("Erreur : " + xhr.status + "-" +
xhr.statusText);
    });
} /// afficherCSV

// --------------------
$(document).ready(init);
```

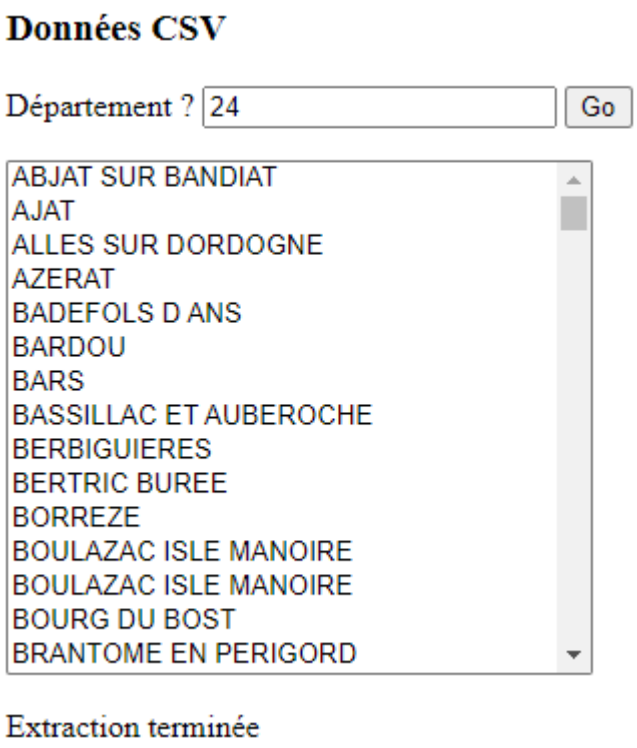
1.4.6 - Exercice : données CSV filtrées

Exercice pour les experts à Paris, Lille, Bordeaux, Lyon, ...

Filtrer des données CSV …

Trouvez sur Internet la liste des communes de France (> 36000) au format CSV.

Créer un formulaire avec une zone de saisie où l'utilisateur saisira un département, une liste qui affichera la liste des villes triée par ordre alphabétique d'un département sur le clic de l'utilisateur.
Vous gérerez aussi le touche « Entrée » du clavier dans la zone de saisie pour valider.
Vous afficherez dans un <label> « Extraction terminée » à la fin du traitement ou un message d'erreur si une erreur s'est produite.



1.4.7 - Obtenir des données JSON

1.4.7.1 - Pré-requis

Avoir suivi le cursus « Présentation de JSON ». Sinon cf JSON.pdf et json_et_js.odt.

Un éditeur de JSON en ligne :
https://jsoneditoronline.org/#left=local.kitide&right=local.jiredo

1.4.7.2 - Objectif du cas 1 : un objet JSON

Récupérer **une donnée JSON (un objet JSON)** stockée dans un fichier (ville.json) et l' afficher.

OK :

Paris [75000] - Habitants : 2000000

Erreur AJAX : 404-Not Found (erreur dans l'URL).

Statut : parsererror (Erreur dans le fichier JSON).

Erreur : SyntaxError: Unexpected token P (Erreur dans le fichier JSON).

ville.json

```
{
    "ville": "Paris",
    "codePostal": "75000",
    "habitants": "2000000"
}
```

1.4.7.3 - Syntaxe

```
let jqXHR = $.get(
       "source.json",
       function(data) {
          // Code
       },
       "json"
       );
```

cf la « syntaxe étendue » au paragraphe 1.4.4.

1.4.7.4 - Codes

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
        <title>JQGet_JSONVille.html</title>
    </head>
    <body>
        <label id="lblMessage"></label>
        <script src="../jquery/jquery.js"></script>
        <script src="../js/JQGet_JSONVille.js"></script>
    </body>
</html>
```

Avec done() et fail()

jqXHR.done() fonctionne à condition de transformer la String de data en objet JSON – désérialisation - avec la méthode JSON.parse() quand c'est un flux généré par un script PHP ou un servlet Java. Quand c'est suite à la lecture d'un fichier JSON – comme dans le script qui suit – ce n'est pas nécessaire.

```
/*
 * JQGet_JSONVille.js
 */
let jqXHR = $.get(
        "../ressources/json/ville.json",
        "json"
        ); /// $.get


jqXHR.done(function(data) {
    console.log(data);
    //data = JSON.parse(data);
    $("#lblMessage").html(data.ville + " [" + data.codePostal + "] -
Habitants : " + data.habitants);
});


jqXHR.fail(function(xhr, statut, erreur) {
    console.log("Erreur AJAX : " + xhr.status + "-" + xhr.statusText + " :
" + statut);
    $("#lblMessage").html(xhr.status + "-" + xhr.statusText);
});
```

1.4.7.5 - Objectif du cas 2 : un tableau d'objets JSON

Récupérer **des données JSON (un tableau d'objets JSON)** stockées dans un fichier (villes.json) et les afficher.

villes.json

```
[
    {
        "ville": "Paris",
        "codePostal": "75000",
        "habitants": "2 000 000"
    },
    {
        "ville": "Lyon",
        "codePostal": "69000",
        "habitants": "450 000"
    },
    {
        "ville": "Marseille",
        "codePostal": "13000",
        "habitants": "800 000"
    }
]
```

1.4.7.6 - Syntaxes

Les mêmes.

1.4.7.7 - Codes

```
<!DOCTYPE html>
<html>
    <head>

        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
        <title>JQGet_JSONVilles.html</title>
    </head>

    <body>
        <label id="lblMessage"></label>
        <script src="../jquery/jquery.js"></script>
        <script src="../js/JQGet_JSONVilles.js"></script>
    </body>
</html>
```

```
/*
 * JQGet_JSONVilles.js
 */
let jqXHR = $.get(
        "../ressources/json/villes.json",
        "json"
        ); /// $.get


jqXHR.done(function(data) {
    // data = JSON.parse(data); // Dé-sérialisation (chaîne vers objet)
    console.log(data);
    let lsVilles = "";
    for (let i = 0; i < data.length; i++) {
        lsVilles += data[i].codePostal + " - " + data[i].ville + "<br>";
    }
    $("#lblMessage").html(lsVilles);
});


jqXHR.fail(function(xhr, statut, erreur) {
    console.log("Erreur AJAX : " + xhr.status + "-" + xhr.statusText + " :
" + statut);
    $("#lblMessage").html(xhr.status + "-" + xhr.statusText);
});
```

1.4.8 - Exercice : données JSON filtrées

Exercice pour les experts à Paris, Lille, Bordeaux, Lyon, ...

Filtrer des données JSON …

1.4.9 - Obtenir des données XML à partir d'un fichier de type Data

1.4.9.1 - Objectif

Récupérer des données XML d'un fichier de type Data (villesData.xml cf les annexes) et les afficher dans un texte brut.

1.4.9.2 - Syntaxes

Fonction	Description
find("nom_d_element")	Recherche et renvoie un ou plusieurs éléments
each(fonction)	Boucle sur une liste d'éléments
attr("nom_d_attribut)	Renvoie la valeur d'un attribut
text()	Renvoie la valeur d'un nœud #text

Ceci va permettre de rechercher plusieurs éléments dans l'arbre DOM et de boucler sur cette liste.

```
$(data).find("élément" | "expression XPath ou autre").each(function() {
```

jQuery et XPath : http://www.ibm.com/developerworksrary/x-xpathjquery/

S'il n'y a qu'un seul résultat :

```
let resultat = $(data).find("élément" | "expression XPath ou autre");
```

```
let resultat = $(data).find("élément" | "expression XPath ou autre").text();
```

1.4.9.3 - Plusieurs résultats

jQuery Ajax XML Data Get

Marseille
Bordeaux
Lille
Paris XI
Paris XII

JQGetXMLData.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>JQGetXMLData</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
    </head>

    <body>
        <div>
            <h3>JQGetXMLData</h3>
            <p id="pResultat"></p>
        </div>

        <script src="../jquery/jquery.js"></script>
        <script src="../js/JQGetXMLData.js"></script>
    </body>
</html>
```

JQGetXMLData.js

```
/*
 * JQGetXMLData.js
 */

// ------------
function init() {
    let jqXHR = $.get(
            "../ressources/xml/villesData.xml"
            );

    jqXHR.done(function(data) {
        let lsVilles = "";
        $(data).find("ville").each(function() {
            // --- Recuperation de la valeur de l'attribut
            let lsNomVille = $(this).attr("nom_ville");
            lsVilles += lsNomVille + "<br/>";
        });
        $("#pResultat").html(lsVilles);
    });

    jqXHR.fail(function(xhr, statut, erreur) {
        $("#pResultat").html("Erreur : " + xhr.status + "-" +
xhr.statusText);
    });

} /// init

// ---------------------
$(document).ready(init);
```

1.4.9.4 - Pour un seul résultat

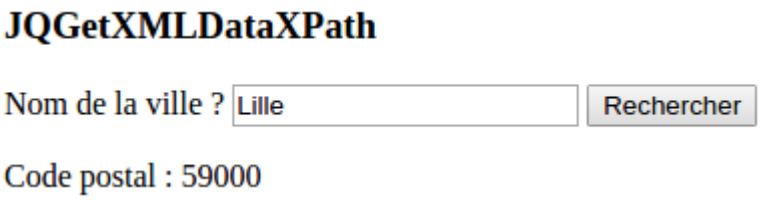
```
<?xml version="1.0" encoding="UTF-8"?>
<!--
villeData.xml
-->
<root>
    <capitale id="France" nom="Paris" />
</root>
```

Le même HTML sauf l'appel au script.

```
/*
 * JQGetXMLDataUnElement.js
 */

// ------------
function init() {

    let jqXHR = $.get(
            "../ressources/xml/villeData.xml"
            ); /// get

    jqXHR.done(function(data) {
        let lsContenu = "";
        let data = $(data).find("capitale");
        // Les attributs
        let attributs = data[0].attributes;
        for (let i = 1; i < attributs.length; i++) {
            lsContenu += attributs[i].nodeValue + "<br>";
        }

        $("#pResultat").html(lsContenu);

    });

} /// init

// ---------------------
$(document).ready(init);
```

1.4.9.5 - Avec XPATH

Objectif



Syntaxes

il faut télécharger la bibliothèque jquery-xpath.

https://github.com/ilinsky/jquery-xpath

```
$(context).xpath(expression, resolver)
```

```
$.xpath(context, expression, resolver)
```

Codes

```
<!DOCTYPE html>
<html>
    <head>
        <title>JQGetXMLDataXPath</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
    </head>

    <body>
        <div>
            <h3>JQGetXMLDataXPath</h3>
            <label>Nom de la ville ?</label>
            <input type="text" id="nomDeLaVille" />
            <button id="btRechercher">Rechercher</button>
            <p id="pResultat"></p>
        </div>

        <script src="../jquery/jquery.js"></script>
        <script src="../jquery/jquery.xpath.js"></script>
        <script src="../js/JQGetXMLDataXPath.js"></script>
    </body>
</html>
```

```
/*
 * JQGetXMLDataXPath.js
 *
 * Rechercher le code postal d'une ville
 *
 */

// ----------
function init() {
    $("#btRechercher").click(rechercher);
} /// init

// ----------------
function rechercher() {

    // "http://localhost/jQueryAjaxCours/ressources/xml/villesData.xml"

    let jqXHR = $.get
            (

"http://localhost/jQueryAjaxCours/ressources/xml/villesData.xml",
                    function(data) {
                        let lsNomDeLaVille = $("#nomDeLaVille").val();
                        let lsExpressionXPath;

                        // A LA SAUCE jQuery
                        //lsExpressionXPath =
"//villes/ville[@nom_ville='" + lsNomDeLaVille + "']/cp"; // KO !!!
                        lsExpressionXPath = "villes ville[nom_ville='" +
lsNomDeLaVille + "']";
                        let uneVille = $(data).find(lsExpressionXPath);

                        // AVEC LA BIBLIOTHEQUE XPATH
                        //lsExpressionXPath =
"//villes/ville[@nom_ville='" + lsNomDeLaVille + "']/cp"; /// KO !!!
//                         lsExpressionXPath = "//villes/ville[@nom_ville='"
+ lsNomDeLaVille + "']";
//                         let uneVille = $(data).xpath(lsExpressionXPath);

                        // Un seul resultat donc uneVille[0]
                        let lsCP = uneVille[0].attributes["cp"].nodeValue;
                        $("#pResultat").html(lsCP);
                    }
            );
    jqXHR.fail(function(xhr, statut, erreur) {
        $("#pResultat").html(xhr.status + "-" + xhr.statusText);
    });
} /// rechercher

// ---------------------
$(document).ready(init);
```

1.4.10 - Obtenir des données XML contenues dans un fichier de type Document

1.4.10.1 - Objectif

Récupérer des données XML d'un fichier de type Document (villesDocument.xml - cf les annexes) et les afficher dans une liste (dans les <li>).

jqGetXmlDocument

- Paris
- Lyon
- Cabourg

1.4.10.2 - Syntaxes

Fonction	Description
find("nom_d_element")	Recherche et renvoie un ou plusieurs éléments
each(fonction)	Boucle sur une liste d'éléments
text()	Renvoie la valeur d'un nœud #text

JQGetXMLDocument.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>JQGetXMLDocument.html</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
    </head>

    <body>
        <div>
            <h3>JQGetXMLDocument</h3>
            <ul id="listeVilles">
            </ul>
        </div>

        <script src="../jquery/jquery.js"></script>
        <script src="../js/JQGetXMLDocument.js"></script>
    </body>
</html>
```

JQGetXMLDocument.js

```
/*
 * JQGetXMLDocument.js
 */

// ------------
function init() {
    let jqXHR = $.get(
            "../ressources/xml/villesDocument.xml",
            "xml"
            );

    jqXHR.done(function(data) {
        console.log(data);
        let lsVilles = "";
        $(data).find("nom_ville").each(function() {
            // --- Recuperation de la valeur de l'attribut
            let li = $("<li>");
            li.html($(this).text());
            $("#listeVilles").append(li);
        });


    });
}

// ----------------------
$(document).ready(init);
```

1.4.11 - Balayer le DOM d'un document XML de type document

Quelques syntaxes

Récupérer un ou plusieurs éléments :

```
find("element")
```

Récupérer le nœud text :

```
element.text()
```

Récupérer l'élément suivant :

```
element.next()
```

Récupérer les éléments enfants d'un élément :

```
element.children()
```

Exemple

- 75000-Paris-033-France
- 69000-Lyon-033-France
- 14200-Cabourg-033-France

jqGetXmlDocEnfants.js

```
// ------------
function init() {
    $.get
    (
        "../ressources/villesDocument.xml",
        function(data) {
            $(data).find("ville").each(function() {
                let enfants = $(this).children();
                let lsCPVille = "";
                enfants.each(function() {
                    lsCPVille += lsEnfant = $(this).text() + "-";
                });
                lsCPVille = lsCPVille.substring(0, lsCPVille.length - 1);

                let li = $("<li>");
                li.html(lsCPVille);
                $("#listeVilles").append(li);
            });
        }
    );
}

$(document).ready(init);
```

1.4.12 – Exercice : afficher des données XML de type data dans une liste déroulante

Ajoutez les villes du fichier villesData.xml dans une liste déroulante (<select>).

Vous pouvez aussi faire le même exercice en travaillant sur villesDocument.xml.

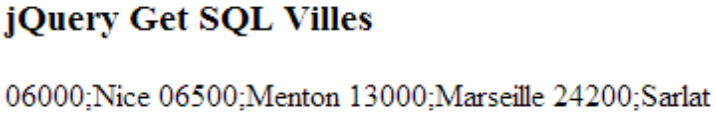
1.4.13 - Obtenir des données SQL sans paramètre

Objectif

Récupérer des données SQL (En fait du texte).

Exemple

Récupérer la liste des villes de la base de données.
Exécuter une requête GET sans paramètre.




```
<!DOCTYPE html>
<!--
JQGetSQLVilles.html
-->
<html>
    <head>
        <title>JQGetSQLVilles</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
    </head>

    <body>
        <div>
            <h3>JQGetSQLVilles</h3>
            <p id="pResultats"></p>
        </div>

        <script src="../jquery/jquery.js"></script>
        <script src="../js/JQGetSQLVilles.js"></script>

    </body>
</html>
```

Le script JS

Pour le script PHP cf les annexes.

```
/*
 * JQGetSQLVilles.js
 */

/**
 *
 * @returns {undefined}
 */
function init() {
    let jqXHR = $.get(
            "../php/VillesSelect.php"
            ); /// $.get

    jqXHR.done(function(data) {
        console.log(data);
        // Pour transformer les RC en <br>
        let regex = /\n/g;
        data = data.replace(regex, "<br>");
        console.log(data);
        $("#pResultat").html(data);
    });

} /// init


// --------------------
$(document).ready(init);
```

1.4.14 - Obtenir des données SQL en passant un paramètre

- **Objectif**

Récupérer les villes d'un pays : exécuter une requête GET avec un passage de paramètres.


```
<!DOCTYPE html>
<!--
JQGetSqlVillesDUnPays.html
-->
<html>
    <head>
        <title>JQGetSQLVillesDUnPays</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
    </head>
    <body>
        <div>
            <h3>JQGetSqlVillesDUnPays</h3>

            <label>Code pays </label>
            <input id="itCP" name="itCP" type="text" value="033" size="4"/
>
            <input id="btVoir" name="btVoir" type="button" value="Voir les
villes" />
            <br/>
            <p id="pResultats"></p>
        </div>
        <script src="../jquery/jquery.js"></script>
        <script src="../js/JQGetSQLVillesDUnPays.js"></script>
    </body>
</html>
```

Le script JS

Cf les annexes.

Le script récupère le code du pays saisi,
puis on passe comme arguments de la méthode GET la ressource PHP qui requête vers MySQL,
puis le code du pays,
et enfin la fonction de récupération avec l'affectation du résultat au <label>.

```
/*
 * JQGetSQLVillesDUnPays.js
 */
// ------------
function init() {
    $("#btVoir").click(voir);
}

// ------------
function voir() {
    let lsCodePays = $("#itCP").val();
    let jqXHR = $.get(
            "../php/VillesDUnPays.php",
            {id: lsCodePays}
    ); /// $.get

    jqXHR.done(function(data) {
        if (data === "") {
            $("#pResultat").html("Aucun résultat");
        }
        else {
            $("#pResultat").html(data);
        }
    });
} /// voir

// ------------------
$(document).ready(init);
```

1.5 - Utiliser $.post()

https://api.jquery.com/jquery.post/

1.5.1 - Objectif

Poster des données SQL (**S**tructured **Q**uery **L**anguage) avec la méthode $.post() via un script PHP (**P**HP : **H**ypertext **P**reprocessor).

1.5.2 - Syntaxe

```
jqXHR = $.post(
"url",
{ attribut1:valeur1 [, attribut2:valeur2] }
[, fonction callback en cas de succès de la requête AJAX ]
[, type]
);
```

Paramètre	Description
url	La ressource requêtée (Une servlet Java, du PHP, de l'ASP, de l'ASPX, …)
Attribut:valeur	L'attribut et la valeur postés
Callback	[Facultatif] Fonction en cas de succès de la requête XHR. callback(data, textStatus, jqXHR) Elle permet d'afficher le résultat de la requête. En cas d'erreur de la requête XHR (url not found, …) rien ne sera affiché. Pour la gestion des erreurs AJAX cf plus loin.
Type	[Facultatif] Le type de contenu renvoyé : text, xml, html, json.

$.post renvoie un objet jqXHR (jQuery XML HTTP Request object).
$.post() est un raccourci de la méthode $.ajax().

Exemple : envoi de trois valeurs pour une insertion via la méthode POST.

```
let jqXHR = $.post(
    "http://localhost:80/jquery_cours/villesInsert.php",
    { cp:lsCp, nom_ville:lsNomVille , id_pays:lsIdPays},
    function(data) {
        $("#lblMessage").html("XHR réussie. " + data);
    },
    "text"
);
```

data : le texte renvoyé par le script PHP, soit "1 enregistrement(s) ajouté(s)" soit un message d'erreur SQL.En gras les noms des paramètres de la requête.

1.5.3 - Avec la gestion d'erreur

```
let jqXHR = $.post(
"ressource"
[, paramètres
, [type] ]
);

jqXHR.done(function(data, statut, xhr) {
   // code
});

jqXHR.fail(function(xhr, statut, erreur) {
   // code
});
```

Les paramètres de done() sont les données, le statut de la requête, un objet XHR.

Les paramètres de fail(xhr, statut, erreur).

Objet/String	Description
xhr	xhr.status : 200, 404, ... xhr.statusText : OK, Not Found, ...
statut	"timeout", "error", "abort", and "parsererror".
erreur	HTTP Status (Not Found, Internal Server Error, ...)

```
let jqXHR = $.post(
    "http://localhost:80/jquery_cours/villesInsert.php",
    { cp:lsCp, nom_ville:lsNomVille , id_pays:lsIdPays},
    "text"
);

jqXHR.done(function(data) {
    // Pour transformer les RC en <br>
    let regex = /\n/g;
    data = data.replace(regex, "<br>");
    $("#pResultat").html("Done<br>" + data);
});

jqXHR.fail(function(xhr, statut, erreur) {
    $("#pResultat").html("Erreur : " + xhr.status + "-" + xhr.statusText);
});
```

1.5.4 - Poster des données SQL via un script PHP

Ajouter un enregistrement dans la table [villes] d'une BD MySQL via un script PHP.

La première fois :

La deuxième fois :

Pour le script PHP cf les annexes.

Les scripts HTML et JS

Les valeurs des champs de saisie sont récupérées et envoyées.
En cas de succès de la requête XHR un message est affiché. Cela peut être un message d'erreur MySQL.

```
<!DOCTYPE html>
<html>
  <head>
    <title>JQPostInsertVille.html</title>
    <meta charset="UTF-8">
  </head>

  <body>
      <label>CP : </label>
      <input type="text" id="itCP" value="06500" size="5" />
      <label>Ville : </label>
      <input type="text" id="itNomVille" value="Menton" />
      <label>ID pays : </label>
      <input type="text" id="itIdPays" value="33" size="4" />

      <button id="btInsert">Enregistrer</button>
      <br/>
      <label id="lblMessage"></label>

    <script src="../jquery/jquery.js"></script>
    <script scr="JQPostInsertVille.js"></script>
  </body>
</html>
```

```
/*
 * JQPostInsertVille.js
 */
// ------------
function init() {
    $("#btInsert").on("click", insert);
} /// init

// --------------
function insert() {
    let lsCp = $("#itCP").val();
    let lsNomVille = $("#itNomVille").val();
    let lsIdPays = $("#itIdPays").val();

    let jqXHR = $.post(
            "../php/VillesInsert.php",
            {cp: lsCp, nom_ville: lsNomVille, id_pays: lsIdPays},
    "text"
            );

    jqXHR.done(function(data) {
        $("#lblMessage").html(data);
    });

    jqXHR.fail(function(xhr, statut, erreur) {
        let sTexte = xhr.status + ":" + xhr.statusText;
        $("#lblMessage").html(sTexte);
    });

} /// insert

/// -------------------
$(document).ready(init);
```

1.5.5 - Exercice : supprimer une ville dans la table [villes]

Objectif

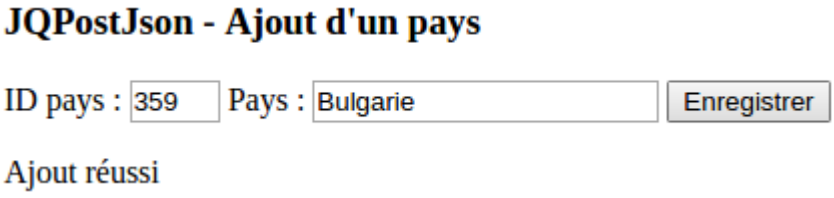
Supprimer une ville.

Le script PHP

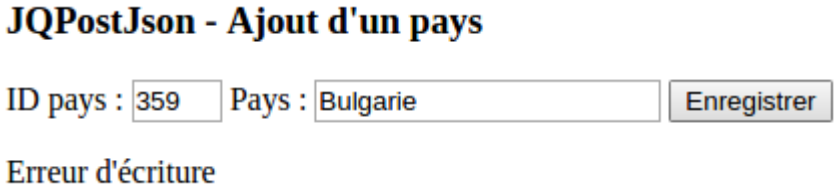
cf les annexes.

1.5.6 - Poster des données JSON

Objectif : ajouter un pays dans pays.json. Ce n'est possible qu'avec un script d'un langage serveur (php, servlet, aspx, …).



Avec une gestion d'erreur




```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
        <title>JQPostJson.html</title>
    </head>

    <body>
        <div>
            <h3>JQPostJson - Ajout d'un pays</h3>

            <p>
                <label>ID pays : </label>
                <input type="text" id="itIdPays" value="359" size="4" />
                <label>Pays : </label>
                <input type="text" id="itNomPays" value="Bulgarie" />
                <button id="btInsert">Enregistrer</button>
            </p>
            <p>
                <label id="lblMessage"></label>
            </p>
        </div>

        <script src="../jquery/jquery.js"></script>
        <script src="../js/JQPostJson.js"></script>
    </body>
</html>
```

```
/*
 * JQPostJson.js
 */

// ------------
function init() {
    $("#btInsert").on("click", insert);
} /// init

// --------------
function insert() {

    let objetJSON = {};
    objetJSON["id_pays"] = $("#itIdPays").val();
    objetJSON["nom_pays"] = $("#itNomPays").val();
    let stringJSON = JSON.stringify(objetJSON);

    let jqXHR = $.post(
            "../php/JQPostJson.php",
            {nouveauPays: stringJSON},
    "text"
            );

    jqXHR.done(function(data) {
        data = data.trim();
        console.log("*" + data + "*");
        let lsMessage = "";
        if (data === "-1") {
            lsMessage = "Erreur d'écriture";
        }
        else {
            lsMessage = "Ajout réussi";
        }

        $("#lblMessage").html(lsMessage);
    });

    jqXHR.fail(function(xhr, statut, erreur) {
        let sTexte = xhr.status + ":" + xhr.statusText;
        $("#lblMessage").html(sTexte);
    });

} /// insert

/// -------------------
$(document).ready(init);
```

pays.json

```
{"id_pays":"359","nom_pays":"Bulgarie"}
```

JQPostJson.php

Note : ce simple script PHP ne fait que créer un nouveau fichier nommé pays.json et écrase l'ancien s'il existe. Pour ajouter cf le script de la page suivante.

```
<?php

/*
 * JQPostJson.php
 */

// Gestionnaire d'erreur
function monGestionnaireErreur($errNo, $errMsg) {
    echo "-1";
}

// Gestion des erreurs
set_error_handler("monGestionnaireErreur");

// Ecriture
$nouveauPays = $_POST["nouveauPays"];
file_put_contents("pays.json", $nouveauPays);

?>
```

Deuxième version : on lit le fichier, on ajoute et on réécrit le fichier.

```
<?php

/*
 * JQPostJson.php
 */

// Gestionnaire d'erreur
function monGestionnaireErreur($errNo, $errMsg) {
    echo "-1";
}

// Gestion des erreurs
set_error_handler("monGestionnaireErreur");

/*
 *  Lecture-Reecriture (version 2)
 */
$nomFichier = "pays.json";
// Lecture du fichier -> une String
$lesPays = file_get_contents($nomFichier);
// Transformation de la String en un tableau ordinal de tableaux
associatifs
$tPays = json_decode($lesPays, true);

// Recuperation du nouveau pays : une String JSON
$nouveauPays = $_POST["nouveauPays"];
// Transformation de la String JSON en un tableau associatif
$nouveauPays = json_decode($nouveauPays, true);
// Ajout dans le tableau ordinal
$tPays[] = $nouveauPays;
// Transformation du tableau en une String
$stringJSON = json_encode($tPays);

// Reecriture sur le disque
file_put_contents($nomFichier, $stringJSON)

?>
```

1.6 - Utiliser $.ajax()

https://api.jquery.com/jquery.ajax/

1.6.1 - Objectif et syntaxe

Objectif

Récupérer ou envoyer des données de tous types (XML, PHP-SQL, JSON, …).
Sans paramètre d'abord puis avec un paramètre.
Permet de gérer finement les propriétés (status, readyState, responseText, …) de l'objet XHR à la différence de $.post(), $.get(), $getJSON().
Et aussi de faire des requêtes synchrones.

Syntaxe

```
jqXHR = $.ajax( {options} );
```

Note : jqXHR = jQuery XMLHttpRequest

Options

Option	Description
async : booléen	**true** ou false
url : "url"	L'url de la requête
cache : booléen	**true** ou false. If set to false, it will force requested pages not to be cached by the browser. Note: Setting cache to false will only work correctly with HEAD and GET requests. It works by appending "_={timestamp}" to the GET parameters.
contentType : "type"	Type de données envoyées au serveur. Valeur par défaut : "application/x-www-form-urlencoded; Charset=UTF-8".
data : objet | chaîne	Données envoyées au serveur (donc des attributs de requête) sous la forme de paires. Soit sous forme d'objet {attribut:valeur, attribut:valeur, …} Soit sous forme de chaîne : attribut=valeur&attribut=valeur
dataType : "type"	Données reçues en retour de la requête : **xml**, **html**, text, json, script ; par défaut le type est reconn (dataType (default: Intelligent Guess (xml, json, script, or html))
global : booléen	**true** ou false. Whether to trigger global AJAX event handlers for this request. The default is true. Set to false to prevent the global handlers like ajaxStart or ajaxStop from being triggered. This can be used to control various AJAX Events.
ifModified : booléen	true ou **false** Allow the request to be successful only if the response has changed since the last request.
username : "nomDuUser"	Le nom de l'utilisateur utilisé pour l'authentification HTTP
password : "motDePasse"	Le mot de passe utilisé pour l'authentification HTTP
processData : booléen	**true** ou false By default, data passed in to the data option as an object (technically, anything other than a string) will be processed and transformed into a query string, fitting to the default content-type "application/x-www-form-urlencoded". If you want to send a DOMDocument, or other non-processed data, set this option to false.
scriptCharset : "charSet"	Seulement pour les requêtes 'jsonp' ou 'script' et GET. La requête force un jeu de caractères. Nécessaire si les jeux de caractères sont différents entre le client et le serveur.
timeout : entier	En millisecondes
method : "méthode"	**GET ou POST ou PUT**
type : "type"	GET ou POST ou PUT
xhr : fonction	Fonction exécutée lors de la création de l'objet XMLHttpRequest. Callback for creating the XMLHttpRequest object. Defaults to the ActiveXObject when available (IE), the XMLHttpRequest otherwise. Override to provide your own implementation for XMLHttpRequest or enhancements to the factory.
beforeSend : fonction	Fonction exécutée avant d'envoyer la requête … pour changer les en-têtes par exemple
complete : fonction	Fonction exécutée lorsque la requête est terminée Renvoie les données dans data.responseText ...
success : fonction	**Fonction exécutée en cas de succès (paramètre : data)** **Renvoie les données dans data sous forme de texte**
error : fonction	**Fonction exécutée en cas d'erreur paramètres : XHR, statut, erreur**
dataFilter : fonction	Fonction exécutée pour filtrer la réponse

Deprecation Notice: The jqXHR.**success**(), jqXHR.**error**(), and jqXHR.**complete**() callbacks are deprecated as of jQuery 1.8. To prepare your code for their eventual removal, use jqXHR.**done**(data), jqXHR.**fail**(xhr, statut, erreur), and jqXHR.**always**() instead.

async : false est aussi obsolète (cf https://xhr.spec.whatwg.org/).

Et d'autres encore cf http://api.jquery.com/jQuery.ajax/

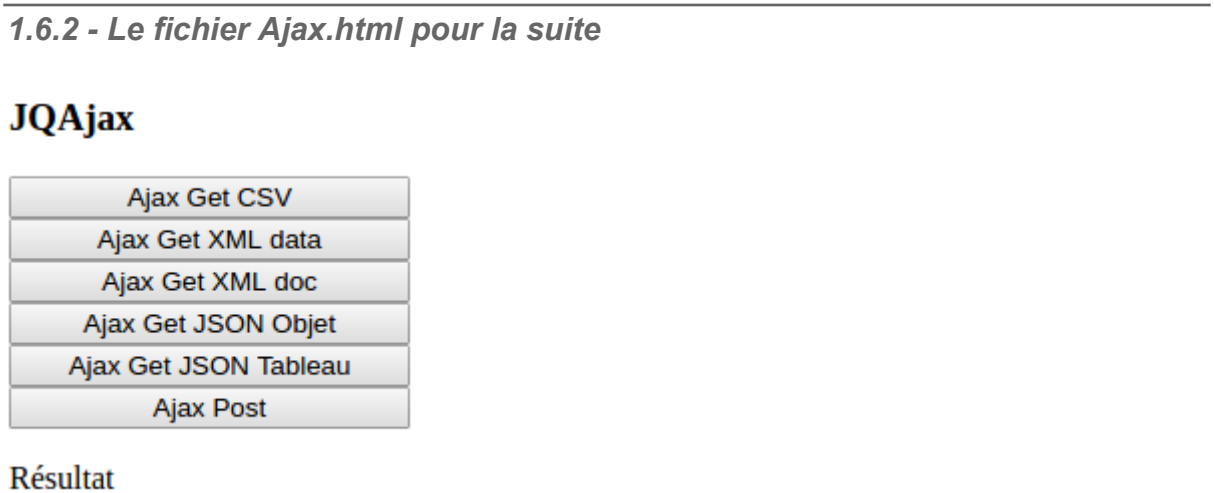
Le nouveau modèle :

```
let jqXHR = $.ajax({
    type: "TYPE",
    url: "URL",
    dataType: "type"
});

jqXHR.done(function(data [,statut, xhr]) {
    // Code
});

jqXHR.fail(function(xhr, statut, erreur) {
    // Code
    console.log(xhr.status + " : " +  xhr.statusText);
});
```

Note pour fail :

xhr :
  xhr.status : code erreur (404 par exemple).
  xhr.statusText (par exemple Not Found).
statut : statut du texte « error ».
erreur : le texte d'erreur (Not Found par exemple).

1.6.2 - Le fichier Ajax.html pour la suite



et d'autres boutons …

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
        <style>
            input[type="button"] {
                display: block;
                width: 200px;
            }
        </style>
        <title>JQAjax.html</title>
    </head>

    <body>
        <div>
            <h3>JQAjax</h3>
            <input type="button" value="AJAX Get CSV" id="btAjaxGetCSV" />
            <input type="button" value="AJAX Get XML data"
id="btAjaxGetXmlData" />
            <input type="button" value="AJAX Get XML doc"
id="btAjaxGetXmlDoc" />
            <input type="button" value="AJAX Get JSON Objet"
id="btAjaxJsonObjet" />
            <input type="button" value="AJAX Get JSON Tableau"
id="btAjaxJsonTableau" />
            <input type="button" value="AJAX Post" id="btAjaxPost" />
            <p id="pResultat">Résultat</p>
        </div>

        <script src="../jquery/jquery.js"></script>
        <script src="../js/JQAjax.js"></script>
    </body>
</html>
```

1.6.3 - La fonction init() du fichier JQAjax.js

```
// ------------
function init() {
    $("#btAjaxGetCSV").on("click", requeterAjaxGetCSV);
    $("#btAjaxGetXmlData").on("click", requeterAjaxGetXmlData);
    $("#btAjaxGetXmlDoc").on("click", requeterAjaxGetXmlDoc);
    $("#btAjaxJsonObjet").on("click", requeterAjaxGetJsonObjet);
    $("#btAjaxJsonTableau").on("click", requeterAjaxGetJsonTableau);
} /// init
```

1.6.4 - Récupérer des données CSV

Objectif : Afficher le contenu d'un fichier CSV dans un paragraphe.

Données CSV

1;Tintin
2;Milou;
3;Haddock
4;Dupont
5;Dupond;
6;Tournesol
7;Castafiore

Le code JavaScript

```
// ------------------------
function requeterAjaxGetCSV() {

    let jqXHR = $.ajax({
        type: "GET",
        url: "../ressources/csv/tintin.txt",
        dataType: "text"
    });

    jqXHR.done(function(data) {
        let regex = /\n/g;
        data = data.replace(regex, "<br>");
        $("#pResultat").html(data);
    });

    jqXHR.fail(function(xhr, statut, erreur) {
        $("#pResultat").html("Erreur : " + xhr.status);
    });
} /// requeterAjaxGetCSV
```

1.6.5 - Récupérer des données JSON : un objet

Objectif : afficher le contenu d'un fichier JSON.

Paris [75000] : 2000000

Le code JavaScript

```
// ------------------------------
function requeterAjaxGetJsonObjet() {

    let jqXHR = $.ajax({
        type: "GET",
        url: "../ressources/json/ville.json",
        dataType: "json"
    });

    jqXHR.done(function(data) {
        let lsResultat = data.ville + " [" + data.codePostal + "] : " +
data.habitants;
        $("#pResultat").html(lsResultat);
    });

    jqXHR.fail(function(xhr, statut, erreur) {
        $("#pResultat").html("Erreur : " + xhr.status);
    });
} /// requeterAjaxGetJsonObjet
```

1.6.6 - Récupérer des données JSON : un tableau d'objets

Objectif : afficher le contenu d'un fichier JSON.

Paris : 75000
Lyon : 69000
Marseille : 13000

Le code JavaScript

```
// --------------------------------
function requeterAjaxGetJsonTableau() {

    let jqXHR = $.ajax({
        type: "GET",
        url: "../ressources/json/villes.json",
        dataType: "json"
    });

    jqXHR.done(function(data) {
        let lsResultat = "";

//        // Recuperation des elements un par un avec each()
//        $.each(data, function(cle, valeur) {
//            // cle est l'indice dans le tableau
//            // valeur est un objet JSON
//            lsResultat += "[" + cle + "] " + valeur.codePostal + " : " +
valeur.Ville + ", " + valeur["habitants"] + " habitants<br/>";
//        });

        // Recuperation des elements un par un avec un for-)
        for (let i = 0; i < data.length; i++) {
            lsResultat += data[i].ville;
            lsResultat += " : ";
            lsResultat += data[i].codePostal;
            lsResultat += "<br>";
        }
        $("#pResultat").html(lsResultat);
    });

    jqXHR.fail(function(xhr, statut, erreur) {
        $("#pResultat").html("Erreur : " + xhr.status);
    });

} /// requeterAjaxGetJsonTableau
```

1.6.7 - Récupérer des données XML de type Data

Objectif : afficher le contenu d'un document XML Data.

Marseille
Bordeaux
Lille
Paris XI
Paris XII
Rome
Milan
Madrid
Barcelone

```
// ----------------------------
function requeterAjaxGetXmlData() {
    let jqXHR = $.ajax({
        type: "GET",
        url: "../ressources/xml/villesData.xml",
        dataType: "xml"
    });

    jqXHR.done(function(data) {
        let lsVilles = "";
        $(data).find("ville").each(function() {
            // --- Recuperation de la valeur de l'attribut
            let lsNomVille = $(this).attr("nom_ville");
            lsVilles += lsNomVille + "<br/>";
        });
        $("#pResultat").html(lsVilles);
    });

    jqXHR.fail(function(xhr, statut, erreur) {
        $("#pResultat").html("Erreur : " + xhr.status);
    });
} /// requeterAjaxGetXmlData
```

1.6.8 - Récupérer des données XML de type Doc

Objectif : afficher le contenu d'un document XML Doc.

Paris
Lyon
Cabourg
Caen
Rome
Turin
Madrid
Barcelone

```
// ---------------------------
function requeterAjaxGetXmlDoc() {
    let jqXHR = $.ajax({
        type: "GET",
        url: "../ressources/xml/villesDocument.xml",
        dataType: "xml"
    });

    jqXHR.done(function(data) {
        let lsVilles = "";
        $(data).find("nom_ville").each(function() {
            // --- Recuperation de la valeur du noeud text
            lsVilles += $(this).text() + "<br>";
        });
        $("#pResultat").html(lsVilles);
    });

    jqXHR.fail(function(xhr, statut, erreur) {
        $("#pResultat").html("Erreur : " + xhr.status);
    });
} /// requeterAjaxGetXmlDoc
```

1.6.9 - Poster des données avec $.ajax()

Objectif

Insérer des données dans la table [villes] d'une BD MySQL.

CP : 33000 Ville : Bordeaux ID pays : 33 Enregistrer

1 enregistrement(s) ajouté(s)

CP : 33000 Ville : Bordeaux ID pays : 33 Enregistrer

SQLSTATE[23000]: Integrity constraint violation: 1062 Duplicate entry '33000' for key 'PRIMARY'

Éléments de Syntaxe :

```
$.ajax
({
  type: "POST",
  url: "url",
  dataType: "text",
  data: "attribut1=valeur1&attribut2=valeur2", (*)
  done: function(data)
  {
  },
  fail: function(xhr, statut, erreur) (**)
  {
  }
});
```

(*)
data: {attribut1:valeur1, attribut2:valeur2}

(**)
xhr :
  xhr.status : code erreur (404 par exemple).
  xhr.statusText (par exemple Not Found).
statut : statut du texte « error ».
erreur : le texte d'erreur (Not Found par exemple).

JQAjaxPost.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>JQAjaxPost</title>
        <meta charset="UTF-8">
    </head>

    <body>
        <label>CP : </label>
        <input type="text" id="itCP" value="06500" size="5" />
        <label>Ville : </label>
        <input type="text" id="itNomVille" value="Menton" />
        <label>ID pays : </label>
        <input type="text" id="itIdPays" value="33" size="4" />

        <button id="btInsert">Enregistrer</button>
        <br/><br>
        <label id="lblMessage"></label>

        <script src="../jquery/jquery.js"></script>
        <script src="../js/JQAjaxPost.js"></script>
    </body>
</html>
```

JQAjaxPost.js

```
/*
 * JQAjaxPost.js
 */
// ------------
function init() {
    $("#btInsert").on("click", insert);
} /// init

// --------------
function insert() {
    let lsCp = $("#itCP").val();
    let lsNomVille = $("#itNomVille").val();
    let lsIdPays = $("#itIdPays").val();

    let jqXHR = $.post(
            "../php/VillesInsert.php",
            {cp: lsCp, nom_ville: lsNomVille, id_pays: lsIdPays},
    "text"
            );

    jqXHR.done(function(data) {
        $("#lblMessage").html(data);
    });

    jqXHR.fail(function(xhr, statut, erreur) {
        let sTexte = xhr.status + ":" + xhr.statusText;
        $("#lblMessage").html(sTexte);
    });

} /// insert

/// -------------------
$(document).ready(init);
```

1.6.10 – Exercice : modifier un pays dans la table [pays]

Modifier un pays via la saisie dans un <input>.

Le script PHP

```
<?php
    // --- paysUpdate.php
    $lsMessage = "";

    try {
        $lcn = new PDO("mysql:host=localhost;dbname=cours", "root", "");
        $lcn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $lcn->exec("SET NAMES 'UTF8'");

        $lsSQL = "UPDATE pays SET nom_pays=? WHERE id_pays=?";
        $lcmd = $lcn->prepare($lsSQL);
        $lcmd->execute(array($_POST["nom_pays"], $_POST["id_pays"]));
        $lsMessage = $lcmd->rowcount() . " enregistrement(s) modifié(s)";

        $lcn = null;
    }
    catch(PDOException $e) {
        $lsMessage = $e->getMessage();
    }

    echo $lsMessage;
?>
```

1.6.11 – $.ajax POST générique

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
        <style>
            .etiquette{
                display: block;
            }
        </style>
        <title>JQAjaxPostGenerique.html</title>
    </head>

    <body>
        <div>
            <h3>JQAjaxPostGenerique</h3>

            <p>
                <label class="etiquette">Cp </label>
                <input id="itCP" type="text" value="06000" />
            </p>
            <p>
                <label  class="etiquette">Ville </label>
                <input id="itNomVille" type="text" value="Nice" />
            </p>
            <p>
                <label  class="etiquette">ID pays : </label>
                <input type="text" id="itIdPays" value="033" />
            </p>
            <p>
                <input id="btAjouter" type="button" value="Ajouter" />
            </p>

            <p>
                <label id="lblMessage"></label>
            </p>
        </div>

        <script src="../jquery/jquery.js"></script>
        <script src="../js/JQAjaxPostGenerique.js"></script>

    </body>
</html>
```

```
/*
 * JQAjaxPostGenerique.js
 */

let tElements = ["itCP", "itNomVille", "itIdPays"];
let tAttributs = ["cp", "nom_ville", "id_pays"];
let url = "../php/VillesInsert.php";
let elMessage = "lblMessage";

// ------------
function init() {
    $("#btAjouter").click(function() {
        ajouter(tAttributs, tElements, url, elMessage)
    });
} /// init



// --- elements, attributs de requete, url, element Message
function ajouter(tAttributs, tElements, asURL, elMessage) {
    //data: "cp=" + lsCp + "&nom_ville=" + lsNomVille,
    //data: "cp=06500&nom_ville=Menton",
    let lsData = "";
    for (let i = 0; i < tAttributs.length; i++) {
        lsData += tAttributs[i] + "=" + $("#" + tElements[i]).val() + "&";
    }

    // --- Elimination du dernier & (Facultatif)
    lsData = lsData.substr(0, lsData.length - 1);

    // --- La requete
    let jqXHR = $.ajax
            ({
                type: "POST",
                url: asURL,
                dataType: "text",
                data: lsData
            });

    jqXHR.done(function(data) {
        $("#" + elMessage).html("Requête XHR réussie<br/>" + data);
    });

    jqXHR.fail(function(xhr, statut, erreur) {
        let sTexte = xhr.status + ":" + xhr.statusText;
        $("#" + elMessage).html(sTexte);
    });
} /// ajouter

// ---------------------
$(document).ready(init);
```

1.6.12 - Récupérer des données SQL

Objectif

Récupérer des données SQL avec la méthode $.ajax().

Exemple

Récupérer la liste des villes : exécuter une requête GET sans paramètre sur une source PHP.

jqAjaxSQLGet

06000;Nice 06500;Menton 13000;Marseille 24200;Sarlat

Eléments de Syntaxe :

```
$.ajax
({
  type: "GET",
  url: "url",
  dataType: "text",
  success: function(data)
  {
  }
});
```

Script

jqAjaxSQLGet.html

```
<script>
    // ------------
    function init() {
        let jqXHR = $.ajax({
            type: "GET",
            url: "../php/villesSelect.php",
            dataType: "text",
        });

        jqXHR.done(function(data) {
            $("#pResultat").html(data);
        });

        jqXHR.fail(function(xhr, statut, erreur) {
            $("#pResultat").html(xhr.status);
        });
     } /// init

     $(document).ready(init);
</script>
```

Rappel :

dataType : type des données en retour

/xhr.status : 404
xhr.statusText : Not Found
statut : error
erreur : Not Found par exemple

1.6.13 - Récupérer des données SQL avec passage de paramètre(s)

Objectif

Afficher les villes d'un pays.

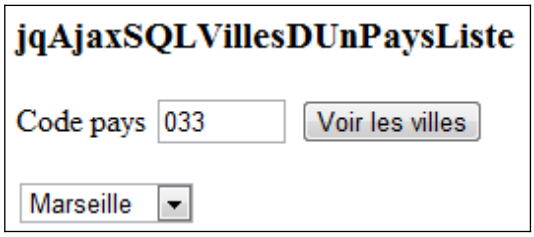
Script : jqAjaxSQLGetParametre.html

```
<script src="../../jquery/jquery.js"></script>
<script>
    // ------------
    function init() {
        $("#btVoir").click(voir);
    }

    // ------------
    function voir() {
        $.ajax({
            type: "GET",
            url: "../php/villesDUnPays.php",
            data: { id: $("#itCP").val() },
            // ou aussi "id=033&…" donc "id=" + $("#itCP").val() + "&
            dataType: "text",
            success: function(data) {
                $("#pResultat").html(data);
            },
            error: function(xhr, statut, erreur) {
                $("#pResultat").html("Erreur");
            }
        });
    }

    // ---------------------
    $(document).ready(init);
</script>
```

```
<body>
<div id="centre">
    <h3>jqAjaxSQLGetParametre</h3>

    <label>Code pays </label>
    <input id="itCP" name="itCP" type="text" value="033" size="5"/>
    <input id="btVoir" name="btVoir" type="button" value="Voir les villes"
/>
    <br/>
    <p id="pResultat"></p>
</div>
</body>
```

1.6.14 – Exercice : remplir une liste déroulante à partir de données SQL

Afficher le même résultat dans une liste déroulante.



1.6.15 - $.ajax GET générique

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>JQAjaxGetGenerique</title>
    </head>

    <body>
        <div>
            <h3>JQAjaxGetGenerique</h3>

            <label>Code pays </label>
            <input id="itCP" type="text" value="033" size="5"/>
            <input id="btVoirVilles" type="button" value="Voir les villes"
/>
            <br/>
            <input id="btVoirPays" type="button" value="Voir les pays" />

            <p id="pResultat"></p>
            <label id="lblMessage"></label>
        </div>

        <script src="../jquery/jquery.js"></script>
        <script src="../js/JQAjaxGetGenerique.js"></script>
    </body>
</html>
```

```
/*
 * JQAjaxGetGenerique.js
 */

let elResultat = "pResultat";
let elMessage = "lblMessage";

// ------------
function init() {
    $("#btVoirVilles").click(function() {
        let tElements = ["itCP"];
        let tAttributs = ["id"];
        let url = "../php/VillesDUnPays.php";
        voir(tAttributs, tElements, url, elResultat, elMessage);
    });

    $("#btVoirPays").click(function() {
        let tElements = null;
        let tAttributs = null;
        let url = "../php/PaysSelect.php";
        voir(tAttributs, tElements, url, elResultat, elMessage);
    });
} /// init

// ----------
function voir(tAttributs, tElements, asURL, elResultat, elMessage) {
    // --- Creation de la chaine attribut=valeur de la requete
    let lsData = "";
    if (tAttributs != null) {
        for (let i = 0; i < tAttributs.length; i++) {
            lsData += tAttributs[i] + "=" + $("#" + tElements[i]).val() +
"&";
        }
        // --- Elimination du dernier & (Facultatif)
        lsData = lsData.substr(0, lsData.length - 1);
    }

    // --- La requete
    let jqXHR = $.ajax({
        type: "GET",
        url: asURL,
        data: lsData,
        dataType: "text"
    });

    jqXHR.done(function(data) {
        $("#" + elResultat).html(data);
    });

    jqXHR.fail(function(xhr, statut, erreur) {
        let sTexte = xhr.status + ":" + xhr.statusText;
        $("#" + elMessage).html(sTexte);
    });
} /// voir
// ---------------------
$(document).ready(init);
```

1.6.16 - Exercice : listes liées à partir de données SQL

Listes déroulantes liées : pays, villes, clients.

1.6.17 - $.ajax, XPath et document DATA

jQuery et XPath : http://www.ibm.com/developerworksrary/x-xpathjquery/

Objectif : rechercher le nom d'une ville ou de plusieurs en fonction du CP.

jQuery Ajax et XPath DATA

Recherche : 75012 Valider

Paris XII

```
<!DOCTYPE html>
<html>
    <head>
        <title>jqAjaxEtXPathDATA</title>
        <meta charset="UTF-8">
        <script src="http://localhost/jquery/jquery.js"></script>
    </head>

    <body>
        <h3>jQuery Ajax et XPath DATA</h3>
        Recherche :
        <input type="text" name="itValeur" id="itValeur" value="75011" />
        <input type="button" value="Valider" id="btValider" />
        <br /><p id="pResultat"></p>

        <script>
            // ----------
            function init() {
                $("#btValider").click(extraire);
            }

            // --------------
            function extraire() {
                let source = "../ressources/villesData.xml";
                let expression = "villes/ville[@cp='" + $
("#itValeur").val() + "']";

                // --- Methode AJAX (Type de requete, url, type de retour,
evenement)
                $.ajax({
                    type: "GET",
                    url: source,
                    dataType: "xml",
                    cache: false,
                    success: function(data)
                    {
                        let lsVilles = "";
                        $(data).find(expression).each(function()
                        {
                            // --- Recuperation de la valeur de l'attribut
```

```
                            lsVilles += $(this).attr("nom_ville");
                        });
                        $("#pResultat").html(lsVilles);
                    },
                    error: function(xhr, statut, erreur)
                    {
                        $("#pResultat").html(xhr.status + " : " +
xhr.statusText);
                    }
                });
            }
            // --------------------
            $(document).ready(init);
        </script>
    </body>
</html>
```

1.6.17.1 - $.ajax, XPath et document DOC

Objectif : rechercher le nom d'une ville en fonction du CP.

C'est pas vraiment du XPath puisque l'on utilise contains(). Et c'est imparfait !

let expression = "villes ville:contains('" + $("#itValeur").val() + "') nom_ville";

jQuery Ajax et XPath DOC

Recherche : 69000 Valider

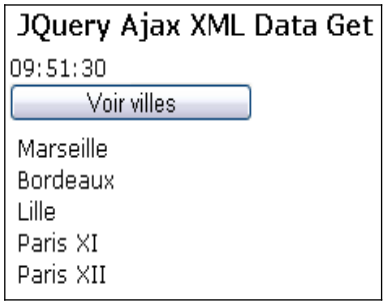
Lyon

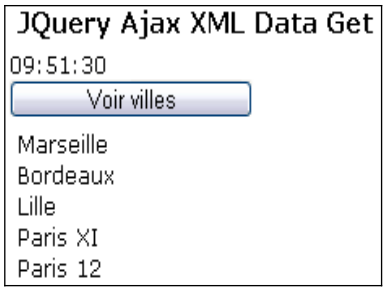
```
<!DOCTYPE html>
<html>
    <head>
        <title>jqAjaxEtXPathDOC</title>
        <meta charset="UTF-8">
        <script src="http://localhost/jquery/jquery.js"></script>
    </head>

    <body>
        <h3>jQuery Ajax et XPath DOC</h3>
        Recherche :
        <input type="text" name="itValeur" id="itValeur" value="69000" />
        <input type="button" value="Valider" id="btValider" />
        <br /><p id="pResultat"></p>

        <script>
            // ----------
            function init() {
                $("#btValider").click(extraire);
            }

            // --------------
            function extraire() {
                // Pour du DOC
                let source = "../ressources/villesDocument.xml";
                //let expression = "villes ville[cp='69000']"; // KO
                let expression = "villes ville:contains('" + $
("#itValeur").val() + "') nom_ville"; // Recupere nom de la ville en
fonction du CP
                //let expression = "ville:first cp"; // Recupere le
premier CP
                //let expression = "ville:last cp"; // Recupere le dernier
CP

                // --- Methode AJAX (Type de requete, url, type de retour,
evenement)
                $.ajax({
                    type: "GET",
                    url: source,
```

```
                    dataType: "xml",
                    cache: false,
                    success: function(data)
                    {
                        let lsVilles = "";
                        $(data).find(expression).each(function()
                        {
                            // --- Recuperation de la valeur du #text
                            lsVilles += $(this).text() + "<br/>";
                        });
                        $("#pResultat").html(lsVilles);
                    },
                    error: function(msg)
                    {
                        $("#pResultat").html(msg.status + " : " +
msg.statusText);
                    }
                });
            }
            // ---------------------
            $(document).ready(init);
        </script>
    </body>
</html>
```

1.6.17.2 - Gérer le cache

L'objectif de ce script est de montrer que lorsque l'on requête avec AJAX seule une partie de la page est mise à jour.
Pour cela la page est une page PHP où l'on affichera l'heure du serveur.
Entre les deux requêtes AJAX les données ont changé (Paris XII -> Paris 12); vous modifiez "alla mano" le contenu du fichier villesData.xml.





Eléments de Syntaxe

```
$.ajax
({
  type: "GET",
  url: "url",
  dataType: "xml",
  cache: false,
  success: function(data)
  {
  }
});
```

Le script

```
<!DOCTYPE html>
<html>

<head>
<meta charset="UTF-8">
<title>jqAjaxXmlDataGet.php</title>

<script src="../../jquery/jquery.js"></script>

<script>
    // ----------------------
    function afficherVilles() {
        // --- Methode AJAX (Type de requete, url, type de retour,
evenement)
        $.ajax({
            type: "GET",
            url: "../ressources/villesData.xml",
            dataType: "xml",
            cache: false,
            success: function(data)
            {
                let lsVilles = "";
                // --- Balayage des elements "ville" du XML en retour
                $(data).find("ville").each(function()
                {
                    // --- Recuperation de la valeur de l'attribut
                    let lsNomVille = $(this).attr("nom_ville");
                    lsVilles += lsNomVille + "<br/>";

                });
                $("#pResultat").html(lsVilles);
            },
            error: function(msg)
            {
                $("#pResultat").html("Erreur");
            }
    });
}
// ---------------------------
$(document).ready(function() {
    afficherVilles();
    $("#btVoirVilles").click(afficherVilles);
});

</script>

</head>

<body>
  <div id="centre">
    <h3>jQuery AJAX XML Data Get</h3>
    <?php echo date("H:i:s"), "<br/>"; ?>
    <input type="button" value="Voir villes" id="btVoirVilles"/>
    <br/><p id="pResultat"></p>
  </div>
```

```
</body>
</html>
```

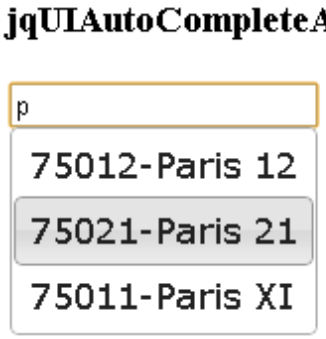
1.6.18 – Exercice : généricisez

Modifiez ce script pour qu'il soit plus générique comme l'était jqAjaxXmlDataGetParam.html.

1.6.19 - AutoComplete, $.ajax et SQL

Objectif

Remplir une liste de suggestions avec de l'AJAX (La table villes de la BD cours).



Démarche

Requêter vers PHP-SQL qui renvoie du texte. La requête est synchrone.
Les enregistrements sont séparés par des \n.
Une variable liste (un tableau) est remplie suite à l'application de la méthode split().
C'est cette variable qui est la source de la méthode autocomplete.

Script

Le script présente 2 versions :

- ✓ Avec jQuery 1.9 et jquery-ui-1.10.0 en version "légère"
- ✓ Avec jQuery 1.9 et jquery-ui-1.10.0 en version "lourde"

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>jqAjaxAutoComplete.html</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

    <link href="../../jquery/ui-1.10.0/themes/base/jquery.ui.all.css"
rel="stylesheet" type="text/css" />

    <script src="../../jquery/jquery.js"></script>

    <script src="../../jquery/ui-1.10.0/jquery.ui.core.js"></script>
    <script src="../../jquery/ui-1.10.0/jquery.ui.widget.js"></script>
    <script src="../../jquery/ui-1.10.0/jquery.ui.button.js"></script>
    <script src="../../jquery/ui-1.10.0/jquery.ui.position.js"></script>
    <script src="../../jquery/ui-1.10.0/jquery.ui.menu.js"></script>
    <script
src="../../jquery/ui-1.10.0/jquery.ui.autocomplete.js"></script>
    <script src="../../jquery/ui-1.10.0/jquery.ui.tooltip.js"></script>

    <!-- TOUT, un peu lourd -->
<!--
<script src="../../jquery/ui-1.10.0/jquery-ui.custom.js"></script>
-->


    <script>
        let liste;
        // ------------
        function init() {
            let liste;

            $.ajax({
                type: "GET",
                url: "../php/villesSelect.php",
                dataType: "text",
                async: true,
                success: function(data)
                {
                    liste = data.split("\n");
                },
                error: function()
                {
                    $("#lblMessage").html("Erreur");
                },
                complete: function()
                {
                    $("#it_suggestions").autocomplete({ source: liste });
                }
```

```
            }); /// $.ajax
        } /// init

        // --------------------
        $(document).ready(init);

    </script>

  </head>

  <body>
    <h3>jqUIAutoCompleteAjax</h3>
    <input type="text" id="it_suggestions" />
    <label id="lblMessage"></label>
  </body>
</html>
```

1.6.20 - Table triée dont la source est SQL

On requête vers un script PHP (villesSelect.php) qui retourne une chaîne CSV du type cp1;ville1\ncp2;ville2.

On ajoute les éléments à la table via DOM.

CP	Ville
14000	Caen
59000	Lille
69000	Lyon
99392	MILAN
92300	Nanterre
92100	Neuilly
75012	Paris 12

jqTableSorterAjaxSQL.html

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <title>jqTableSorterAjaxSQL.html</title>

   <link href="../../jquery/tablesorter/tablesorter.css" rel="stylesheet"
type="text/css" media="print, projection, screen" />

   <script src="../../jquery/jquery.js"></script>
   <script src="../../jquery/tablesorter/jquery.tablesorter.js"></script>

   <script>
       // --- Un table triable remplie avec Ajax
   // ------------
   function init() {
        afficherTable();
   } /// init

   // ---------------------
   function afficherTable() {
      $.ajax({
         type: "GET",
         url: "../php/villesSelect.php",
         dataType: "text",
         async: true,
         success: function(data)
         {
            // --- Creation des <tr><td> facon DOM
            tEnrs = data.split("\n");

            for(i = 0; i < tEnrs.length; i++) {
              let tr = $("<tr>");
              // --- Creation des <td>
              tChamps = tEnrs[i].split(";");
              for(j = 0; j < tChamps.length; j++) {
                 let td = $("<td>");
                 td.html(tChamps[j]);
                 tr.append(td);
              } /// Creation des <td>

              $("#corpsTable").append(tr);
            } /// Creation des <tr><td>

            // --- Les data sont affectees a la table
            // --- On peut rendre la table "triable"
            $("#tableVilles").tablesorter();
         },
         error: function(xhr)
         {
            $("#spMsg").html("Erreur : " + xhr.status);
         }
      });
   } /// afficherTable
```

```
   // ------------------
   $(document).ready(init);

   </script>

</head>

<body>
    <div id="centre">
        <h3>jqTableSorterAjax2</h3>
        <table id='tableVilles' class='tablesorter'>
          <thead>
             <tr><th scope="col">CP</th><th scope="col">Ville</th></tr>
          </thead>
          <tbody id='corpsTable'>

          </tbody>
        </table>
        <span id="spMsg"></span>
    </div>
</body>
</html>
```

1.6.21 - Accordéon, $.ajax et XML

Objectif

Remplir un accordéon avec des données XML (Type document).
Cf dans les annexes le fichier **accordeon.xml**.

Démarche

Création d'une requête AJAX, de type **synchrone**, pour créer et remplir les soufflets.
Création des éléments (pour chaque soufflet, <h5> pour le titre, <a> pour le lien du titre, <p> pour le contenu de chaque soufflet).

Recherche des rubriques ($(data).find("rubrique")) et balayage de la liste des rubriques (each(function() …)).
A l'intérieur de chaque rubrique on recherche d'abord le titre de la rubrique ($(this).find("titre")) puis le contenu de la rubrique ($(this).find("contenu")).
A l'intérieur du contenu de la rubrique on récupère la liste des "enfants" (parent.children()).
On balaie la liste des "enfants" (enfant.each(function()).
Et pour chaque "enfant" on récupère le node text ($(this).text()).

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>jqGetAccordeon</title>

    <link href="../../jquery/ui/themes/base/ui.all.css" rel="stylesheet"
type="text/css" />
    <style type="text/css">
        .soufflet, contenuSoufflet{margin:0; padding:0;}
        .contenuSoufflet ul{list-style-type:none; color:dimgray;}
    </style>

    <script src="../../jquery/jquery.js"></script>
    <script src="../../jquery/ui/ui.core.js"></script>
    <script src="../../jquery/ui/ui.accordion.js"></script>
<script>
// ------------
function init()
// ------------
{
$.ajax
({
   type: "GET",
   url: "../ressources/accordeon.xml",
   dataType: "xml",
   async: false,
   success: function(data)
   {
        $(data).find("rubrique").each(function()
        {
           // --- Creation d'une div pour chaque rubrique
           let divSoufflet = $("<div>");
           // --- L'attribut class du soufflet
           divSoufflet.attr("class","soufflet");
           // --- Creation d'un titre de soufflet
           let h5 = $("<h5>");
           // --- Creation d'une ancre pour le titre du soufflet
           let a = $("<a>");
           a.attr("href", "#");
           // --- Recherche du titre dans l'arbre XML
           let titre = $(this).find("titre");
           // --- Texte de l'ancre
           a.html(titre.text());
           // --- Creation d'un paragraphe pour le contenu du soufflet
           let contenu = $("<p>");
           // --- Creation du contenu
           let lsContenu = "";
           // --- Recuperation du contenu XML
           let elContenu = $(this).find("contenu");
           // --- Recuperation des valeurs des elements enfants XML
           let valeurs = elContenu.children();
           valeurs.each(function()
           {
               lsContenu += $(this).text() + "<br/>";
           });
           // --- Ajout du texte au contenu
```

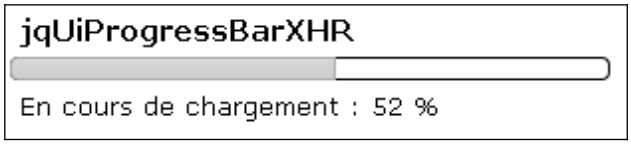
```
            contenu.html(lsContenu);
            // --- Ajout de l'ancre au titre du soufflet
            h5.append(a);
            // --- Ajout du titre au soufflet
            divSoufflet.append(h5);
            // --- Ajout du contenu au soufflet
            divSoufflet.append(contenu);
            // --- Ajout d'un soufflet a l'accordeon
            $("#accordeon").append(divSoufflet);
        });
    } // /success
}); // / $.ajax
$("#accordeon").accordion({ header:"h5" });
}

// ---------------------
$(document).ready(init);

</script>
</head>

<body>
    <div id="centre">
        <h3>ACCORDEON</h3>
        <!-- Copiez votre code HTML perso ici !!! -->
        <div id="accordeon">
        </div> <!-- /div accordeon -->

        <p id="pResultat"></p>
        <label id="lblMessage"></label>
    </div> <!-- /div centre -->
</body>
</html>
```

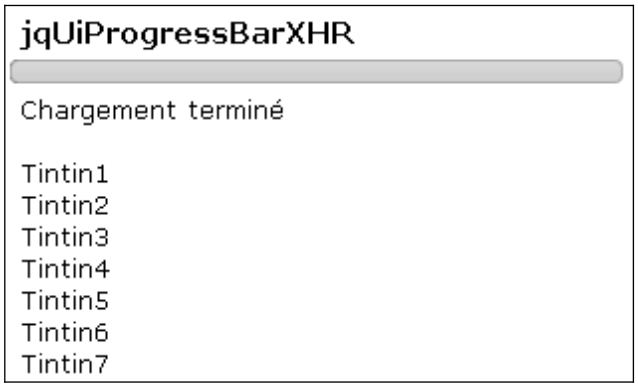
1.6.22 - ProgressBar et XHR

Objectif

Utiliser XHR (XMLHttpRequest) et non pas $.ajax qui sera utilisé au paragraphe suivant. jQuery est bien entendu utilisé avec jQuery UI pour la jauge (ProgressBar).

A titre d'exemple nous chargerons 100 000 enregistrements d'une table en affichant la progression du chargement via une jauge.





Démarche

Créer une table nommée gt avec 100 000 enregistrements. Cf le script nommé gtCreateInserts.php.

Calculer la taille, en octets, de la table gt. Cf le script nommé gtSize.php.

Récupérer les données de la table gt. Cf le script nommé gtSelect.php.

Exécuter deux requêtes asynchrone (Une pour la taille en octets, une autre pou les données).

Tester le statut (200:OK) et l'état (3:envoi de données) de la requête :

```
if(xhr.readyState==3 && xhr.status==200)
```

Calculer le nombre d'octets reçus

```
xhr.responseText.length
```

Calculer le pourcentage d'octets reçus.

Faire progresser la jauge (l'attribut value veut un numérique).

```
$("#jauge").progressbar("value", pourCent);
```

Afficher le pourcentage (avec 2 chiffres après la virgule; toFixed() renvoie une String).

```
$("#" + asIdConteneur).html("En cours de chargement : " +
pourCent.toFixed(0) + " %");
```

Script : jqUiProgressBarXHR.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>jqUiProgressBarXHR</title>

    <link href="../css/_coursJQuery.css" rel="stylesheet"
type="text/css" />
    <link href="../../jquery/ui/themes/base/ui.all.css" rel="stylesheet"
type="text/css" />

    <style type="text/css">
        #jauge{ width:300px; }
        .ui-progressbar{ border-color:black; height:4px; padding:0; }
        .ui-progressbar .ui-progressbar-value { height:4px;  }
    </style>

    <script src="../../jquery/jquery.js"></script>
    <script src="../../jquery/ui-1.10.0/jquery-ui.custom.js"></script>

    <script>
    let i = 0;
    let iiTaille = 0;

    // ------------
    function init()
    // ------------
    {
        $("#jauge").progressbar(
        {
            value: 0 // En pourcentage
        } );

        // --- Calcule et affiche la taille de la table
        executerRequete("../php/gtSize.php", "pResultat", "taille");
        // --- Affiche la table
        executerRequete("../php/gtSelect.php", "pResultat", "contenu");
    } // / init

    // ---------------------------------------------------
    function executerRequete(asUrl, asIdConteneur, asType)
    // ---------------------------------------------------
    {
        let xhr = new XMLHttpRequest();

        // --- L'evenement onReadyStateChange
        xhr.onreadystatechange = function()
        {
            recupererContenuTXT(xhr, asIdConteneur, asType);
        };
        // --- Cela vide le cache et on a rien !!!
        //xhr.setRequestHeader("Cache-Control","no-cache");
        xhr.open('GET', asUrl, true); // --- Asynchrone
        xhr.send(null);
```

```
        // --- Ne passe jamais ici en mode asynchrone, en mode synchrone
si
        //$("#" + asIdConteneur).html("En-têtes : " +
xhr.getAllResponseHeaders() + "<br/>)";
        // --- Ne passe jamais ici en mode asynchrone, en mode synchrone
si
        //$("#" + asIdConteneur).html("FIN");
    } // /executerRequete

    // ----------------------------------------------------
    function recupererContenuTXT(xhr, asIdConteneur, asType)
    // ----------------------------------------------------
    {
        // --- Donnees accessibles en partie et OK
        if(xhr.readyState==3 && xhr.status==200)
        {
            // --- Si l'on est en train d'extraire les donnees, on affiche
l'etat du chargement
            if(asType=="contenu") {
                // --- Recupere la taille de la "bufferisation"
                let tailleReponseText = xhr.responseText.length;
                // --- Calcul en pourcentage
                let pourCent = (tailleReponseText / iiTaille) * 100;
                $("#jauge").progressbar("value", pourCent);
                // --- Affichage du pourcentage de chargement et de la
progression dans la jauge
                $("#" + asIdConteneur).html("En cours de chargement : " +
pourCent.toFixed(0) + " %");
            } /// contenu
        } /// readyState 3

        // --- Donnees accessibles totalement et OK
        if(xhr.readyState==4 && xhr.status==200)
        {
            // --- Si l'on etait en train de calculer la taille de la
table en octets
            // --- on recupere cette taille dans une variable d'instance
            // --- et on affiche (eventuellement) cette taille
            if(asType=="taille") {
                iiTaille = xhr.responseText;
            } /// requete taille quand getSize() est execute
            // --- Si l'on etait en train d'extraire les donnees, on
affiche les donnees
            if(asType=="contenu") {
                $("#" + asIdConteneur).html("Chargement terminé<br/><br/>"
+ xhr.responseText);
                // --- Pour etre sur que visuellement la jauge est remplie
                $("#jauge").progressbar("value", 100);
            } /// requete sur contenu
        } /// readyState 4
    } // /recupererContenuTXT

    // ---------------------
    $(document).ready(init);
    </script>
</head>

<body>
```

```
    <div id="centre">
        <h3>jqUiProgressBarXHR</h3>
        <div id="jauge" class="ui-progressbar"></div>
        <p id="pResultat"></p>
    </div>
</body>
</html>
```

1.6.23 - ProgressBar et $.ajax

Objectif

Le même que précédemment

Démarche

1.6.24 - Les événements AJAX

Objectif

Réagir aux événements AJAX :

- ✓ success (done depuis la version 1.8),
- ✓ error (fail depuis la version 1.8),
- ✓ start,
- ✓ stop,
- ✓ complete (always depuis la version 1.8),
- ✓ beforeSend,
- ✓ dataFilter.

Script

```
<script>
// ------------
function init() {
     // $.ajax(Type de requete, url, type de retour, evenement)
     $.ajax(
     {
           type: "GET",
           url: "villesData.xml",
           dataType: "xml",
           success: function(data)
           {
                 let lsVilles = "<strong>Success</strong>";
                 // --- Les elements "ville" du XML en retour
                 $(data).find("ville").each(function()
                 {
                       let lsNomVille = $(this).attr("nom_ville");
                       lsVilles += "<br/>" + lsNomVille;
                 });
                 $("#pResultat").html($("#pResultat").html() + lsVilles);
           },
           error: function()
           {
                 $("#pResultat").html($("#pResultat").html() +
"<br/><strong>Erreur</strong>");
           },
           start: function()
           {
                 $("#pResultat").html($("#pResultat").html() +
"<br/><strong>Start</strong>");
           },
           stop: function()
           {
                 $("#pResultat").html($("#pResultat").html() +
"<br/><strong>Stop</strong>");
           },
           complete: function()
```

```
            {
                  $("#pResultat").html($("#pResultat").html() +
"<br/><strong>Complete</strong>");
            }
      });
}

$(document).ready(init);
</script>
```

1.6.25 - Gestion d'erreur et autres « externe »

http://api.jquery.com/jQuery.ajax/#jqXHR

1.6.25.1 - Syntaxes

```
let jqXHR = $.ajax({
   // code
});
```

```
jqXHR.event(function([data]) {
```

Les « events » sont :

done : remplace success
always : remplace complete
fail : en cas d'erreur

deprecated ≥ 1.8
success : en cas de succès
complete : en cas de succès après success
error : en cas d'erreur

1.6.25.2 - Code

```
/*
 * Villes.js
 */

function init() {
    $("#btVoirVilles").click(requeteGetVilles);
} /// init

function requeteGetVilles() {

    let jqXHR = $.ajax({
        method: "GET",
        url: "../php/VillesSelectAjaxLocal.php",
        dataType: "text",
        async: true
    });

    jqXHR.done(function(data) {
        console.log("*** done ***");
        console.log("Données<br>");
        console.log(data);
        let lsContenu = "";
        lsContenu += "Catégories with AJAX<br>";
        lsContenu += data;
        lsContenu = lsContenu.replace(/\n/g, '<br>');
        $("#listeCategories").html(lsContenu);

    });
    jqXHR.always(function() {
        console.log("*** always ***");
    });
    jqXHR.progress(function() {
        console.log("*** progress ***");
    });
    jqXHR.fail(function(xhr, textStatus, erreur) {
        // xhr.statusText renvoie la meme chose que erreur.
        console.log("*** fail ***");
        console.log(textStatus + " - " + erreur + " - " + xhr.status);
    });

    jqXHR.success(function(data) {
        // deprecated jq >= 1.8
        console.log("*** success ***");
        console.log("Données<br>");
        console.log(data);
    });
    jqXHR.complete(function(data) {
        // deprecated jq >= 1.8
        console.log("*** complete ***");
        console.log("Données<br>");
        console.log(data);
    });
    jqXHR.error(function(xhr, textStatus, erreur) {
        // deprecated jq >= 1.8
```

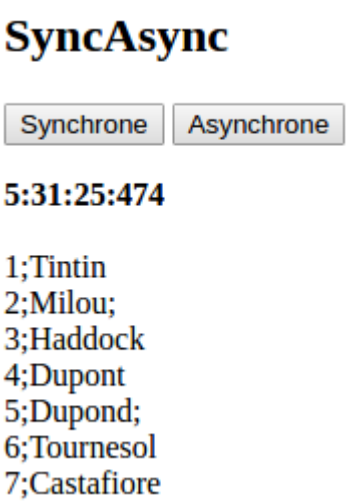
```
        console.log("*** error ***");
        console.log(textStatus + " - " + erreur + " - " + xhr.status);
    });

} /// requeteGetVilles

// --------
$(document).ready(init);
```

1.6.26 - Requête Synchrone et requête Asynchrone

L'objectif de ce paragraphe est de montrer les différences d'ordre de traitement entre une requête ~~synchrone~~ et une requête asynchrone. L'ordre des appels est le même (requête vers le serveur pour récupérer les données puis affichage de l'heure du client) dans les 2 cas avec bien entendu un affichage inversé.
Dans le cas d'un appel synchrone l'heure ne sera affichée que lorsque toutes les données auront été récupérées du serveur.
Dans le cas d'un appel asynchrone l'heure sera affichée que lorsque toutes les données auront été récupérées du serveur.

Synchrone

Asynchrone



```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
        <title>SyncAsync.html</title>
    </head>

    <body>
        <article id="article">
            <h1>SyncAsync</h1>
            <p>
                <input type="button" value="Synchrone" id="btSynchrone" />
                <input type="button" value="Asynchrone"
id="btAsynchrone" />
            </p>

            <p id="pResultats"></p>
        </article>

        <script src="../jquery/jquery.js"></script>
        <script src="../js/SyncAsync.js"></script>
    </body>
</html>
```

```
/*
 * SyncAsync.js
 */

// ----------
function init() {
    $("#btSynchrone").on("click", function() {
        $("#pResultats").html("");
        reqSynchrone();
        getHeure();
    });
    $("#btAsynchrone").on("click", function() {
        $("#pResultats").html("");
        reqAsynchrone();
        getHeure();
    });
} /// init

// ------------------
function reqSynchrone() {
    let jqXHR = $.ajax({
        async: false,
        type: "GET",
        url: "../ressources/tintin.txt",
        dataType: "text"
    });

    jqXHR.done(function(data) {
        let regex = /\n/g;
        data = data.replace(regex, "<br>");
        let pResultats = $("#pResultats").html() + "<br>";
        pResultats += data;
        $("#pResultats").html(pResultats);
    });
} /// reqSynchrone

// -------------------
function reqAsynchrone() {
    let jqXHR = $.ajax({
        async: true,
        type: "GET",
        url: "../ressources/tintin.txt",
        dataType: "text"
    });

    jqXHR.done(function(data) {
        let regex = /\n/g;
        data = data.replace(regex, "<br>");
        let pResultats = $("#pResultats").html() + "<br>";
        pResultats += data;
        $("#pResultats").html(pResultats);
    });
} /// reqAsynchrone

// --------------
function getHeure() {
    let d = new Date();
```

```
    let heure = d.getHours() + ":" + d.getMinutes() + ":" + d.getSeconds()
+ ":" + d.getMilliseconds();
    let pResultats = $("#pResultats").html();
    pResultats += "<b>" + heure + "<b><br>";
    $("#pResultats").html(pResultats);
} /// getHeure

// --------------------
$(document).ready(init);
```

1.7 - Utiliser $.getJSON()

https://api.jquery.com/jquery.getjson/

1.7.1 - Présentation

Objectif

Load JSON-encoded data from the server using a GET HTTP request.

Récupérer des données JSON (JavaScript Object Notation).
Les structures JSON sont des tableaux ou des objets (cf le support JSON pour plus de détails).

Note : validateur JSON à l'URL http://jsonlint.com/

Syntaxes

```
$.getJSON(url [, data] [, success(data, textStatus, jqXHR)])
```

```
let jqXHR = $.getJSON(url);

jqXHR.done(function(data) {
});
```

Paramètre	Description
url	URL requêtée qui renvoie des données JSON (fichier JSON, ressource PHP renvoyant du JSON, …)
data	Données sous forme de String JSON
success	Fonction de callback

Ceci est un raccourci de :

```
$.ajax({
  url: url,
  dataType: 'json',
  data: data,
  success: success
});
```

La fonction done() renvoie un objet JSON ou un tableau JSON et parse automatiquement la réponse avec la fonction parseJSON().
La fonction parseJSON(chaîne au format JSON) renvoie un objet JSON.
Donc il ne faut surtout pas l'utiliser dans son code.

1.7.2 - Un JSON de type objet

Exemple

jqJSON

[Ville] Paris
[CodePostal] 75000
[Habitants] 2000000

Pour le fichier ville.json cf les annexes.

Le script HTML

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>JQGetJSONObjet.html</title>

    <script src="../jquery/jquery.js"></script>
    <script src="../js/JQGetJSONObjet.js"></script>
</head>

<body>
    <div id="centre">
        <h3>jqJSON</h3>
        <p id="pResultat"></p>
    </div>
</body>
</html>
```

Le script JavaScript

```
/*
 * JQGetJSONObjet.js
 */

// ------------
function init() {

    let jqXHR = $.getJSON("../ressources/json/ville.json");

    jqXHR.done(function(data) {
        console.log(data);
        let lsResultat = data.Ville + " [" + data.codePostal + "] : " +
data.habitants;
        $("#pResultat").html(lsResultat);
    });
} /// init

// --------------------
$(document).ready(init);
```

1.7.3 - Un JSON de type tableau d'objets

jqJSON2

[0] 75000 : Paris, 2 000 000 habitants
[1] 69000 : Lyon, 450 000 habitants
[2] 13000 : Marseille, 800 000 habitants

Pour le fichier villes.json cf les annexes.

```
<!DOCTYPE html>
<!--
JQGetJSONTableau.html
-->
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
        <link rel="stylesheet" type="text/css" href="../css/style.css">
        <title>JQGetJSONTableau.html</title>
    </head>

    <body>
        <div>
            <h3>JQGetJSONTableau</h3>
            <p id="pResultat"></p>
        </div>
        <script src="../jquery/jquery.js"></script>
        <script src="../js/JQGetJSONTableau.js"></script>

    </body>
</html>
```

```
/*
 * JQGetJSONTableau.js
 */
// ------------
function init() {

    let jqXHR = $.getJSON("../ressources/json/villes.json");

    jqXHR.done(function(data) {
        let lsResultat = "";

        // Recuperation des elements un par un avec each()
        $.each(data, function(cle, valeur) {
            // cle est l'indice dans le tableau
            // valeur est un objet JSON
            lsResultat += "[" + cle + "] " + valeur.codePostal + " : " +
valeur.Ville + ", " + valeur["habitants"] + " habitants<br/>";
        });

        // Recuperation des elements un par un avec un for-)
        for (let i = 0; i < data.length; i++) {
            lsResultat += data[i].ville;
            lsResultat += " : ";
            lsResultat += data[i].codePostal;
            lsResultat += "<br>";
        }
        $("#pResultat").html(lsResultat);
    });
} /// init

// --------------------
$(document).ready(init);
```

1.7.4 - Récupérer des données au format JSON produites par du PHP

1.7.4.1 - Le code PHP

Cf les annexes. PaysEnJSON.php.

1.7.4.2 - Le code AJAX

```
/*
 * JQGetJSONFromPHP.js
 */
// ------------
function init() {

    let jqXHR = $.getJSON("../php/PaysSelectEnJSON.php", function(data)
    {
        console.log(data);
    });

    jqXHR.done(function(data) {
        let lsResultat = "";
        console.log(data);
        console.log(data.length);

        // Recuperation des elements un par un avec un for-)
        for (let i = 0; i < data.length; i++) {
            lsResultat += data[i].id_pays;
            lsResultat += " : ";
            lsResultat += data[i].nom_pays;
            lsResultat += "<br>";
        }
        $("#pResultat").html(lsResultat);
    });
} /// init

// --------------------
$(document).ready(init);
```

1.7.4.3 - Le code HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>JQGetJSONFromPHP.html</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
    </head>

    <body>
        <div>
            <h3>JQGetJSONFromPHP</h3>
            <p id="pResultat">Message</p>
        </div>

        <script src="../jquery/jquery.js"></script>
        <script src="../js/JQGetJSONFromPHP.js"></script>

    </body>
</html>
```

1.8 - Utiliser $.getScript()

https://api.jquery.com/jquery.getscript/

Objectif

Charger et exécuter un script .js.

L'intérêt est de pourvoir charger une bibliothèque JavaScript. Cette bibliothèque peut être distante étant donné que le Cross Domaine est autorisé pour les scripts JavaScript.

Certes vous pourriez charger la bibliothèque distante de façon classique :

```
<script src="http://10.57.149.222:8084/PourMobiles/js/script.js"></script>
```

mais cette technique permet de charger une bibliothèque seulement au moment où elle est utilisée.

Syntaxe

$.getScript(String url [, Function callback]) returns XMLHttpRequest

Exemple

Le fichier HTML.

```
<!DOCTYPE html>
<!-- JqgetScript.html -->
<html>
    <head>
        <title>JQgetScript</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
    </head>

    <body>
        <div>
            <h3>JQgetScript</h3>
            <label></label>
            <input type="button" value="Exécuter Script"
id="btExecuterScript" />
            <label id="lblMessage"></label>
        </div>

        <script src="../jquery/jquery.js"></script>
        <script src="../js/JQgetScript.js"></script>
    </body>
</html>
```

Soit avec un appel local, soit avec un appel distant (Cross Domain).

```
/*
 * JQgetScript.js
 */

// ------------
function init() {
    $("#btExecuterScript").click(executerJS);
}

// ------------------
function executerJS() {

    //$.getScript("../js/script.js", function(data, textStatus, jqXHR) {
    $.getScript("http://10.57.149.222:8084/PourMobiles/js/script.js",
function(data, textStatus, jqXHR) {
        console.log("Le script script.js a été exécuté");
        /*
         * data renvoi le script execute sauf s'il est distant; dans ce
cas renvoie undefined.
         */
        console.log("Données : " + data);
        console.log("Statut : " + textStatus);
        console.log("jqXHR : " + jqXHR);

        $("#lblMessage").html(getMessageDistant() + "<br>" +
majuscule("azerty") + "-" + minuscule("YTREZA"));
    }
    );
}

// --------------------
$(document).ready(init);
```

Le fichier script.js à exécuter ... volontairement simple.

```
/*
 * script.js
 */

function majuscule(mot) {
    return mot.toUpperCase();
}

function minuscule(mot) {
    return mot.toLowerCase();
}
```

1.9 - JSONP

1.9.1 - Généralités

Des données JSON (ou autres) ne peuvent être requêtées à partir d'un serveur vers un autre serveur sans une autorisation spécifique (Autorisation CrossDomain).
Le fournisseur de données doit encapsuler les données dans une fonction JavaScript.
Le client doit requêter en précisant que le type de données est du « jsonp » s'il utilise $ajax(). De plus une fonction de même nom doit être présente dans le script appelant.

http://www.ibm.com/developerworksrary/wa-aj-jsonp1/

P pour Padding (encapsulé).
Travailler avec du JSONP permet de requêter sur un autre domaine.

Voilà un exemple de données JSON encapsulées dans une fonction :
Cette ligne de code sera stockée dans un fichier nommé JSONPObject.js sur le serveur.

```
/*
 * JSONPObject.js
 */
getObject({"pays": "France", "capitale": "Paris"});
```

Testez ça dans votre navigateur (cela renvoie le contenu du fichier JS) :

http://192.168.122.1/jQueryAjaxCours/jsonp/JSONPObject.js?callback=getObject

http://192.168.122.1/jQueryAjaxCours/jsonp/JSONPObject.js?callback=?

La requête sera effectuée soit via $.ajax() soit via $.getJSON().

Note importante :
sur la même machine physique localhost, 192.168.122.1, localhost:8084 sont 3 machines logiques différentes.

1.9.2 - Version statique

Le fichier HTML


```
<!DOCTYPE html>
<!--
JQGetJSONP.html
-->
<html>
    <head>
        <title>JQGetJSONP</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
    </head>

    <body>
        <div>
            <h3>JQGetJSONP</h3>

            <button id="btAfficherInfo">Afficher info</button>
            <p id="pObject"></p>
            <hr>

            <!-- Pour l'exercice -->
            <button id="btAfficherInfos">Afficher infos</button>
            <p id="pObjects"></p>
        </div>

        <script src="../jquery/jquery.js"></script>
        <script src="../js/JQGetJSONP.js"></script>

    </body>
</html>
```

Le fichier JS

```
/*
 * JQGetJSONP.js
 */
/*
 * UN OBJET
 */

/**
 *
 */
function afficherJSONPObjectViaAjax() {

    let url = "http://192.168.122.1/jQueryAjaxCours/jsonp/JSONPObject.js?
callback=getObject";

    $.ajax({
        type: "GET",
        url: url,
        dataType: "jsonp"
    });

    jqXHR.done(function(data) {
        // Sollicite la fonction getObject() presente plus bas
    });

} /// afficherJSONPViaAjax


/*
 * Le JSONP "désenrobé"
 * Le nom de la fonction est le même que celui du script du fournisseur
 */
function getObject(data) {
    $("#pObject").html("Pays : " + data.pays + ", Capitale : " +
data.capitale);
} /// getObject

/**
 *
 * @returns {undefined}
 */
function init() {
    $("#btAfficherObject").on("click", afficherJSONPObjectViaAjax);

    // Pour l'exercice
    $("#btAfficherObjects").on("click", afficherJSONPObjectsViaAjax);
} /// init

// --------------------
$(document).ready(init);
```

Exercice : afficher une liste de pays

Le fichier JSONPObject**s**.js

```
/*
 * JSONPObjects.js
 */
getData(
[
    {"pays": "France", "capitale": "Paris"},
    {"pays": "Italie", "capitale": "Rome"},
    {"pays": "Angleterre", "capitale": "Londres"}
]
);
```

1.9.3 - Version dynamique

1.9.3.1 - Objectif

Récupérer dynamiquement des données dans la fonction getObject() vue plus haut.

```
getObject({"pays": "France", "capitale": "Paris"});
```

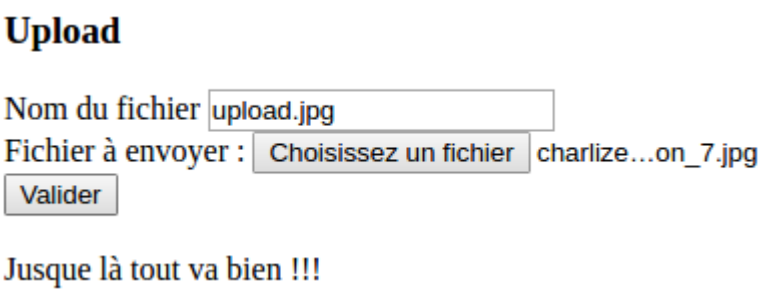
Les données, stockées dans un fichier JSON, sont sur le même serveur que le fichier qui contient la fonction getObject().

1.9.3.2 - Démarche

1.10 - Upload de fichier

L'objectif est de « uploader » un fichier du client vers le serveur.
L'intérêt de passer par AJAX ?
Encore une fois de ne pas recharger toute la page du client pour signifier que tout c'est bien passé ou pas !

1.10.1 - Version utilisant un script JavaScript et un script PHP



Le client saisi le nom du fichier cible (ici « upload.jpg »).
Ensuite il sélectionne le fichier à « uploader » à partir du bouton « Choisissez un fichier » qui un input type="file".
Enfin il clique sur le bouton « Valider » qui est de type « submit ».

La page HTML contient un formulaire en mode POST avec l'action vers le script PHP.

Le script JavaScript inhibe le comportement par défaut du bouton « submit » et émet une requête AJAX vers le script PHP.

Le script PHP « uploade » le fichier et renvoie un code en JSON.

```
<!DOCTYPE html>
<!-- UploadImageAjax.html -->
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
        <title>UploadImageAjax.html</title>
    </head>

    <body>
        <div>
            <h3>UploadImageAjax</h3>

            <form id="formUpload" action="UploadImageAjax.php"
method="post" enctype="multipart/form-data">

                <label for="nomDuFichierCible">Nom du fichier cible
</label>
                <input type="text" name="nomDuFichierCible"
id="nomDuFichierCible" value="upload.jpg" />
                <input type="hidden" name="MAX_FILE_SIZE"
value="1232896" /> <!-- 1 Mo -->

                <br>
                <label>Fichier &agrave; envoyer : </label>
                <input type="file" name="fichierACharger"
id="fichierACharger" />

                <br>
                <input type="submit" />

            </form>

            <br>
            <label id="lblMessage">Message</label>

        </div>

        <script src="../jquery/jquery.js"></script>
        <script src="UploadImageAjax.js"></script>
    </body>
</html>
```

cf plus loin la variante pour ne pas avoir le « vilain » bouton de sélection du fichier !

```
/* UploadImageAjax.js */

$(document).ready(function() {

    $('#formUpload').on('submit', function(e) {

        // Annule le comportement par defaut du navigateur, ie le SUBMIT
        e.preventDefault();

        // Le formulaire
        let form = $(this);
        /*
         The FormData interface provides a way to easily construct a set
of key/value pairs representing form fields and their values, which can
then be easily sent using the XMLHttpRequest.send() method. It uses the
same format a form would use if the encoding type were set to
"multipart/form-data".
         An object implementing FormData can directly be used in a
for...of structure, instead of entries(): for (let p of myFormData) is
equivalent to for (let p of myFormData.entries()).
         */
        let formdata = (window.FormData) ? new FormData(form[0]) : null;
        let data = (formdata !== null) ? formdata : form.serialize();

        let nomDuFichierCible = $('#nomDuFichierCible').val();
        let fichierACharger = $('#fichierACharger').val();

        if (nomDuFichierCible === '' || fichierACharger === '') {
            $("#lblMessage").html('Les champs doivent êtres remplis');
        } else {
            $.ajax({
                url: form.attr('action'),
                type: form.attr('method'),
                contentType: false, // obligatoire pour de l'upload
                processData: false, // obligatoire pour de l'upload
                data: data,
                dataType: 'json',
                success: function(json) {
                    console.log(json);
                    if (json.reponse === '1') {
                        $("#lblMessage").html('Jusque là tout va
bien !!!');
                    } else {
                        $("#lblMessage").html('Erreur : ' + json.reponse);
                    }
                } // success
            }); /// ajax
        } /// else
    }); /// function
}); /// ready
```

```php
<?php

/* UploadImageAjax.php */

// La source
$source = $_FILES['fichierACharger']['tmp_name'];

// Chemin de stockage du fichier
$cible = "../images/" . $_POST['nomDuFichierCible'];

// Moving Uploaded file
$ok = move_uploaded_file($source, $cible);

$code = "-1";
if ($ok) {
    $code = "1";
}

echo json_encode(["reponse" => $code]);
?>
```

La variante !

Le « bouton file » est remplacé par un « button ».

Il faut ajouter un « button » identifié par « btSelection ».
Il faut ajouter du CSS pour masquer le « bouton file » et du JavaScript pour émuler un click sur l'input « file » à partir du « button ».

```
<input type="button" id="btSelection" value="Sélectionnez un fichier" />
```

```
<style>
    #fichierACharger{
        display: none;
    }
    input[type="button"] {
        margin-top: 1em;
        width: 150px;
    }
    input[type="submit"] {
        margin-top: 1em;
        width: 150px;
    }
</style>
```

```
<script>
    $("#btSelection").on("click", function() {
        $("#fichierACharger").click();
    });
</script>
```

1.10.2 - Version utilisant seulement un script JavaScript sans PHP

1.11 - Download de fichier

L'objectif du client est de récupérer sur son poste un fichier situé sur un serveur distant (of course!).

L'intérêt de passer par AJAX ?

1.12 - Requête Cross Domain

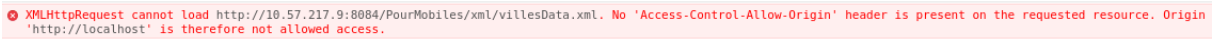
1.12.1 - Principes

Les requêtes AJAX ne peuvent être exécutées que sur le même domaine (ou sous domaine).

Sauf s'il s'agit :

- ✔ d'une requête appelant un script JavaScript avec $.get ou $getScript,
- ✔ d'une requête JSONP avec $.get ou $.getJSON,
- ✔ d'une requête vers un code d'un autre domaine, le code autorisant de Cross Domain,
- ✔ et quelques autres (cf les liens ci-dessous).

Voilà le résultat d'une requête qui n'utilise pas une de ces techniques :



XMLHttpRequest cannot load http://10.57.217.9:8084/PourMobiles/xml/villesData.xml. No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://localhost' is therefore not allowed access.

Excellent :
http://jquery-howto.blogspot.fr/2013/09/jquery-cross-domain-ajax-request.html

Excellent :
http://zinoui.com/blog/cross-domain-ajax-request

En AJAX pur, très peu de jQuery.
http://www.html5rocks.com/en/tutorials/cors/

En français :
https://openclassrooms.com/courses/ajax-et-l-echange-de-donnees-en-javascript/l-xmlhttprequest-cross-domain

1.12.2 - Code PHP simplifié

```php
<?php

/*
 * GetHoraires.php
 */

header("Access-Control-Allow-Methods: POST, GET, OPTIONS");
header("Access-Control-Allow-Headers: *");
header("Access-Control-Allow-Origin: *");

require_once './simplejson.php';
// Recuperation du contenu du fichier sous forme de flux de caracteres
$contenuFichier = file_get_contents("../ressources/json/horaire.json");
// Affichage du contenu du fichier
echo $contenuFichier;
?>
```

1.12.3 - Autorisation côté serveur

1.12.3.1 - Avec du PHP

http://127.0.0.1/jQueryAjaxCours/html/JQAjaxCrossDomain.html

Regardez bien l'IP ! Ce n'est pas du cross-domain puisque la requête est du même domaine.

JQAjaxCrossDomain

Requêtez !

Ce n'est pas une requête cross-domain, ça vient du même domaine

http://10.57.255.168/jQueryAjaxCours/html/JQAjaxCrossDomain.html

Regardez bien l'IP ! C'est du cross-domain parce l'on requête d'un autre domaine autorisé.

JQAjaxCrossDomain

Requêtez !

From CrossDomain.php, donc dans la liste des autorisés !

http://192.168.122.1/jQueryAjaxCours/html/JQAjaxCrossDomain.html

Regardez bien l'IP ! C'est du cross-domain mais l'on requête d'un autre domaine non autorisé.

JQAjaxCrossDomain

Requêtez !

error : 0

Syntaxes PHP

$_SERVER['HTTP_ORIGIN']

Méthode de requête OPTIONS : la méthode OPTIONS représente une demande d'informations sur les options de communication disponibles sur la chaîne de requête / réponse identifiée par l'URI.

$_SERVER['REQUEST_METHOD'] == 'OPTIONS'

URI, URL, URN :

Une URI, "Uniform Resource Identifier", peut être classée comme un localisateur, un nom, ou les deux. Le terme "Uniform Resource Locator" (URL) se réfère au sous-ensemble des URI qui, en plus d'identifier une ressource, fournissent un moyen de localisation de la ressource en décrivant son mécanisme d'accès primaire (par exemple, son « emplacement » réseau). Le terme «Uniform Resource Name" (URN) a été utilisé historiquement pour désigner les deux.

Modification des entêtes de requête :

```
header("Access-Control-Allow-Origin: " . $origin);
```

L'autorisation de certificats (cookies, SSL,)

```
header("Access-Control-Allow-Credentials: true");
```

Les méthodes de requête autorisées

```
header("Access-Control-Allow-Methods: POST, GET, OPTIONS");
```

Les entêtes autorisés

```
header("Access-Control-Allow-Headers: Origin");
```

Les scripts :

HTML

```
<!DOCTYPE html>
<!--
JQAjaxCrossDomain.html
-->
<html>
    <head>
        <title>JQAjaxCrossDomain</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
    </head>

    <body>
        <div>
            <h3>JQAjaxCrossDomain</h3>

            <button id="btRequeter">Requêtez !</button>
            <p id="lblMessage">Résultat</p>
        </div>

        <script src="../jquery/jquery.js"></script>
        <script src="../js/JQAjaxCrossDomain.js"></script>
    </body>
</html>
```

JavaScript

```
/*
 * JQAjaxCrossDomain.js
 */
function init() {
    $("#btRequeter").click(getData);
} /// init

function getData() {
    console.clear();
    // ça déjà c'est KO si la page HTML est requetée avec
http://localhost/...
    let url = "http://127.0.0.1/jQueryAjaxCours/php/CrossDomain.php";
//    let url = "http://10.57.217.9/jQueryAjaxCours/php/CrossDomain.php";
//    let url = "http://10.57.217.9:8084/PourMobiles/CrossDomainServlet";

    /*
     * AVEC $.ajax
     */
    $.ajax({
        xhrFields: {
            // Cookies, SSL, Authentification.
            withCredentials: false
        },
        type: "GET",
        url: url,
        success: function(data) {
            console.log(data);
            $("#lblMessage").html(data);
        },
        error: function(xhr, statut, erreur) {
            console.log(xhr);
            $("#lblMessage").html(statut + " : " + xhr.status);
        },
        crossDomain: true
    });
} /// getData

/*
 *
 */
$(document).ready(init);
```

PHP

```php
<?php

/*
 * CrossDomain.php
 *
 * http://10.57.217.9/jQueryAjaxCours/php/CrossDomain.php
 */

/*
 * Si la requete vient du meme domaine
 */
if (!isset($_SERVER['HTTP_ORIGIN'])) {
    // Ce n'est pas une requete cross-domain
    echo "Ce n'est pas une requête cross-domain, ça vient du même domaine";
    exit;
}
/*
 * Set $wildcard to TRUE if you do not plan to check or limit the domains
 * TRUE : tout le monde peut requeter
 * FALSE : il faut lister les domaines autorises
 */
$wildcard = FALSE;
/*
 * Set $credentials to TRUE if expects credential requests (Cookies, Authentication, SSL certificates)
 */
$credentials = FALSE;
/*
 * La liste des domaines autorises a requeter ici
 */
$allowedOrigins = array('http://10.57.255.168', 'http://localhost', 'http://127.0.0.1');
$allowedOrigins = array('http://localhost', 'http://127.0.0.1');
//$allowedOrigins = array("*");

/*
 * Si ce n'est pas dans la liste ou pas *
 */
if (!in_array($_SERVER['HTTP_ORIGIN'], $allowedOrigins) && !$wildcard) {
    // Origin is not allowed
    echo "Ce domaine n'est pas autorisé";
    exit;
}

// SI caractere generique et pas de certificats
// alors affecter '*' a $origine
// sinon affecter _SERVER['HTTP_ORIGIN'] a $origine
$origin = $wildcard && !$credentials ? '*' : $_SERVER['HTTP_ORIGIN'];

// Liste des domaines autorises
header("Access-Control-Allow-Origin: " . $origin);

// Si $credentiels on autorise les certificats
if ($credentials) {
```

```
    header("Access-Control-Allow-Credentials: true");
}
// Les methodes de requete autorisees
header("Access-Control-Allow-Methods: POST, GET, OPTIONS");
//
header("Access-Control-Allow-Headers: Origin");

// Handling the Preflight (controle en amont)
// Si la methode est OPTIONS ... cf plus haut
if ($_SERVER['REQUEST_METHOD'] == 'OPTIONS') {
    exit;
}

// Response
echo "From CrossDomain.php, donc dans la liste des autorisés !"
?>
```

1.12.3.2 - Avec du code Java

```
/*
 * Servlet LesPaysEnCSV.java
 */
package fr.pb.servicesweb;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;

public class LesPaysEnCSV extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        StringBuilder lsbContenu = new StringBuilder();
        response.setContentType("text/plain;charset=UTF-8");
        response.setHeader("Access-Control-Allow-Origin", "*");
        PrintWriter out = response.getWriter();

        try {
            // Connexion
            String lsURL = "jdbc:mysql://localhost:3306/cours";
            Class.forName("com.mysql.jdbc.Driver");
            Connection lcn = DriverManager.getConnection(lsURL, "root",
"");

            // Les pays
            String lsSQL = "SELECT * FROM pays ORDER BY nom_pays";

            PreparedStatement lpst = lcn.prepareStatement(lsSQL);
            ResultSet lrs = lpst.executeQuery();

            while (lrs.next()) {
                lsbContenu.append(lrs.getString("id_pays"));
                // Separateur de champ
                lsbContenu.append(";");
                lsbContenu.append(lrs.getString("nom_pays"));
                // Separateur d'enregistrement
                lsbContenu.append("\n");
            }

            lrs.close();
            lpst.close();
            lcn.close();
        } catch (ClassNotFoundException | SQLException e) {
            lsbContenu.append(e.getMessage());
        }

        out.print(lsbContenu.toString());
```

```
    } /// doGet

} /// class
```

1.12.4 - Récapitulatif Cross-Domain

jQuery

	PHP	SERVLET
CSV	OK	OK
XML	OK	OK
JSON	KO	OK

JavaScript et XHR et XDR

	PHP	SERVLET
CSV	OK	OK
XML	KO	OK
JSON	OK	OK

CHAPITRE 2 - ANNEXES

2.1 - Les ressources utilisées

2.1.1 - Les fichiers CSV

tintin.txt

1;Tintin
2;Milou
3;Haddock
4;Dupont
5;Dupond
6;Tournesol
7;Castafiore

2.1.2 - Les fichiers XML

2.1.2.1 - Prémisses

XML (eXtensible Markup Language) est un langage à balises de la famille **SGML** (Standard Generalized Markup Language) comme **HTML** (HyperText Markup Language). Alors qu'avec HTML (jusqu'à la version 4 et pour le « noyau » de la version 5) vous utilisez des balises pré-définies par le W3C, avec XML vous créez vous propres balises.

Un document XML est un fichier ASCII composé d'un prologue, est régi – éventuellement, c'est hautement recommandé - par une grammaire de type DTD - Document Type Definition - (un fichier .dtd) ou de type « schema » (un fichier .xsd) , éventuellement de commentaires (même syntaxe qu'en HTML), et d'éléments (une balise ouvrante auto-fermées – pensez à la balise
 de HTML - ou une balise ouvrante liée à une balise fermante).
Une balise ouvrante peut contenir des attributs.
Un élément « parent » - donc avec une balise ouvrante et une balise fermante, pensez à la balise <form> de HTML - contient des éléments enfants.

Taxinomie des fichiers XML

Type data :

un document XML de type data est un document qui correspond à une structure tabulaire ie chaque élément contient un nombre fixe d'attributs. En SQL cela correspond à une table (en 3ème FN). Pensez aux données stockant la liste des 235 pays de la planète Terre.

Type document :

un document XML de type document est un document qui correspond à une structure non tabulaire – plutôt patadoïdale - ie chaque élément contient un nombre variable d'éléments enfants. Pensez au document XML des communes de France où à un code postal correspond plusieurs communes.
En SQL cela correspond à deux tables « reliées » par un relation 1,N.

Type mixte :

le mélange des deux (cf un fichier RSS).

2.1.2.2 - villesData.xml

Document XML référençant les villes de France et d'ailleurs de plus de 100 000 habitants.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- villesData.xml -->
<villes>
  <ville cp="13000" nom_ville="Marseille" id_pays="33" nom_pays="France"/>
  <ville cp="33000" nom_ville="Bordeaux" id_pays="33" nom_pays="France"/>
  <ville cp="59000" nom_ville="Lille" id_pays="33" nom_pays="France"/>
  <ville cp="75011" nom_ville="Paris XI" id_pays="33" nom_pays="France"/>
  <ville cp="75012" nom_ville="Paris XII" id_pays="33" nom_pays="France"/>
  <ville cp="99100" nom_ville="Rome" id_pays="39" nom_pays="Italie"/>
  <ville cp="99101" nom_ville="Milan" id_pays="39" nom_pays="Italie"/>
  <ville cp="99200" nom_ville="Madrid" id_pays="34" nom_pays="Espagne"/>
  <ville cp="99201" nom_ville="Barcelone" id_pays="34" nom_pays="Espagne"/
>
</villes>
```

2.1.2.3 - villesDocument.xml

Document XML référençant les grandes villes de France. Ce document est un « faux » document de type « document » puisqu'il représente une structure tabulaire et est plus prolixe donc d'un poids supérieur au document de type « data » équivalent.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- villesDocument.xml -->
<villes>
  <ville>
    <cp>75000</cp>
    <nom_ville>Paris</nom_ville>
    <id_pays>033</id_pays>
    <nom_pays>France</nom_pays>
  </ville>
  <ville>
    <cp>69000</cp>
    <nom_ville>Lyon</nom_ville>
    <id_pays>033</id_pays>
    <nom_pays>France</nom_pays>
  </ville>
  <ville>
    <cp>14200</cp>
    <nom_ville>Cabourg</nom_ville>
    <id_pays>033</id_pays>
    <nom_pays>France</nom_pays>
  </ville>
  <ville>
    <cp>14000</cp>
    <nom_ville>Caen</nom_ville>
    <id_pays>033</id_pays>
    <nom_pays>France</nom_pays>
  </ville>
</villes>
```

2.1.2.4 - communesDocument.xml

Document XML référençant les communes de France.
Plusieurs communes ont le même code postal. Il est nécessaire de créer autant d'éléments enfants que nécessaire.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- communesDocument.xml -->
<communes>
    <cp cp="75000">
        <commune>Paris</commune>
        <id_pays>033</id_pays>
        <nom_pays>France</nom_pays>
    </cp>
    <cp cp="69000">
        <commune>Lyon</commune>
        <id_pays>033</id_pays>
        <nom_pays>France</nom_pays>
    </cp>
    <cp cp="13000">
        <commune>Marseille</commune>
        <id_pays>033</id_pays>
        <nom_pays>France</nom_pays>
    </cp>
    <cp cp="24200">
        <commune>Sarlat</commune>
        <commune>Vitrac</commune>
        <commune>Carsac</commune>
        <commune>Aillac</commune>
        <id_pays>033</id_pays>
        <nom_pays>France</nom_pays>
    </cp>
</communes>
```

2.1.2.5 - accordeon.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- accordeon.xml -->
<root>
    <soufflet>
        <titre>Cinema</titre>
        <contenu>
            <valeur>Sean Penn</valeur>
            <valeur>Charlotte Gainsbourg</valeur>
            <valeur>Penelope Cruz</valeur>
        </contenu>
    </soufflet>
    <soufflet>
        <titre>Theatre</titre>
        <contenu>
            <valeur>L'avare</valeur>
            <valeur>Andromaque</valeur>
            <valeur>Le cid</valeur>
        </contenu>
    </soufflet>
    <soufflet>
        <titre>Litterature</titre>
        <contenu>
            <valeur>Flaubert</valeur>
            <valeur>Zola</valeur>
            <valeur>Balzac</valeur>
        </contenu>
    </soufflet>
</root>
```

2.1.3 - Les fichiers JSON

Le fichier JSON : ville.json (attention! La clé ne peut être numérique !!! Même entre "").

ville.json

Un objet ...

```
{
    "ville": "Paris",
    "codePostal": "75000",
    "habitants": "2000000"
}
```

villes.json

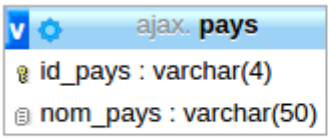
Un tableau d'objets ...

```
[
    {
        "ville": "Paris",
        "codePostal": "75000",
        "habitants": "2 000 000"
    },
    {
        "ville": "Lyon",
        "codePostal": "69000",
        "habitants": "450 000"
    },
    {
        "ville": "Marseille",
        "codePostal": "13000",
        "habitants": "800 000"
    }
]
```

pays.json


```
[
  {
    "id_pays": "033",
    "nom_pays": "France"
  },
  {
    "id_pays": "034",
    "nom_pays": "Espagne"
  },
  {
    "id_pays": "035",
    "nom_pays": "Angleterre"
  },
  {
    "id_pays": "039",
    "nom_pays": "Italie"
  },
  {
    "id_pays": "040",
    "nom_pays": "Roumanie"
  },
  {
    "id_pays": "359",
    "nom_pays": "Bulgarie"
  }
]
```

2.1.4 - La BD ajax




```
DROP DATABASE IF EXISTS ajax;

CREATE DATABASE ajax DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;

USE ajax;


CREATE TABLE ajax.pays (
id_pays VARCHAR(4) NOT NULL ,
nom_pays VARCHAR(50) NOT NULL,
PRIMARY KEY (id_pays)
) ENGINE = MYISAM ;



INSERT INTO ajax.pays (id_pays, nom_pays)
VALUES
('033', 'France'),
('039', 'Italie')
```

2.1.5 - Les scripts PHP

2.1.5.1 - VillesInsert.php

Insertion d'une ville dans la table villes (cp, nom_ville, id_pays).

Le script récupère les attributs de la requête HTTP, puis tente une insertion dans la table. Si tout se passe bien un message personnalisé est renvoyé autrement un message d'erreur MySQL est renvoyé.

Il faudrait ajouter dans les 2 scripts des tests, au moins pour savoir si les zones sont saisies.

```
<?php
    // --- VillesInsert.php

    $cp = filter_input(INPUT_POST, "cp");
    $nomVille = filter_input(INPUT_POST, "nom_ville");
    $idPays = filter_input(INPUT_POST, "id_pays");
    $lsMessage = "";

    try {
        $lcn = new PDO("mysql:host=localhost;dbname=ajax", "root", "");
        $lcn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $lcn->exec("SET NAMES 'UTF8'");

        $lsSQL = "INSERT INTO villes(cp, nom_ville, id_pays)
VALUES(?,?,?)";
        $lcmd = $lcn->prepare($lsSQL);
        $lcmd->execute(array($cp, $nomVille, $idPays));

        $lsMessage = $lcmd->rowcount() . " enregistrement(s) ajouté(s)";

        $lcn = null;
    }
    catch(PDOException $e) {
        $lsMessage = $e->getMessage();
    }
    echo $lsMessage;
?>
```

2.1.5.2 - VillesDelete.php

Suppression d'une ville dans la table villes (cp, nom_ville, id_pays).

Le script récupère les attributs de la requête HTTP, puis tente une suppression dans la table.
Si tout se passe bien un message personnalisé est renvoyé autrement un message d'erreur MySQL est renvoyé.

Il faudrait ajouter dans les 2 scripts des tests, au moins pour savoir si les zones sont saisies.

```
<?php
    // --- VillesDelete.php

    $cp = filter_input(INPUT_POST, "cp");
    $lsMessage = "";

    try {
        $lcn = new PDO("mysql:host=localhost;dbname=ajax", "root", "");
        $lcn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $lcn->exec("SET NAMES 'UTF8'");

        $lsSQL = "DELETE FROM villes WHER cp = ?";
        $lcmd = $lcn->prepare($lsSQL);
        $lcmd->execute(array($cp));

        $lsMessage = $lcmd->rowcount() . " enregistrement(s) supprimé(s)";

        $lcn = null;
    }
    catch(PDOException $e) {
        $lsMessage = $e->getMessage();
    }
    echo $lsMessage;
?>
```

2.1.5.3 - VillesSelect.php

Sélection des cp et nom_ville de la table villes.
Le script renvoie la liste des villes (cp;ville\n) ou un message d'erreur.

```
<?php
    // --- VillesSelect.php
    $lsContenu = "";

    try {
        $lcn = new PDO("mysql:host=localhost;dbname=ajax", "root", "");
        $lcn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $lcn->exec("SET NAMES 'UTF8'");

        $lsSQL = "SELECT cp, nom_ville FROM villes";
        $lrs = $lcn->prepare($lsSQL);
        $lrs->execute();

        foreach($lrs as $enr) {
            $lsContenu .= "$enr[0];$enr[1]\n";
        }
        $lsContenu = substr($lsContenu, 0, -1);

        $lcn = null;
    }
    catch(PDOException $e) {
        $lsContenu = $e->getMessage();
    }

    echo $lsContenu;
?>
```

2.1.5.4 - VillesDUnPays.php

Sélection des cp et nom_ville de la table villes en fonction de l'id_pays.
Le script renvoie la liste des villes (cp;ville\n) ou un message d'erreur.

```php
<?php
    // --- VillesDUnPays.php

    $id = filter_input(INPUT_GET, "id");

    $lsContenu = "";

    try {
        $lcn = new PDO("mysql:host=localhost;dbname=ajax", "root", "");
        $lcn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $lcn->exec("SET NAMES 'UTF8'");

        $lsSQL = "SELECT cp, nom_ville FROM villes WHERE id_pays = ?";
        $lrs = $lcn->prepare($lsSQL);
        $lrs->execute(array($id));

        foreach($lrs as $enr) {
            $lsContenu .= "$enr[0];$enr[1]\n";
        }

        if($lsContenu != "") {
            $lsContenu = substr($lsContenu, 0, -1);
        }

        $lcn = null;
    }
    catch(PDOException $e) {
        $lsContenu = $e->getMessage();
    }

    echo $lsContenu;
?>
```

2.1.5.5 - PaysInsert.php

```
<?php
    // --- paysInsert.php
    $idPays  = $_POST["id_pays"];
    $nomPays = $_POST["nom_pays"];
    $lsMessage = "";

    try {
        $lcn = new PDO("mysql:host=localhost;dbname=cours", "root", "");
        $lcn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $lcn->exec("SET NAMES 'UTF8'");

        $lsSQL = "INSERT INTO pays(id_pays, nom_pays) VALUES(?,?)";
        $lcmd = $lcn->prepare($lsSQL);
        $lcmd->execute(array($idPays, $nomPays));

        $lsMessage = $lcmd->rowcount() . " enregistrement(s) ajouté(s)";

        $lcn = null;
    }
    catch(PDOException $e) {
        $lsMessage = $e->getMessage();
    }
    echo $lsMessage;

?>
```

2.1.5.6 - PaysSelect.php

Sélection des id_pays et nom_pays de la table pays.
Le script renvoie la liste des pays (cp;pays
) ou un message d'erreur.

```
<?php
    // --- PaysSelect.php

    $lsContenu = "";

    try {
        $lcn = new PDO("mysql:host=localhost;dbname=ajax", "root", "");
        $lcn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $lcn->exec("SET NAMES 'UTF8'");

        $lsSQL = "SELECT id_pays, nom_pays FROM pays";
        $lrs = $lcn->prepare($lsSQL);
        $lrs->execute();

        foreach($lrs as $enr) {
            $lsContenu .= "$enr[0];$enr[1]\n";
        }
        $lsContenu = substr($lsContenu, 0, -1);

        $lcn = null;
    }
    catch(PDOException $e) {
        $lsContenu = $e->getMessage();
    }

    echo $lsContenu;
?>
```

2.1.5.7 - PaysSelectEnJSON.php

```php
<?php
// --- PaysSelectEnJSON.php
header("Access-Control-Allow-Origin:*");
//header("Content-Type: text/plain");

// On renvoie une chaine JSON
// Ou un objet JSON

$chaineJSON = "";

try {
    $lcn = new PDO("mysql:host=localhost;dbname=cours", "root", "");
    $lcn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $lcn->exec("SET NAMES 'UTF8'");

    $lsSQL = "SELECT id_pays, nom_pays FROM pays";
    $lrs = $lcn->prepare($lsSQL);
    $lrs->execute();
    $lrs->setFetchMode(PDO::FETCH_ASSOC);

//    // Un tableau de pays
//    // Tableau nomme
//    $chaineJSON = '[';
//    while ($enr = $lrs->fetch()) {
//        $chaineJSON .= '{"id_pays":"' . $enr["id_pays"] .
'","nom_pays":"' . $enr["nom_pays"] . '"},';
//    }
//    // La derniere virgule
//    $chaineJSON = substr($chaineJSON, 0, -1);
//    // Tableau nomme
//    $chaineJSON .= "]";

    $tPays = array();
    while ($enr = $lrs->fetch()) {
        $tPays[] = $enr;
    }

    $chaineJSON = json_encode($tPays);

    $lcn = null;
} catch (PDOException $e) {
    //$chaineJSON = $e->getMessage();
}

// On renvoie une chaine JSON (OK en JS pur, KO en jQuery)
echo $chaineJSON;
// On renvoie un objet JSON ... un vrai bazar cote JS pur
// Mais avec jQuery il faut renvoyer un objet
//echo json_encode($chaineJSON);
?>
```

2.1.5.8 - VillesDUnPaysEnJSON.php

```php
<?php

/**
 * VillesDUnPaysEnJSON.php
 */
/*
 * Connexion a la BD
 */
try {
    $lcnx = new PDO("mysql:host=localhost;port=3306;dbname=ajax;", "root",
"");
    $lcnx->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $lcnx->exec("SET NAMES 'UTF8'");
} catch (PDOException $e) {
    echo "Echec de l'exécution : " . $e->getMessage();
}


/*
 * Recuperation de l'attribut de requete
 */
$cp = filter_input(INPUT_GET, "cp");

try {
    $lsSelect = "SELECT * FROM villes WHERE cp = ?";
    $lrs = $lcnx->prepare($lsSelect);
    $lrs->execute(array($cp));
    $lrs->setFetchMode(PDO::FETCH_NUM);
    $enr = $lrs->fetch();

//    header("Content-Type: application/json; charset=UTF-8");

    $tVille = array();

    if ($enr[1] === null) {
        /*
         * DU JSON
         */
        $tVille["cp"] = "";
        $tVille["nomVille"] = "Introuvable";
        $tVille["idPays"] = "";
    } else {
        $tVille["cp"] = $enr[0];
        $tVille["nomVille"] = $enr[1];
        $tVille["idPays"] = $enr[2];
    }

    $lrs->closeCursor();
} catch (PDOException $e) {
    $tVille["cp"] = "";
    $tVille["nomVille"] = $e->getMessage();
    $tVille["idPays"] = "";
}
```

```php
// Transforme les donnees PHP (un tableau associatif)
// en donnees JSON (un objet JSON)
// C'est ce qui est renvoye a l'appelant en l'occurrence le code AJAX

echo json_encode($tVille);

?>
```

2.1.5.9 - gtCreateInserts.php

Création d'une table avec 100 000 enregistrements.

```
<!-- gtCreateInserts.php -->
<form method="get" action="" >
    Combien d'enregistrements ? <input type="text" name="tb_enr"
value="100000" />
    <input type="submit" />
</form>

<?php
if(isSet($_GET["tb_enr"])) {

    $lsMessage = "OK, c'est fini";

    try {
        $lcn = new PDO("mysql:host=localhost;dbname=ajax", "root", "");
        $lcn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $lcn->exec("SET NAMES 'UTF8'");

        $lcn->exec("DROP TABLE IF EXISTS gt");

        $lsCreate = "CREATE TABLE IF NOT EXISTS gt(id int(5) default NULL,
        nom varchar(50) default NULL) ENGINE=MyISAM CHARACTER SET utf8
COLLATE utf8_general_ci";
        $lcn->exec($lsCreate);

        $lcmd = $lcn->prepare("INSERT INTO gt(id, nom) VALUES(?,?)");

        for($i = 1; $i ≤ $_GET["tb_enr"]; $i++) {
            $lsNom = "Tintin" . (string)$i;
            $lcmd->execute(array($i, $lsNom));
        }
    } // fin du try
    catch(PDOException $e) {
        $lsMessage = $e->getMessage();
    }
    echo $lsMessage;
} // fin du if isSet()
?>
```

2.1.5.10- gtSize.php

Calcul de la taille en octets de la table gt.

```
<?php
    // --- gtSize.php
    // --- Calcul de la taille en octets de la table gt
    // --- taille = nb enr * (longueur moyenne de ID + longueur moyenne de
nom)
    try {
        $lcn = new PDO("mysql:host=localhost;dbname=ajax", "root", "");
        $lcn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $lcn->exec("SET NAMES 'UTF8'");

        // --- gt(id, nom)

        // --- Nombre d'enregistrements (100 000)
        $lsSQL = "SELECT COUNT(*) FROM gt";
        $lrs   = $lcn->query($lsSQL);
        $enr   = $lrs->fetch();
        $count = $enr[0];

        // --- Longueur moyenne de l'ID (environ 5)
        $lsSQL = "SELECT AVG(LENGTH(id)) from gt";
        $lrs   = $lcn->query($lsSQL);
        $enr   = $lrs->fetch();
        $lnID  = $enr[0];

        // --- Longueur moyenne du nom (environ 11)
        $lsSQL = "SELECT AVG(LENGTH(nom)) from gt";
        $lrs   = $lcn->query($lsSQL);
        $enr   = $lrs->fetch();
        $lnNom = $enr[0];

        $taille = $count * ($lnID + $lnNom);
    }
    catch(PDOException $e) {
        $lsContenu = $e->getMessage();
    }
    echo $taille;
?>
```

2.1.5.11 - gtSelect.php

Récupère les enregistrements de la table gt.

```
<?php
    // --- gtSelect.php
    header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
    header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
    header("Cache-Control: no-store, no-cache, must-revalidate");
    header("Cache-Control: post-check=0, pre-check=0", false);
    header("Pragma: no-cache");

    try {
        $lcn = new PDO("mysql:host=localhost;dbname=ajax", "root", "");
        $lcn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $lcn->exec("SET NAMES 'UTF8'");

        // --- gt(id, nom)
        $lsSQL = "SELECT nom FROM gt";

        $lrs = $lcn->query($lsSQL);
        foreach($lrs as $enr) {
            echo $enr[0] . "<br/>";
        }
    } // fin du try
    catch(PDOException $e) {
        $lsContenu = $e->getMessage();
    }
?>
```

2.2 - AJAX, ASP, JSP

2.2.1 - Obtenir des données Texte avec ASP

Objectif

Récupérer le contenu d'un fichier texte (fichier.txt) avec ASP.
Chaque ligne du fichier est concaténée avec un
.

HTML

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>jQuery Get fichier texte avec ASP</title>
    <script src="../../jquery/jquery.js"></script>
    <script>
        // ------------
        function voir() {
            let r =
$.get("http://localhost:8055/asp20emeSiecle/lireFichier0.asp",
function(data) { $("#p_villes").html(data); } );
        }
        $(document).ready(voir);
    </script>
</head>

<body>
    <h3>jQuery Get fichier texte avec ASP</h3>
    <p id="p_villes"></p>
</body>
</html>
```

ASP

```
<% option explicit%>
<html>
<head><title>lireFichier.asp</title></head>
<body>
<%
   ' --- Variables
   Dim fso
   Dim fichier
   Dim FTS
   Dim lsCheminAbsolu
   Dim lsLigne

   Set fso        = Server.createObject("Scripting.FileSystemObject")
   lsCheminAbsolu = Server.MapPath(".")
   Set fichier    = fso.GetFile(lsCheminAbsolu & "\fichier.txt")
   Response.Write(lsCheminAbsolu & "<br/>")

   ' --- Lecture du fichier en question
   Set FTS = fichier.OpenAsTextStream()
   Do While not FTS.AtEndOfStream
      lsLigne = Server.HTMLEncode(FTS.ReadLine)
      Response.Write(lsLigne & "<br/>")
   Loop
   FTS.close
%>
</body>
</html>
```

2.2.2 - Obtenir des données SQL Server via ASPX

HTML

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>jQuery Get SQL avec ASPX</title>

    <script src="jquery/jquery.js"></script>
    <script>
        // ------------
        function voir()
        // ------------
        {
            let r =
$.get("http://localhost:1255/site_bd/villesSelect.aspx", function(data)
{ $("#p_villes").html(data); } );
        }
        $(document).ready(voir);
    </script>

</head>

<body>
    <h3>jQuery Get SQL avec ASPX et SQL Server</h3>
    <p id="p_villes"></p>
</body>
</html>
```

villesSelect.ASPX.VB

```
Imports System.Data.SqlClient

Partial Class villesSelect
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        Dim cn As SqlConnection
        Dim cmd As SqlCommand
        Dim dr As SqlDataReader

        Dim lsCn As String = ""
        Dim lsResultat As String = ""
        Dim i As Integer

        lsCn = "Data Source=ASUS\SQLEXPRESS;Initial
Catalog=bd_cours_2009;Integrated Security=True"
        cn = New SqlConnection(lsCn)
        cn.Open()

        cmd = New SqlCommand("SELECT * FROM villes", cn)
        dr = cmd.ExecuteReader()
        Do While dr.Read()
            For i = 0 To dr.FieldCount - 1
                lsResultat &= dr.Item(i) & "-"
            Next
            lsResultat &= "<br/>"
        Loop
        dr.Close()
        cn.Close()
        Response.Write(lsResultat)
    End Sub
End Class
```

2.2.3 - Obtenir des données Oracle via JSP


```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>jqGetSqlVillesJSP</title>
<script src="../../jquery/jquery.js"></script>
<script>
    // ------------
    function voir() {
        let r =
$.get("http://localhost:8085/WebAppCours/jsp/villesSelect0.jsp",
function(data) { $("#p_villes").html(data); } );
    }
    $(document).ready(voir);
</script>
</head>

<body>
    <h3>jQuery Get SQL Villes JSP/Oracle</h3>
    <p id="p_villes"></p>
</body>
</html>
```

villesSelect0.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8" language="java"
import="java.io.IOException,java.sql.*,packageBD.*" %>
<!DOCTYPE HTML>
<html>
<head>
    <meta charset="UTF-8">
    <title>VILLES</title>
</head>
<body>
<%
try {
    Class.forName("oracle.jdbc.driver.OracleDriver");
    Connection lcConnexion =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","p","b")
;

    Statement lstSql = lcConnexion.createStatement();
    ResultSet lrs    = lstSql.executeQuery("SELECT * FROM villes");
    while(lrs.next()) {
        out.println(lrs.getString(1) + "-" + lrs.getString(2) + "<br/>");
    }
    lrs.close();
    lcConnexion.close();
}
catch(Exception erreur) {
    out.println(erreur.getMessage());
}
%>
</body>
</html>
```

2.2.4 - Obtenir des données MySQL au format CSV via un servlet

Code de la servlet :

```
/*
 * Servlet LesPaysEnCSV.java
 */
package fr.pb.servicesweb;

import java.io.*;
import javax.servlet.ServletException;
import javax.servlet.http.*;
import java.sql.*;

public class LesPaysEnCSV extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        StringBuilder lsbContenu = new StringBuilder();
        response.setContentType("text/plain;charset=UTF-8");
        PrintWriter out = response.getWriter();

        try {
            // Connexion
            Class.forName("com.mysql.jdbc.Driver");
            Connection lcn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/cours", "root",
"");

            // Les pays
            String lsSQL = "SELECT * FROM pays ORDER BY nom_pays";
            PreparedStatement lpst = lcn.prepareStatement(lsSQL);
            ResultSet lrs = lpst.executeQuery();
            while (lrs.next()) {
                lsbContenu.append(lrs.getString("id_pays"));
                // Separateur de champ
                lsbContenu.append(";");
                lsbContenu.append(lrs.getString("nom_pays"));
                // Separateur d'enregistrement
                lsbContenu.append("\n");
            }
            lrs.close();
            lpst.close();
            lcn.close();
        } catch (ClassNotFoundException | SQLException e) {
            lsbContenu.append(e.getMessage());
        }

        out.print(lsbContenu.toString()); // Et pas println
    } /// doGet

} /// class
```

Code AJAX :

```
function getCSV() {

    let jqXHR = $.get
            (
                    "http://localhost:8087/PourSmartphones/LesPaysEnCSV",
                    function(data) {
                        $("#lblMessage").html(data);
                    },
                    "text"
                    );
    jqXHR.error(function() {
        let lsTexte = "Erreur serveur ou réseau ... : " + jqXHR.status +
"-" + jqXHR.statusText;
        $("#lblMessage").html(lsTexte);
    });

} /// getCSV
```

2.2.5 - Obtenir des données MySQL au format XML via un servlet

Code de la servlet :

```
/*
 * Servlet LesPaysEnXML.java
 */
package fr.pb.servicesweb;

import java.io.*;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;
import java.util.*;
import javax.xml.parsers.*;
import javax.xml.transform.*;;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.*;


public class LesPaysEnXML extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        response.setContentType("text/xml;charset=UTF-8");
        PrintWriter out = response.getWriter();

        // On recupere le contenu SQL sous forme de List de Map
        List<Map<String, String>> listeMap = getSQLAsList();
        // On cree un objet DOM
        Document doc = creerDOM("racine");
        // On ajoute dans l'objet DOM
        doc = ajouterNoeudsADOM(doc, listeMap);
        //
        String lsContenuDOC = getDocumentAsString(doc);
        // On envoie
        out.println(lsContenuDOC);
    } /// doGet

    /**
     * Cree un arbre xml avec un element racine
     *
     * @param asRacine
     * @return
     */
    private Document creerDOM(String asRacine) {

        Document doc = null;
```

```
        try {
            // Creation d'un objet DOM
            DocumentBuilderFactory usineDOM =
DocumentBuilderFactory.newInstance();
            DocumentBuilder fabricant = usineDOM.newDocumentBuilder();
            doc = fabricant.newDocument();

            // Creation de l'element racine
            Element racine = doc.createElement(asRacine);

            // Ajoute la racine au document
            doc.appendChild(racine);
        } catch (ParserConfigurationException | DOMException e) {
            System.out.println(e);
        }

        return doc;
    } /// creerDOM

    /**
     *
     * @param doc
     * @param listePays
     * @return
     */
    private Document ajouterNoeudsADOM(Document doc, List<Map<String,
String>> listePays) {

        try {
            // Recuperer la racine
            Element racine = doc.getDocumentElement();
            // Balayer ...
            for (int i = 0; i < listePays.size(); i++) {
                Map<String, String> mapPays = listePays.get(i);
                Element pays = doc.createElement("pays");
                pays.setAttribute("id_pays", mapPays.get("id_pays"));
                pays.setAttribute("nom_pays", mapPays.get("nom_pays"));
                racine.appendChild(pays);
            }
        } catch (DOMException e) {
            System.out.println(e);
        }
        return doc;
    } /// ajouterNoeudsADOM

    /**
     *
     * @param doc
     * @return
     */
    private String getDocumentAsString(Document doc) {
        String lsXML = "";
        try {
            DOMSource source = new DOMSource(doc);

            TransformerFactory transformerFactory =
TransformerFactory.newInstance();
            Transformer transformer = transformerFactory.newTransformer();
```

```
            transformer.setOutputProperty(OutputKeys.METHOD, "xml");
            transformer.setOutputProperty(OutputKeys.INDENT, "yes");
            transformer.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION,
"yes");

            StringWriter writer = new StringWriter();
            StreamResult destination = new StreamResult(writer);

            /// Ecriture
            transformer.transform(source, destination);

            lsXML = writer.toString();
        } catch (IllegalArgumentException | TransformerException e) {
            System.out.println(e);
        }
        return lsXML;
    } /// getDocumentAsString


    /**
     *
     * @return
     */
    private List<Map<String, String>> getSQLAsList() {
        List<Map<String, String>> liste = new ArrayList();
        try {
            // Connexion
            Class.forName("com.mysql.jdbc.Driver");
            Connection lcn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/cours", "root",
"");

            // Les pays
            String lsSQL = "SELECT * FROM pays ORDER BY nom_pays";

            PreparedStatement lpst = lcn.prepareStatement(lsSQL);
            ResultSet lrs = lpst.executeQuery();
            Map<String, String> map;
            while (lrs.next()) {
                map = new HashMap();
                map.put("id_pays", lrs.getString(1));
                map.put("nom_pays", lrs.getString(2));
                liste.add(map);
            }

            lrs.close();
            lpst.close();
            lcn.close();
        } catch (ClassNotFoundException | SQLException e) {

            System.out.println(e);
        }
        return liste;
    } /// getSQLAsList

} /// class
```

Code AJAX :

```
function getXML() {
    console.log("Requête AJAX vers Servlet Java - Du XML");
    let jqXHR = $.get
            (
                    "http://localhost:8087/PourSmartphones/LesPaysEnXML",
                    function(data) {
                        let lesPays = "";
                        $(data).find("pays").each(function() {
                            let idPays = $(this).attr("id_pays");
                            let nomPays = $(this).attr("nom_pays");
                            lesPays += idPays + "-" + nomPays + "<br>";
                        });
                        $("#lblMessage").html(lesPays);
                    },
                    "xml"
                    );
    jqXHR.error(function() {
        console.log("Erreur AJAX-XML");
        let lsTexte = "Erreur serveur ou réseau ... : " + jqXHR.status +
"-" + jqXHR.statusText;
        $("#lblMessage").html(lsTexte);
    });

} /// getXML
```

2.2.6 - Obtenir des données MySQL au format JSON via un servlet

Avec l'API json-simple (cf json_java.odt)..
Le jar :
https://code.google.com/archive/p/json-simple/
You need to put the latest json-simple-1.1.1.jar in your application.

Code du servlet :

```
/*
 * Servlet pour Service Web
 */
package fr.cours.controls;

import org.json.simple.*;
import java.util.*;
import java.sql.*;
import java.io.*;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;

@WebServlet(name = "PaysJSON", urlPatterns = {"/PaysJSON"})
public class PaysJSON extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        response.setContentType("application/json;charset=UTF-8");
        PrintWriter out = response.getWriter();

        String stringJSON = "";
        JSONArray objetArray = new JSONArray();
        Map<String, String> map;

        String lsSQL = "SELECT id_pays, nom_pays FROM pays";

        try {
            Connection lcn =
DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/cinescope2014",
"root", "");
            Statement lstSQL = lcn.createStatement();
            ResultSet lrs = lstSQL.executeQuery(lsSQL);

            while (lrs.next()) {
                map = new LinkedHashMap();
                map.put("id_pays", lrs.getString(1));
                map.put("nom_pays", lrs.getString(2));
                objetArray.add(map);
            }

            lrs.close();
            lstSQL.close();
            lcn.close();
```

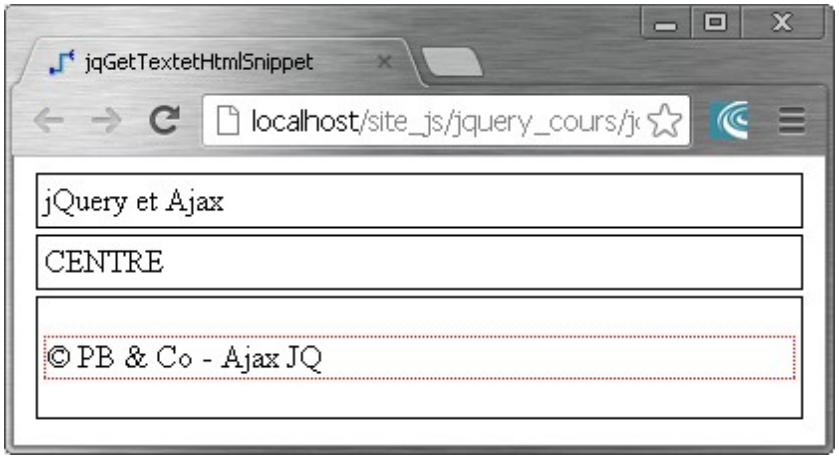
```
        } catch (SQLException e) {
            map = new LinkedHashMap();
            map.put("id_pays", "0");
            map.put("nom_pays", e.getMessage());
            objetArray.add(map);

            System.out.println(e.getMessage());
        }
        stringJSON = objetArray.toJSONString();
        out.print(stringJSON);
    } /// doGet

} /// classe
```

Code AJAX :

```
function getJSON() {
    /*
     * Renvoie un tableau d'objets JSON (id_pays, nom_pays)
     */

    let jqXHR = $.get
            (
                    "http://localhost:8087/PourSmartphones/PayJSON",
                    function(data) {
                        let sPays = "";
                        for (let i=0; i<data.length; i++){
                           sPays+=  data[i].id_pays + "-" +
data[i].nom_pays + "<br>";
                        }
                        $("#lblMessage").html(sPays);
                    },
                    "json"
                    );
    jqXHR.error(function() {
        let lsTexte = "Erreur serveur ou réseau ... : " + jqXHR.status +
"-" + jqXHR.statusText;
        $("#lblMessage").html(lsTexte);
    });

} /// getJSON
```

2.2.7 - Obtenir des données Text et des données HTML

L'objectif est d'intégrer du contenu txt dans une page HTML ainsi que du contenu HTML.



Le fichier texte : **entete.txt**

```
jQuery et AJAX
```

Le fichier html : **pied.html**

```
<p>
    &copy; PB & Co - AJAX JQ
</p>
```

```
<!DOCTYPE html>
<html>
<head>
    <title>JQGetTextetHtmlSnippet</title>
    <meta charset="utf-8" />
    <style type="text/css">
        div{border:1px solid black; margin: 3px; padding: 3px;}
        p{border:1px dotted red;}
    </style>
</head>

<body>
    <div id="entete"></div>
    <div id="centre">
        CENTRE
    </div>
    <div id="pied"></div>

    <script src="../jquery/jquery.js"></script>
    <script>
        // ----------------------
        function getTextSnippet() {
            let jqXHR = $.get
            (
               "../ressources/entete.txt",
               function(data) {
                   $("#entete").html(data);
               },
               "text"
            );
        } /// getTextSnippet

        // ----------------------
        function getHtmlSnippet() {
            let jqXHR = $.get
            (
               "../ressources/pied.html",
               function(data) {
                   $("#pied").html(data);
               },
               "html"
            );
        } /// getHtmlSnippet

        // --------------
        $(document).ready(function() {
            getTextSnippet();
            getHtmlSnippet();
        });
    </script>
</body>
</html>
```

2.3 - Exercice récapitulatif

Du JSON (pour le menu des diaporamas) et du XML (pour chaque diaporama).

MenuDiaporama

- Diaporama Hommes
- Diaporama Femmes
- Diaporama Enfants

DiapoFemmes

Le code HTML

```
<!DOCTYPE html>
<!--
MenuDiaporama.html
-->
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
        <link rel="stylesheet" type="text/css" href="../css/style.css">
        <title>MenuDiaporama.html</title>
    </head>

    <body>
        <div>
            <h3>MenuDiaporama</h3>
            <ul id="MenuDiaporama">
            </ul>
        </div>

        <script src="../jquery/jquery.js"></script>
        <script src="../js/MenuDiaporama.js"></script>
    </body>
</html>
```

Les ressources

MenuDiaporamas.json

```
[
    {
        "id": "DiapoHommes.html", "nom": "Diaporama Hommes"
    },
    {
        "id": "DiapoFemmes.html", "nom": "Diaporama Femmes"
    },
    {
        "id": "DiapoEnfants.html", "nom": "Diaporama Enfants"
    }
]
```

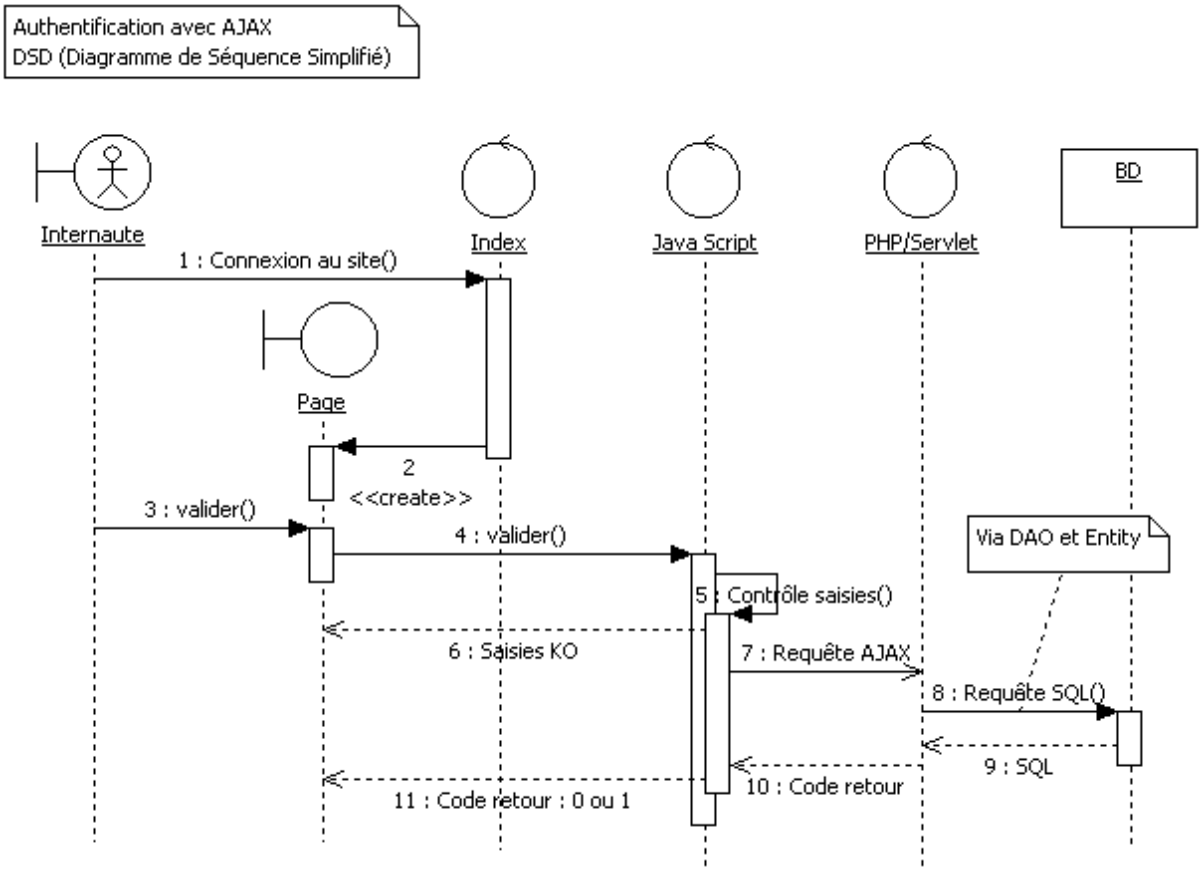
DiaporamaFemmes.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
DiaporamaFemmes
-->

<DiaporamaFemmes>
    <image id="1" source="f1.png" />
    <image id="2" source="f2.png" />
    <image id="3" source="f3.png" />
</DiaporamaFemmes>
```

2.4 - RÉCAPITULATIF

Technique	Technologie serveur	Exemple	Exercice
$.post			
	SQL	Insert Villes	Delete Villes
$.get			
	SQL		
	SQL paramétré		
	XML data		
	XML data paramétré		
	XML document		
	XML document paramétré		
$.ajax			
POST	SQL	Insert Pays	Update Pays
GET	SQL		
	XML		
$.getJSON	JSON	Visualiser des données JSON	
$.getScript	Aucun	Exécuter un script	

2.5 - AJAX et Diagramme de Séquence Détaillé



2.6 - En attente

2.6.1 - AJAX Asynchrone

http://www.webjax.eu/p/225-jQuery-manipulation-dom-javascript-framework-ajax-library-jquery-j-query-Framework-Javascript-Library-Ajax-asyncronous-web2.0-ria-dom-control-enhancer