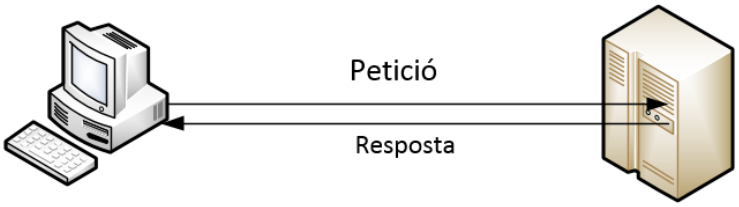
1.1 Mecanismes de comunicació asíncrona

Les pàgines web estan basades en el protocol HTTP, que funciona seguint el model de **"Petició →Resposta per part del servidor → Nou processament al navegador"**.



Amb aquest cicle finalitza tota la relació entre el navegador i el client. Si l'usuari s'ha de comunicar novament amb el servidor ho farà mitjançant l'enviament de dades a través d'un formulari o passant-li informació amb enllaços pel mètode "GET". Això provoca que es torni a crear novament la pàgina generant un nou cicle *"Petició –Resposta".*

Els passos que segueix una aplicació Web són els següents:

1. L'usuari sol·licita una pàgina.
2. El servidor retorna el contingut HTML corresponent a aquesta sol·licitud, que pot ser un simple fitxer HTML o una pàgina generada a partir d'alguna tecnologia de servidor (com ara PHP, ASP.NET, Ruby, JSP).
3. El navegador rep el codi HTML, crea el DOM associat a la pàgina i finalment genera i visualitza el contingut resultant.
4. L'usuari interactua amb HTML, enviant un formulari al servidor o prement un enllaç, i es repeteix el cicle des del pas 1: se sol·licita la pàgina, es torna el seu contingut, es processa i es visualitza.

Aquest procés es coneix com comunicació ***síncrona***, es a dir, l'usuari ha fet una petició i resta bloquejat fins que li arriba tota la informació que ha sol·licitat. HTTP estava pensat per funcionar així quan es va concebre.

El problema és que qualsevol acció en la pàgina que requereixi suport del servidor implica demanar una altra pàgina complerta, descarregar-la i processar-la. Per qualsevol petit canvi que es faci en una pàgina, s'ha de reenviar tot el contingut en lloc de només la informació canviada, la qual cosa és percebuda de manera negativa pels usuaris ("es nota" el temps d'espera i el refresc de la pàgina) i es crea una sensació que la web és poc amigable. A més si el retorn de la pàgina triga més de uns pocs milisegons es perd capacitat de resposta de la interfície ja que, durant el procés el navegador no respon.

Així aquest model web ens provoca una sèrie de problemes:

- Resposta lenta.
- Pèrdua del context durant el refresc de la pàgina.
- Perdem la informació en la pantalla que havíem emplenat prèviament.
- Perdem la posició del scroll de la pantalla.
- No tenim resposta immediata als nostres actes.
- Hem que esperar que arribi la següent pàgina.

La tendència actual en tots els desenvolupaments web, però, és crear aplicacions i pàgines cada vegada més semblants a aplicacions d'escriptori, apropant les pàgines web a la forma de funcionament dels programes que s'executen en l'equip local. Això implica que els molestos i alhora inevitables viatges al servidor no haurien de ser percebuts pels usuaris i que la interfície de la pàgina ha de respondre en tot moment i no bloquejar-se mai. La sensació per als usuaris ha de ser la de que l'aplicació està tot el temps en el seu equip, deixant de banda el servidor, com si fos una aplicació d'escriptori tradicional.

Per aconseguir aquesta sensació es fan servir mecanismes de comunicació **asíncrona** que permeten recarregar en segon pla una part de la pàgina web, deixant la resta operativa desbloquejada.

És per això que van néixer les (**RIA**), Rich Internet Application Technologies:

- Applets, Adobe Flash, DHTML, AJAX

AJAX

Aquesta és la tecnologia que veurem en aquesta Unitat Formativa, ja que està estretament lligada amb JavaScript.

AJAX (Asynchronous JavaScript And XML) que ho podem traduir com JavaScript Asíncron i XML, és una tècnica que ens permet la creació d'aplicacions interactives. En la actualitat és la base d'aplicacions web molt populars, com ara Gmail, Google Maps, Facebook, SkyDrive, etc.

Ajax es fonamenta en l'ús d'un nou objecte ***XMLHttpRequest*** que està implementat per tots els navegadors moderns. Amb aquest objecte podem fer peticions de documents XML al servidor mitjançant el protocol HTTP. La idea original era que, utilitzant aquest objecte es puguin sol·licitar al servidor dades en format XML i que, un cop rebudes pel navegador, es mostrin els resultats modificant dinàmicament els elements de la pàgina mitjançant JavaScript.

Però aquesta idea ha evolucionat i avui en dia el més habitual és que s'utilitzi un altre format per transferir les dades: **JSON**, que veurem més endavant en aquesta mateixa Unitat Formativa. Però el que realment constitueix el cor d'aquesta tècnica és l'objecte ***XMLHttpRequest***.

Tot i que existeixen biblioteques construïdes per facilitar la programació asíncrona, com ara ***jQuery, MooTools, Dojo, Yui…*** En aquesta Unitat Didàctica veurem els fonaments d'Ajax sense entrar en cap de les llibreries esmentades, de manera que puguem entendre la base en la que se sustenten totes elles.

Història d'Ajax

Als anys noranta, la gran majoria de les pàgines web es basaven en pàgines HTML complertes; Com ja hem vist, cada acció de l'usuari requeria que la pàgina es recarregués des del servidor.

El 1999, Microsoft va crear el control ***ActiveX XMLHTTP*** per l'Internet Explorer 5, posteriorment va ser adoptat per Mozilla (actual Firefox), Chrome, Safari, Opera i altres navegadors com a l'objecte ***XMLHttpRequest*** de JavaScript que va passar a ser considerar l'estàndard, deixant curiosament al seu creador original Microsoft fora de l'estàndard, ja que continuava fent servir ***ActiveX XMLHTTP***. Finalment Microsoft ha adoptat el model estàndard ***XMLHttpRequest*** des de l'Internet Explorer 7, tot i que la versió ***ActiveX*** encara és suportada pels seus navegadors.

Les peticions HTTP en segon pla i les tecnologies web asíncrones van romandre força desconegudes fins que van aparèixer aplicacions en línia a gran escala com ***l'Outlook Web Access*** (2000) i ***Oddpost*** (2002), però podem dir que quan realment es van fer populars pel gran públic va ser quan Google va emprar Ajax amb el ***Gmail*** (2004) i ***Google Maps*** (2005).

Jesse James Garrett es va inventar el terme Ajax al febrer de 2005 en un article anomenat "*Ajax: A New Approach to Web Applications*".

El en abril de 2006 el **W3C** va publicar l'esborrany de la primera especificació de l'objecte ***XMLHttpRequest***, en un intent de crear un estàndard web oficial.

1.2 Objectes relacionats. Propietats i mètodes dels objectes

1.2.1 XMLHttpRequest

Com acabem de veure, el primer objecte i més important que hem de conèixer de AJAX és ***XMLHttpRequest*** aquest objecte ens permet comunicar-nos directament amb el servidor fent servir JavaScript sense haver de recarregar la pàgina, executant-se en segon pla i sense que l'usuari ni tant sol se n'adoni.

PRIMERA PRESA DE CONTACTE

```
<html>
<head>
<script>
 function cargarText(url){
  if (window.XMLHttpRequest){// codi per IE7+, Firefox, Chrome, Opera, Safari
    ajx=new XMLHttpRequest();
  }
  else{// codi per IE6, IE5
   ajx =new ActiveXObject("Microsoft.XMLHTTP");
 }
  ajx.open("GET",url,false);
  ajx.send(null);
  document.getElementById('midiv').innerHTML= ajx.responseText;
 }
</script>
</head>
<body>
<div id="midiv">
    <h2>Clica els botons i Ajax canviarà el text</h2>
</div>
<button type="button" onclick="cargarText('text1.txt')">Pulsame</button>
<button type="button" onclick="cargarText('text2.txt')">Pulsame</button>
</body>
</html>
```

***Nota**: Aquest fitxer l'hem de desar al servidor, i també en la mateixa carpeta ha d'haver els fitxers* **text1.txt** *i* **text2.txt** *que els podeu crear amb el bloc de notes i amb qualsevol text.*

El primer que hem de fer és crear una instància de l'objecte **XMLHttpRequest**, en aquest moment ens trobem amb un petit problema. Tots els navegadors moderns (incloses les versions de Internet Explorer 8 i superiors) reconeixen l'objecte *XMLHttpRequest,* però les versions d'IE7 i anteriors en comptes de *XMLHttpRequest* fan servir l'objecte original de Microsoft, el control activeXObject ***Microsoft.XMLHTTP,*** i més encara, si el navegador del client és realment antic, no suportarà cap de les dues formes.

Per tant hem de resoldre el problema mirant si el navegador del client té definits els objectes ***Microsoft.XMLHTTP*** o ***XMLHttpRequest.*** Si un navegador suporta un d'aquest objectes estarà implementat en forma d'una propietat de l'objecte ***window***, per tant les expressions ***if(window.XMLHttpRequest)*** i ***if(window.ActiveXObject)*** prendran un valor diferent de ***null*** i ***d'undefined*** la qual cosa interpretarà JavaScript com un resultat ***true***.

Un altre problema amb el que ens podríem trobar és que hi ha algunes versions de navegadors que tot i implementar l'objecte ***XMLHttpRequest,*** no ho fan com a una propietat de ***window***, sinó com a un objecte nadiu de JavaScript. En aquest cas haurem de fer servir l'operador ***typeof***: ***if(typeof XMLHttpRequest != undefined)*** si existeix serà diferent ***d'undefined*** i retornarà true.

Per tant l'Script del codi anterior el podem reescriure, de la següent manera:

```
<script>
 function cargarText(url){
  if (window.XMLHttpRequest || (typeof XMLHttpRequets)!=undefined){
      ajx =new XMLHttpRequest();
  }
  else if(window. ActiveXObject){
```

```
        ajx =new ActiveXObject("Microsoft.XMLHTTP");
    }
    else{
        alert("El seu navegador no suporta AJAX")
    }
     ajx.open("GET",url,false);
     ajx.send(null);
     document.getElementById('midiv').innerHTML= ajx.responseText;
    }
</script>
```

1.2.2 Mètodes i propietats de XMLHttpRequest

Per comunicar-nos amb el servidor hem d'utilitzar els mètodes ***open()*** i ***send()***.

- **open()** s'encarrega d'obrir un canal de comunicació entre la pàgina i el servidor. Aquesta instrucció només proporciona informació sobre a qui s'ha de sol·licitar la informació, el mètode que farem servir per comunicar-nos, etc. però no fa cap enviament ni recepció d'informació, només prepara el camí.

  La seva sintaxis és ***open(mètode, URL, [asíncron], [usuari], [clau])***

  El primer paràmetre ***mètode*** indica de quina manera volem enviar la petició al servidor, que pot ser per ***GET*** o ***POST***. ***URL*** és l'adreça web a la que cridem. ***Asíncron*** és un valor booleà que indica si la connexió es realitzarà de forma asíncrona (true, per defecte) o no. Els dos últims paràmetres serveixen per especificar un nom d'usuari i una contrasenya d'accés per a recursos protegits per autenticació, tot i que no tenen gaire sentit ja que estan escrits en el codi JavaScript a la vista de tothom, per la qual cosa no s'acostumen a utilitzar.

- **send()**, aquest mètode és el que realment ens comunica amb el servidor, és el que posa en marxa el procés de petició fent servir la connexió o el canal de comunicació que hem obert amb el mètode open() anteriorment. Per tant, perquè aquest mètode funcioni, abans s'ha d'haver executat amb èxit el mètode open().

  La seva sintaxis és ***send(contingut).*** *Contingut* és la informació que volem transferir a l'aplicació del servidor que rebrà la petició. En l'exemple anterior, hem fet la petició directa d'un arxiu de text (`text1.txt o text2.txt`) pel mètode GET i per tant no és necessària cap informació més, per això li hem passat com a paràmetre un valor **null**. També podríem haver deixat el parèntesis en blanc, sense passar cap valor, però això ens pot donar problemes en alguns navegadors.

  Generalment, quan es fa una petició amb el mètode ***GET***, la informació addicional ja va incorporada como una cadena de parelles de nom/valor en la pròpia URL, amb la qual cosa el mètode send() també s'enviarà sense informació. Més endavant veurem un exemple d'això i altres en que enviem la informació pel mètode POST.

- **abort()**, aquest mètode ens permet cancel·lar una petició **send()** enviada prèviament. Imagineu que em llençat una petició i que el servidor està trigant massa a respondre, en aquest cas ens pot interessar avortar l'operació passat un cert període de temps. També veurem un exemple més endavant.

Estat de la petició

Podem saber en tot moment l'estat de la nostra petició consultant el valor emmagatzemat en la propietat de només lectura ***readyState*** de la instància de l'objecte ***XMLHttpRequest***. Els possibles valors són:

- **0** S'ha creat una instància de ***XMLHttpRequest***, però encara no s'ha configurat cap petició -no s'ha executat el mètode open() -.
- **1** La petició ja s'ha configurat, però encara no s'ha enviat –s'ha fet open(), però encara no s'ha fet send()-
- **2** La petició s'ha enviat i encara no hem obtingut resposta -ja s'ha fet send()-
- **3** S'està rebent la informació
- **4** S'ha rebut la informació i ja està disponible per ser processada

Assignació de la funció que s'encarregarà de gestionar la resposta

En el moment que hem realitzat una petició asíncrona, la resta del programa continua funcionant. És a dir, no s'atura en espera de la resposta. Per tant ha d'haver alguna manera en que JavaScript se n'adoni del que està passant amb la nostra petició.

Per controlar-ho disposem de l'esdeveniment ***onreadystatechange*** (tot en minúscules); al que li haurem d'assignar una funció. Aquest esdeveniment és dispara automàticament cada vegada que canvia l'estat de la nostra petició i conseqüentment s'executarà la funció associada que li hem assignat.

Informació sobre el resultat de la petició

Teniu cura amb la interpretació que feu del punt anterior. Que ***readyState*** retorni un 4 no vol dir que la petició hagi acabat amb èxit. Només vol dir que la nostra petició es dona per finalitzada.

Per saber si ha finalitzat amb èxit o ha tingut qualsevol problema haurem de mirar la propietat **status** que ens retorna el codi d'estat HTTP resultat de la petició.

Per exemple 200 (finalitzat correctament), 404 (no trobat), 500 (error intern), etc ... Aquests codis es corresponen amb els estàndard d'HTTP que pots consultar al següent enllaç:

http://www.w3.org/Protocols/HTTP/HTRESP.html

També tenim al nostre abast la propietat **statusText**, es funciona igual que la propietat ***status***, però en comptes de mostrar el codi numèric mostra la cadena de text amb el missatge, per exemple: *OK, Not found...*

Recollir el resultat de la petició

Finalment, si tot ha anat bé (readyState=4 i status=200) la informació resultant de la nostra petició la podem trobar en dues propietats de la nostra instància XMLHttpRequest:

responseXML: conté un arbre DOM en format XML retornat per la petició. Aquest DOM és igual al que hem après a utilitzar per a una pàgina web, només que actua sobre un document XML genèric. Aquesta propietat només és útil si el que ens retorna el servidor és un document XML. Actualment s'ha abandonat gairebé per complet a favor del format JSON, que després veurem.

responseText: és la resposta a la nostra petició en forma de text pur. És la que farem servir habitualment per obtenir el contingut del recurs demanat, ja que **JSON** es retorna com si fos text pur.

RECAPITULACIÓ

Així doncs, un cicle complert de treball amb Ajax seria el següent:

1. Creem una instancia de l'objecte ***HMLHttpRequest*** o de ***Microsoft.XMLHTTP*** segons el navegador que tinguem.
2. Executem el mètode open de la instància que acabem de crear. Passant-li els paràmetres necessaris.
3. Li assignem a l'esdeveniment ***onreadystatechange*** la funció que s'encarregarà de gestionar tota la petició.

4. Llancem la petició amb **send()**
5. La funció que controla qualsevol canvi de l'estat de la nostra petició, s'executarà repetides vegades a cada canvi de l'estat, però a nosaltres només ens interessa quan entra en estat 4.

   Si a més d'entrar en estat 4 el **status** és 200, això vol dir que tot ha anat bé i processarem el resultat, que el tindrem en la propietat ***responseXML*** o en ***responseText***.

   Si l'status és diferent de 200 mostrarem un missatge d'error.

L'exemple anterior era molt bàsic i no incloïa tots els passos, ara veurem un exemple una mica més complert, es tracta d'emplenar una llista desplegable amb opcions diferents segons l'opció triada en una altra llista:

```
<!DOCTYPE html>
<html>
<head>
<title>Exemple llistes dinàmiques amb AJAX</title>
<meta charset="UTF-8">
<script type="text/javascript">

var ajx //objecte per realitzar peticions HTTP(instància de XMLHttpRequest)
var resposta //Contindrà l'XML obtingut per als elements de la llista

//S'encarrega de carregar els ítems de la llista secundària
function carregaElements(tipus)
{
 url="damesecundario.asp?tipus=" + tipus
 //Suportat por Firefox, Opera, Chrome, Safari i IE nous
 if (window.XMLHttpRequest)
 {
   ajx = new XMLHttpRequest();
 }
   else //Per Internet Explorer antic en forma de control Active X
 {
   ajx = new ActiveXObject("Microsoft.XMLHTTP");
 }
 //Ja hauria de tenir una referència a l'objecte
 if (ajx != null)
 {
    ajx.onreadystatechange = finCarga;
    ajx.open("GET", url, true) //El true del final és perquè ho sol·liciti de
                               //forma asíncrona
   ajx.send(null); //en ser una petició GET no hi ha res en el cos de la petició,
                   //d'aquí el null
 }
}

//Funció que s'executa quan canvia l'estat de la càrrega de l'objecte ajx
function finCarga()
{
 if (ajx.readyState == 4) // 4: completat, 3: en curs, 1: carregat, 0: no iniciat
 {
   if (ajx.status == 200) //200: OK (el codi HTTP, podria haver estat 404 (no
                          //trobat), 500 (error), etc ...)
    {
     resposta = ajx.responseXML.getElementsByTagName("item");
      //Agafa només els elements "item"
      procesaElements();
    }
    else // error
    {
     alert("No s'ha pogut recuperar la información de la llista secundària: " +
                                                              ajx.statusText);
    }
 }
}
```

```
function procesaElements()
{
   listaSec = document.getElementById("menu2");
   if (listaSec != null)
   {
     listaSec.options.length = 0;     //Buida la llista d'opcions
     //Carreguem les noves opcions
     for(i=0; i<resposta.length; i++)
     {
            var opc = document.createElement("OPTION");
            opc.text = resposta[i].childNodes[0].nodeValue;
            opc.value = resposta[i].childNodes[0].nodeValue;
            listaSec.options.add(opc);
     }
   }
}

</script>
</head>
<body>
<h3>Exemple per auto-completar din&agrave;micament des del servidor amb AJAX</h3>
<p>Tria una província de la primera llista i s'omplirà la segona
<br>din&aacute;micament amb AJAX des del servidor:</p>
<form name="f1">
    <p><select name="menu1" id="menu1" size="1"
             onchange="carregaElements(this.value);">
    <option value=""></option>
    <option value="Barcelona">Barcelona</option>
    <option value="Girona">Girona</option>
    <option value="Lleida">Lleida</option>
    <option value="Tarragona">Tarragona</option>
    </select><select name="menu2" id="menu2" size="1">
    <option value="">[Sin elementos]</option>
    </select></p>
</form>
<p> </p>
<A href="damesecundario.asp?tipus=Tarragona">Crida tradicional a la provincia
   Tarragona</A>
</body>
</html>
```

Pàgina d'Asp Clàssic que gestionarà la petició, s'ha de dir ***damesecundario.asp*** i s'ha de situar a la mateixa carpeta del servidor on estava la pàgina HTML anterior.

```
<%
   Dim items
   Select Case Request("tipus")
     Case "Barcelona"
            items = Array("Barcelona","Mataró","Badalona","Manresa")
     Case "Girona"
            items = Array("Figueres","Roses","Olot","Banyoles","Puigcerdà")
     Case "Lleida"
            items = Array("Lleida","Tarrega","Cervera","Mollerusa","Balaguer")
     Case "Tarragona"
            items = Array("Tarragona","Reus","Valls","Tortosa","Motblanc","Salou")
   End Select

   '--- Es retorna com XML
   Response.ContentType = "text/xml"
   '--- Evitem que es desi a la memòria cau
   Response.AddHeader "cache-control", "no-cache"
   Response.Expires = -1
```

```
'--- Retornem el XML
   Response.Write resultat(items)

'--- Aquesta funció s'encarrega de crear el XML apropiat
Function resultat(arrItems)
   Dim sRes
   sRes = "<?xml version=""1.0"" encoding=""ISO-8859-1"" ?>" & vbCrLf & "<items>"

   If IsArray(arrItems) Then
      '--- Se recorre la matriz para construir el XML
      For Each item In arrItems
            sRes = sRes + "<item>" + item + "</item>"
      Next
   Else
      sRes = sRes + "<item>[No hi han dades]</item>"
   End If
   sRes = sRes + "</items>"
   resultat = sRes
End Function
%>
```

A continuació veurem un altre exemple, en el que a mida que l'usuari va entrant en una caixa de text el contingut d'un nom, dinàmicament li anirem fent us suggeriment d'un possible nom mitjançant Ajax.

```
<!DOCTYPE html>

<html>
<head>
<script>
    var xmlhttp
    function recomanar(nom){
        if (nom.length==0){
            document.getElementById("resposta").innerHTML="";
            return;
        }
        xmlhttp=creaInstancia();
        if (xmlhttp==null){
            alert ("El teu navegador no suporta AJAX!");
            return;
        }
        var url="recomanacions.asp";
        url=url+"?q="+nom;
        url=url+"&sid="+Math.random();
        xmlhttp.onreadystatechange=canviEstat;
        xmlhttp.open("GET",url,true);
        xmlhttp.send(null);
    }
    function canviEstat(){
        if (xmlhttp.readyState==4){
            document.getElementById("resposta").innerHTML=xmlhttp.responseText;
        }
    }
    function creaInstancia(){
        if (window.XMLHttpRequest){
        //  IE7+, Firefox, Chrome, Opera, Safari
        return new XMLHttpRequest();
        }
       if (window.ActiveXObject){
        // IE6, IE5
        return new ActiveXObject("Microsoft.XMLHTTP");
        }
      return null;
    }
</script>
</head>
<body>
<form>
```

```
Nom: <input type="text" id="txt1"
onkeyup="recomanar(this.value)" />
</form>
<p>Recomanacions: <span id="resposta"></span></p>
</body>
</html>
```

El codi anterior l'haurem de desar en la carpeta específica del nostre servidor, en el nostre cas com que fem servir IIS serà qualsevol carpeta dins de WWWROOT, a continuació haurem de crear la pàgina d'Asp que s'encarregarà de donar resposta a les peticions AJAX, com podeu veure al codi anterior aquesta s'ha d'anomenar ***recomanacions.asp*** i la desarem a la mateixa carpeta que la pàgina anterior, seria la següent:

```
<%
'-------------------------------------------------------------------------
'RECOMANACIONS.ASP
'-------------------------------------------------------------------------
response.expires=-1
a=array("Anna","Bea","Charo","Dioni","Ernest","Eva","Flor","Flor","Gerarard", _
   "Hipolit","Ines","Joan","Kely","Lara","Leo","LLuis","Miquel","Nina","Olga", _
   "Puri","Pere","Rosa","Ramon","Sonia","Irene","Toni","Tere","Ursula","Victor", _
   "Yaiza","Santiago","Jose","Narcís","Xavier","Marisa","Mario","Jaume","Manel", _
    "Andrea","Hortensia","Azucena","Rocio","Macarena","Nicolas","Francisca" )
q=ucase(request.querystring("q"))
'-- Si hem rebut alguna cosa
if len(q)>0 then
    resposta=""
    for i=0 to ubound(a)
        if q=ucase(mid(a(i),1,len(q))) then
            if resposta="" then
                resposta=a(i)
            else
                resposta=resposta & " , " & a(i)
            end if
        end if
    next
end if

if resposta="" then
    response.write("no tinc dades")
else
    response.write(resposta)
end if
%>
```

Fins ara hem vist que la resposta a la nostra petició asíncrona la hem de recollir de la instància de l'objecte ***XMLHttpRequest***, mitjançant la propietat ***responseText*** o bé amb ***responseXML,*** amb el primer accedim a la resposta en forma de text pla, sen se cap tipus de format, mentre que en el segon s'assumeix que el resultat que reben és un document XML i per tant té una estructura interna a la que podem accedir. Ho podeu veure al següent exemple:

```
<!DOCTYPE html>
<html>
<head>
<script>
function carregar(url)
{
var objecteRQ;
if (window.XMLHttpRequest)
  {//--- codi per IE7+, Firefox, Chrome, Opera, Safari
  objecteRQ=new XMLHttpRequest();
  }
else
  {//--- codi per IE6, IE5
  objecteRQ=new ActiveXObject("Microsoft.XMLHTTP");
```

```
    }

  objecteRQ.onreadystatechange=function(){
    if (objecteRQ.readyState==4){
      if(objecteRQ.status==200){
      document.getElementById('A1').innerHTML=objecteRQ.status;
      document.getElementById('A2').innerHTML=objecteRQ.statusText;
      //--- Tractament amb responseText
      document.getElementById('A3').innerHTML=objecteRQ.responseText;
      //--- Tractament amb responseXML
      //--- responseXML retorna el XML en forma d'estructura DOM
      var x,n,i,txt
      x=objecteRQ.responseXML.documentElement.getElementsByTagName("correu")
      txt="<table border='1'><tr><th>Per</th><th>de</th><th>Missatge</th></tr>"
      for (i=0;i<x.length;i++){
        txt=txt + "<tr>"
      //--- columna 1
        n=x[i].getElementsByTagName("per")
        {
          try
            {
              txt=txt + "<td>" + n[0].firstChild.nodeValue + "</td>";
            }
          catch (er)
            {
            txt=txt + "<td> </td>";
            }
          }
        //--- columna2
        n=x[i].getElementsByTagName("de")
        {
          try
            {
              txt=txt + "<td>" + n[0].firstChild.nodeValue + "</td>";
            }
          catch (er)
            {
            txt=txt + "<td> </td>";
            }
          }
      //--columna 3
        n=x[i].getElementsByTagName("cos")
        {
          try
            {
              txt=txt + "<td>" + n[0].firstChild.nodeValue + "</td>";
            }
          catch (er)
            {
            txt=txt + "<td> </td>";
            }
          }
        txt=txt + "</tr>";
      }
      txt=txt + "</table>";
      document.getElementById('A4').innerHTML=txt
        }
      else{
        alert("No s'ha pogut recuperar la información de la llista
                                                secundària\nError: "
              + objecteRQ.status+" "+ objecteRQ.statusText)
      }
    }
  }
  objecteRQ.open("GET",url,true);
  objecteRQ.send(null);
  }
```

```
</script>
</head>
<body>
    <h2>Recuperar dades de l'arxiu XML</h2>
    <p><b>Status: </b><span id="A1"></span></p>
    <p><b>Status text: </b><span id="A2"></span></p>
    <p><b>Response: </b><span id="A3"></span></p>
    <div style="width:auto;height: 200px;border: solid thin 1px;
                          background-color: #ff0" id="A4"> </div>
    <button onclick="carregar('dades.xml')">Obtenir dades XML</button>
</body>
</html>
```

Aquesta pàgina porta associat una arxiu ***dades.xml*** que és el que sol·licitem quan fem la petició des de AJAX, el fitxer és el següent:

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

<dades>
    <correu>
   <de>Manola</de>
   <per>Manolin</per>
   <assumpte>Reunió</assumpte>
   <cos>Recorda que tenim reunió el dimarts</cos>
    </correu>
    <correu>
   <de>Baldomero</de>
   <per>Sra. Robinson</per>
   <assumpte>Deute</assumpte>
   <cos>Estimada senyora, si no paga el seu deute serà desnonada</cos>
     </correu>
</dades>
```

1.2.3 Atacant les bases de dades

El procés sempre és el mateix, capturem algun esdeveniment en la nostra pagina web i associat a ell creen una petició, en segon pla que llancem al servidor de manera asíncrona i restem a l'espera de resposta controlant cada vegada que es dispari l'esdeveniment ***onreadystatechange*** i vigilant que a més la propietat ***readyState*** ha canviat al valor **4**. En aquest moment farem el tractament de la informació que rebem associada en la propietat ***responseText*** o en ***responseXML***. Per tant pel que respecta estrictament a la nostra pàgina i a la part d'Ajax la feina ja està acabada, però també es clar que haurem de crear en el servidor l'aplicació que proporcioni la informació que estem demanant, de fet la tècnica que utilitzem és totalment indiferent per la nostra pàgina, ja pot ser PHP, ASP, ASP.NET, Java, Phyton... i també és indiferent si generem les dades sobre la marxa o les obtenim d'una Base de Dades com a l'exemple següent:

```
<!DOCTYPE html>
<html>
<head>
<script>
    var instanciaRQ
    function mostraCustomer(str){
        instanciaRQ=dameInstancia();
        if (instanciaRQ==null){
         alert ("El teu navegador no suporta AJAX!");
         return;
        }
        var url="customer.asp";
        url=url+"?q="+str;
        url=url+"&sid="+Math.random(); //Per evitar la memòria cau
        instanciaRQ.onreadystatechange=haCanviat;
        instanciaRQ.open("GET",url,true);
        instanciaRQ.send(null);
    }
    function haCanviat(){
```

```
        if (instanciaRQ.readyState==4 && instanciaRQ. status==200){
            document.getElementById("elDiv").innerHTML=instanciaRQ.
            responseText;
        }
    }
    function dameInstancia(){
        if (window.XMLHttpRequest){
        // codi IE7+, Firefox, Chrome, Opera, Safari
        return new XMLHttpRequest();
        }
        if (window.ActiveXObject){
        // codi IE6, IE5
            return new ActiveXObject("Microsoft.XMLHTTP");
        }
        return null;
    }
</script>
</head>
<body>
<form>
Seleciona client:
<select name="customers" onchange="mostraCustomer(this.value)">
<option value="ANTON">Antonio Moreno Taquería</option>
<option value="CACTU">Cactus Comidas para llevar</option>
<option value="COMMI">Comércio Mineiro</option>
<option value="FISSA">FISSA Fabrica Inter. Salchichas S.A.</option>
<option value="GALED">Galería del gastrónomo</option>
</select>
</form>
<div id="elDiv"><b>Aquí sortira la informació del client.</b></div>
</body>
</html>
```

El programa que gestiona la BD s'anomena ***customer.asp*** i extraurem les dades de la taula ***customer*** de la base de dades d'exemple d'accés ***Northwind.mdb*** que us he deixat al Moodle del curs.

```
<%
response.expires=-1
sql="SELECT * FROM CUSTOMERS WHERE CUSTOMERID="
sql=sql & "'" & request.querystring("q") & "'"
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open(Server.Mappath("/db/northwind.mdb"))
set rs=Server.CreateObject("ADODB.recordset")
rs.Open sql,conn
response.write("<table>")
do until rs.EOF
   for each x in rs.Fields
      response.write("<tr><td><b>" & x.name & "</b></td>")
      response.write("<td>" & x.value & "</td></tr>")
   next
   rs.MoveNext
loop
response.write("</table>")
%>
```

1.2.4 Enviar la petició pel mètode POST

El més habitual en les peticions asíncrones en AJAX és que la petició es faci sempre pel mètode GET, però ja hem vist que també podem fer servir el mètode POST, tot seguit veurem un exemple en el que enviem a una pàgina ASP el contingut de tots els camps d'un formulari, pel mètode POST. La pàgina ASP només retornarà les dades que ha rebut i informarà si tot ha anat bé o si ens hem deixat en blanc qualsevol camp informarà d'aquest fet

Una altra propietat de l'objecte ***XMLHttpRequest*** a la que podem accedir és ***getAllResponseHeaders*** que ens proporciona informació addicional sobre la resposta que estem rebent. A l'exemple també mostrarem el contingut d'aquesta propietat després d'haver executat la petició.

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <script>
        var instanciaRQ;
        function validar(){
            if(window.ActiveXObject){
                instanciaRQ=new ActiveXObject("Microsoft.XMLHttp");
            }
            else if((window.XMLHttpRequest) || (typeof
  XMLHttpRequest)!=undefined){
                instanciaRQ=new XMLHttpRequest();
            }
            else{
                alert("El seu navegador no suporta AJAX");
                return;
            }
            enviaPeticio();
        }
        function enviaPeticio(){
            instanciaRQ.open("POST",document.forms[0].action,true);
            instanciaRQ.onreadystatechange=procesaResultat;
            instanciaRQ.setRequestHeader("Content-Type", "application/x-www-form-
  urlencoded");
            var dades=obtenerDades();
            //s'envíen les dades en el cos de la petició
            instanciaRQ.send(dades);
        }
        function obtenerDades(){
            var controls=document.forms[0].elements;
            var dades=new Array();
            var cad="";
            for(var i=0;i<controls.length;i++){
                cad=encodeURIComponent(controls[i].name)+"=";
                cad+=encodeURIComponent(controls[i].value);
                dades.push(cad);
            }
            cad=dades.join("&");
            return cad;
        }
        function procesaResultat(){
            if(instanciaRQ.readyState==4){
                if(instanciaRQ.status==200){

  document.getElementById("resultat").innerHTML=instanciaRQ.responseText;
                    var capceleres="<h2>Capçalera de la resposta</h2>";
                    capceleres+=instanciaRQ.getAllResponseHeaders();
                    document.getElementById("capceleres").innerHTML=capceleres;
                }
                else
                    alert(instanciaRQ.statusText);
            }
        }
    </script>
  </head>
  <body>
          <h1>Formulari de sol·licitut de dades</h1>
          <br/>
          <br/>
          <form action="validador.asp">
          Nom: <input type="text" name="nom"/> <br />
          Cognom: <input type="text" name="cognom"/><br />
```

```
            Email: <input type="text" name="mail"/><br />
            Ciutat: <input type="text" name="ciutat"/><br /><br />
                        <input type="button" name="btnvalidar" onclick="validar();"
    value="Procesar"/>
            </form>
            <br/>
            <br/>
            <div id="resultat">
            </div>
             <div id="capceleres">
            </div>
   </body></html>
```

Tot seguit tenim la part corresponent al codi ASP, es tracta de l'arxiu ***validador.asp***

```
<%
    response.expires=-1
    ok=true
    n=request("nom")
    c=request("cognom")
    e=request("mail")
    ci=request("ciutat")
    Response.write("He rebut: <b>" & request.form &"</b><br />")
    if len(n)=0 then
       response.write("El nom no es pot deixar en blanc <br />")
       ok=false
    end if
    if len(c)=0 then
       response.write("El cognom no es pot deixar en blanc <br />")
       ok=false
    end if
    if len(e)=0 then
       response.write("El email no es pot deixar en blanc <br />")
       ok=false
    end if
    if len(ci)=0 then
       response.write("La ciutat no es pot deixar en blanc <br />")
       ok=false
    end if
    if ok then response.write("DADES CORRECTES")
%>
```

Proveu-lo i observeu la part on es mostren les capçaleres, tot seguit elimineu la primera línia del codi ASP `(response.expires=-1)` i observeu com canvia el resultat:

Capçalera de la resposta Sense response.expires=-1

Date: Tue, 07 May 2013 21:41:48 GMT Cache-Control: private Server: Microsoft-IIS/7.5 X-Powered-By: ASP.NET Content-Length: 233 Content-Type: text/html

Capçalera de la resposta Amb response.expires=-1

Date: Tue, 07 May 2013 21:42:16 GMT Cache-Control: private Server: Microsoft-IIS/7.5 Content-Type: text/html X-Powered-By: ASP.NET Content-Length: 233 Expires: Tue, 07 May 2013 21:41:16 GMT

1.2.5 Errors en les peticions AJAX

No podem assumir que les peticions que fem al servidor funcionin sempre. Hi pot haver un error en el codi del servidor, és possible que hagin canviat la URL i que no existeixi la pàgina que cridem (si és una pàgina externa al nostre lloc que no controlem) , que hi hagi errors de permisos, etc ... El que passi al servidor està fora del nostre control (des de JavaScript).

La forma de controlar aquestes situacions és, com ja hem vist, a través del codi d'estat que retorna el servidor.

Però de vegades ens podem trobar amb una situació molt perillosa, que no podem controlar mirant el codi d'estat: podria ser que la petició asíncrona al servidor no torni o no ho faci en un temps raonable, és a dir que es produeixi el que s'anomena un ***timeout***. En aquest cas, no podem comptar amb la notificació de final de càrrega ja que, al no tornar la petició no saltarà l'esdeveniment ***onreadystatechange***.

El que es fa en aquests casos és establir un temporitzador amb el temps màxim que desitgem esperar, després de transcorregut aquets temps anul·larem la petició directament, sense esperar més a que arribi la resposta. Ho podem veure en aquest exemple:

```
<!DOCTYPE html>

<html>
  <head>
    <title></title>
    <script>
        var instanciaRQ, tmrAnular
        function validar(){
            if(window.ActiveXObject){
                instanciaRQ=new ActiveXObject("Microsoft.XMLHttp");
            }
            else if((window.XMLHttpRequest) || (typeof
   XMLHttpRequest)!=undefined){
                instanciaRQ=new XMLHttpRequest();
            }
            else{
                alert("El seu navegador no suporta AJAX");
                return;
            }

            instanciaRQ.open("GET","validador.asp",true);
            tmrAnular = setTimeout("AnularPeticio()", 10000); //10 segons
            instanciaRQ.send(null)
            instanciaRQ.onreadystatechange=procesaResultat;
        }
        function AnularPeticio() {
          alert("abortat")
          instanciaRQ.abort();
        }
       function procesaResultat(){
            if(instanciaRQ.readyState==4){
                clearTimeout(tmrAnular);
                if(instanciaRQ.status==200){

   document.getElementById("resultat").innerHTML=instanciaRQ.responseText;
                }
                else
                    alert("S'ha produit un error"+instanciaRQ.statusText);
            }
        }
    </script>
  </head>
  <body>
          <h1>Formulari de sol·licitut de dades</h1>
          <br/>
          <br/>
          <form action="validador.asp">
          Nom: <input type="text" name="nom"/> <br />
          Cognom: <input type="text" name="cognom"/><br />
          Email: <input type="text" name="mail"/><br />
          Ciutat: <input type="text" name="ciutat"/><br /><br />
                      <input type="button" name="btnvalidar" onclick="validar();"
   value="Procesar"/>
          </form>
          <br/>
```

```
            <br/>
            <div id="resultat">
            </div>
             <div id="capceleres">
            </div>
   </body>
</html>
```

La pàgina ASP ***validador.asp*** està formada per un bucle infinit, d'aquesta manera aconseguim que la petició no torni mai i que s'avorti el procés al cap de 10 segons. També podeu provar d'executar la pàgina però comentant la part del bucle, veureu que en aquest cas s'executa correctament sense cap problema.

```
<%  '--- validador.asp
    response.expires=-1
    do while true
    loop
    Response.write("Hola  he rebut la petició")
%>
```