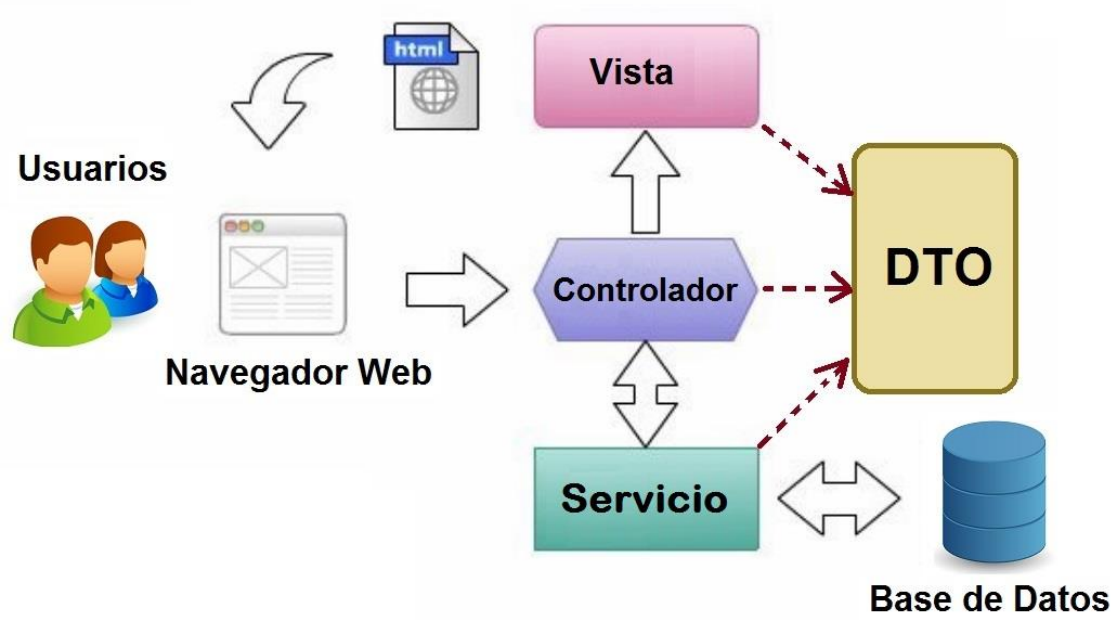
UNIVERSIDAD CIENTÍFICA DEL SUR

TALLER DE PROGRAMACIÓN II

Eric Gustavo Coronel Castillo
gcoronelc.blogspot.com
egcc.ucs@gmail.com

Modelo MVC para la Web



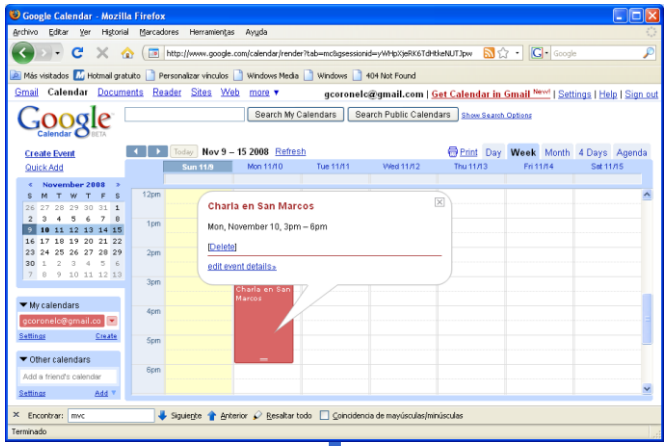
AJAX

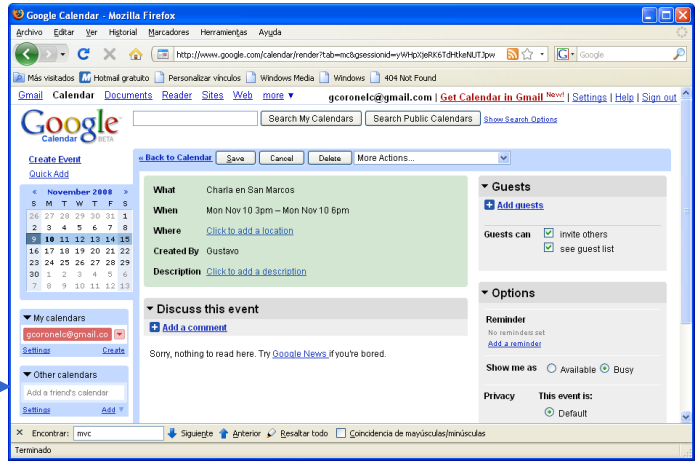
Índice

- Objetivo
- Definición
- Conversación Tradicional Cliente-Servidor
- Interacción AJAX Cliente-Servidor
- El Objeto XMLHTTPRequest
- Implementación
- AJAX con JQuery

Objetivo

- Desarrollar aplicaciones Web que utilicen AJAX para mejor su rendimiento y la experiencia de usuario (Usabilidad).





Definición

- AJAX, acrónimo de **A**synchronous **J**avaScript **A**nd **X**ML (**JavaScript** y **XML** asíncronos, donde XML es un acrónimo de e**X**tensible **M**arkup **L**anguage).

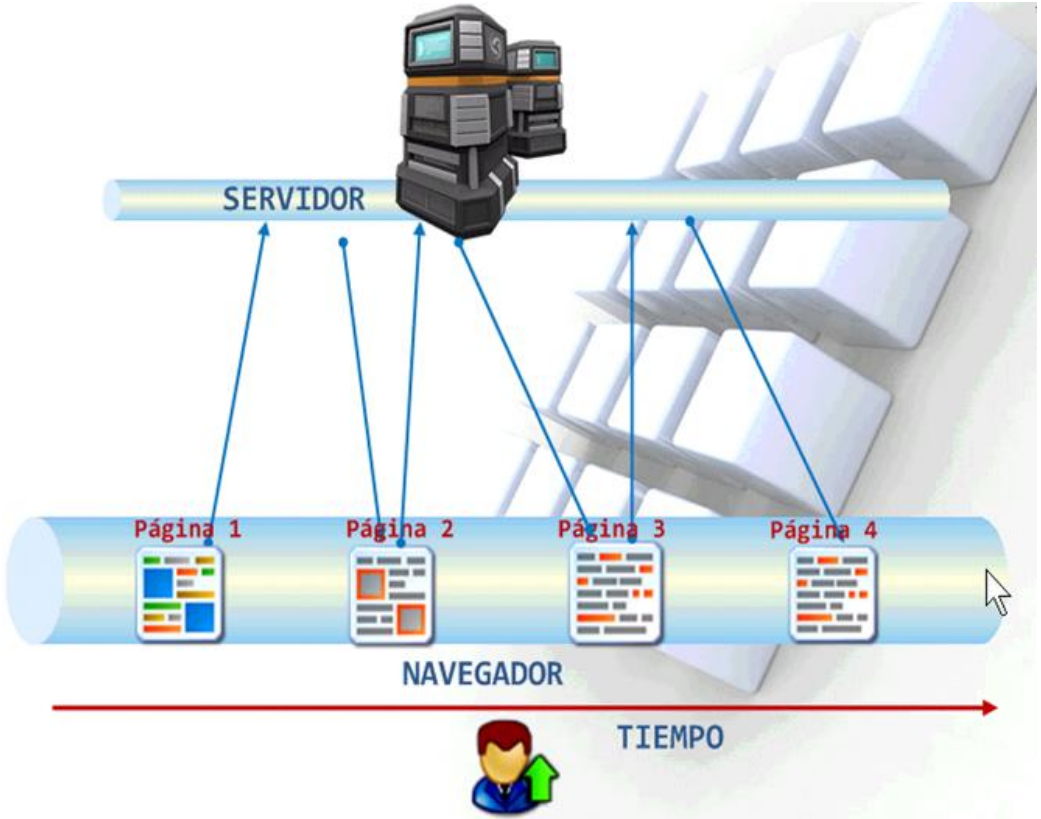
- Es una técnica de desarrollo web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente, es decir, en el navegador del usuario, y mantiene comunicación asíncrona con el servidor en segundo plano.

- De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma.

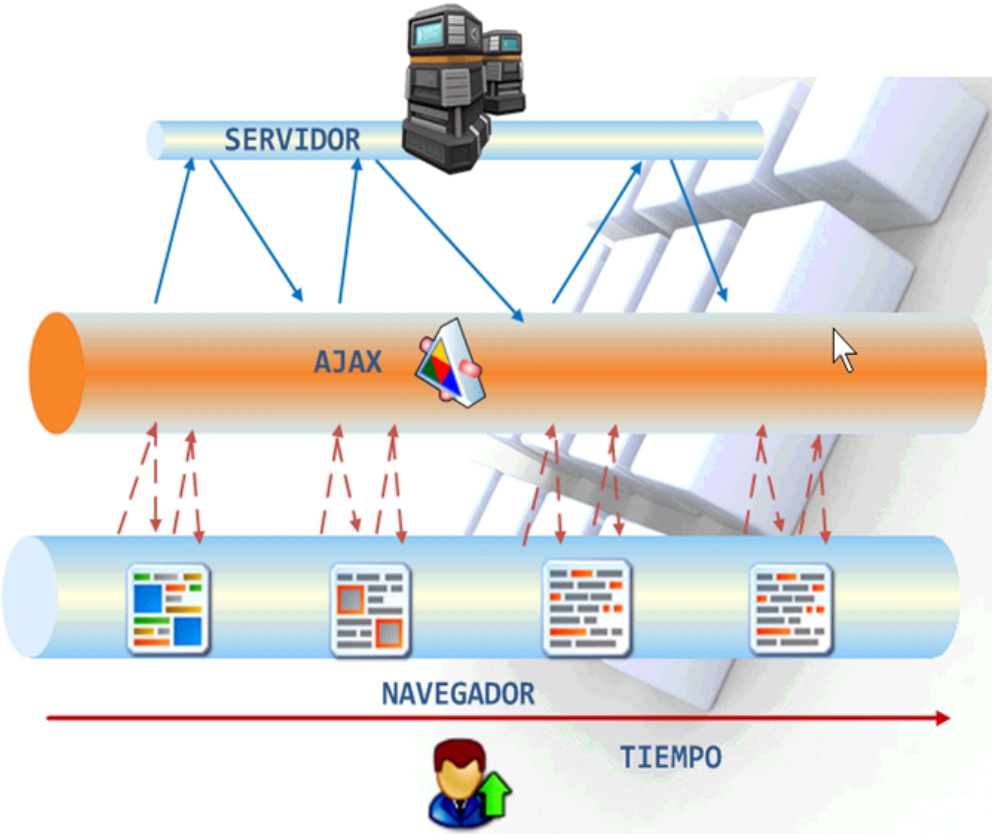
Definición

- **AJAX es la combinación de tres tecnologías ya existentes:**
  - XHTML y hojas de estilos en cascada (CSS) para el diseño que acompaña a la información.
  - Document Object Model (DOM) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones de ECMAScript como JavaScript, para mostrar e interactuar dinámicamente con la información presentada.
  - El objeto XMLHttpRequest para intercambiar datos asincrónicamente con el servidor web.
  - XML es el formato usado comúnmente para la transferencia devuelta por el servidor, aunque cualquier formato puede funcionar, incluyendo HTML preformateado, texto plano, JSON y hasta EBML.

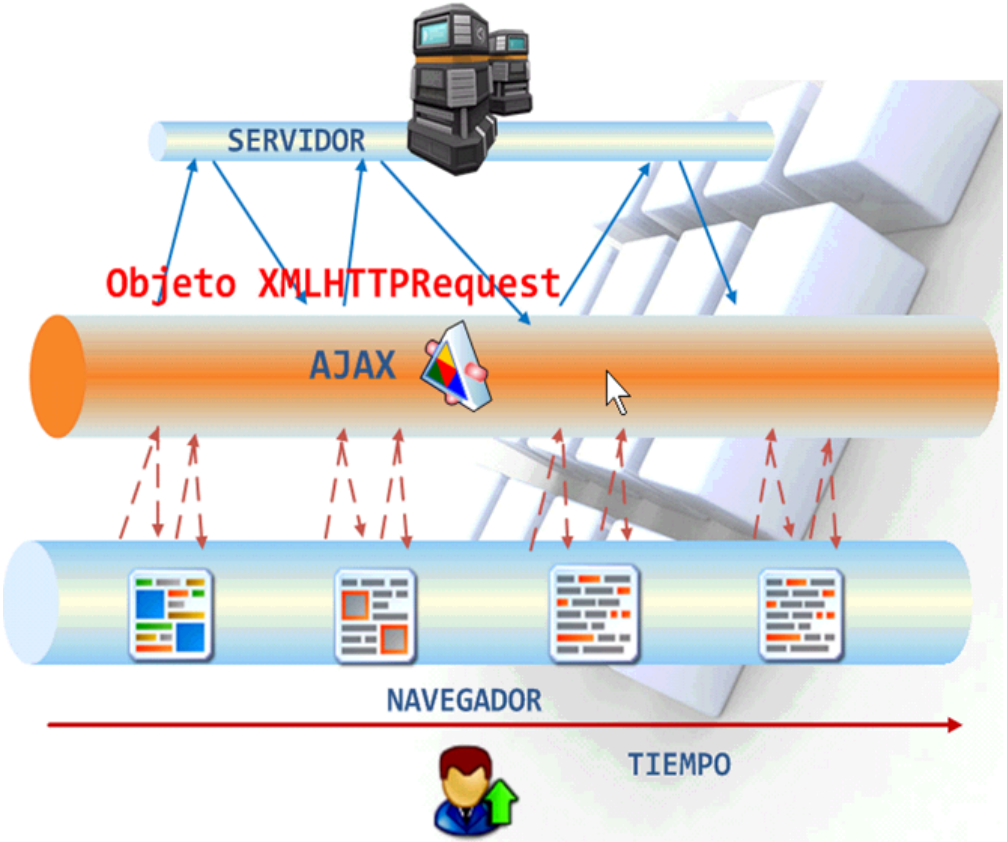
Conversación tradicional Cliente-Servidor



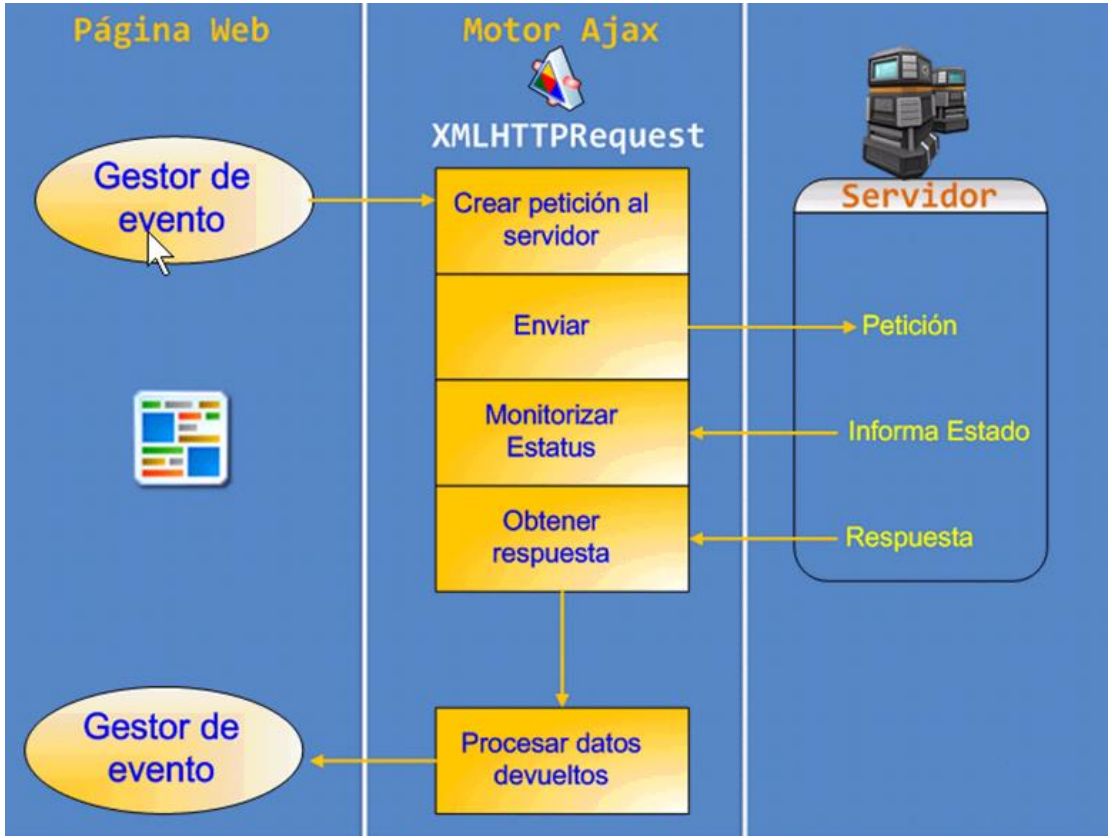
Interacción AJAX Cliente-Servidor



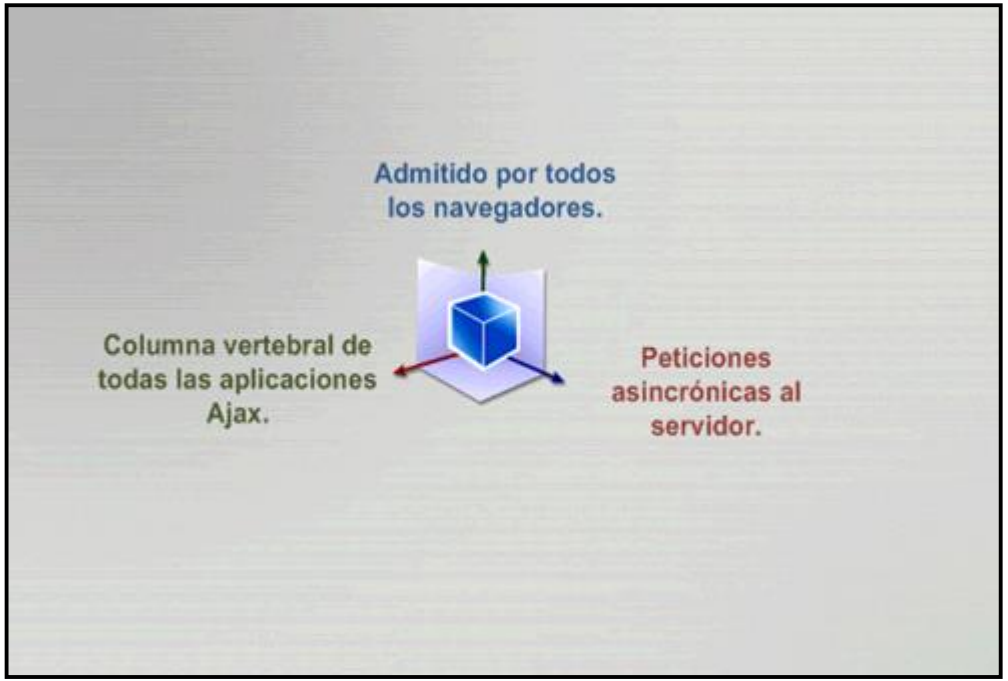
Interacción AJAX Cliente-Servidor



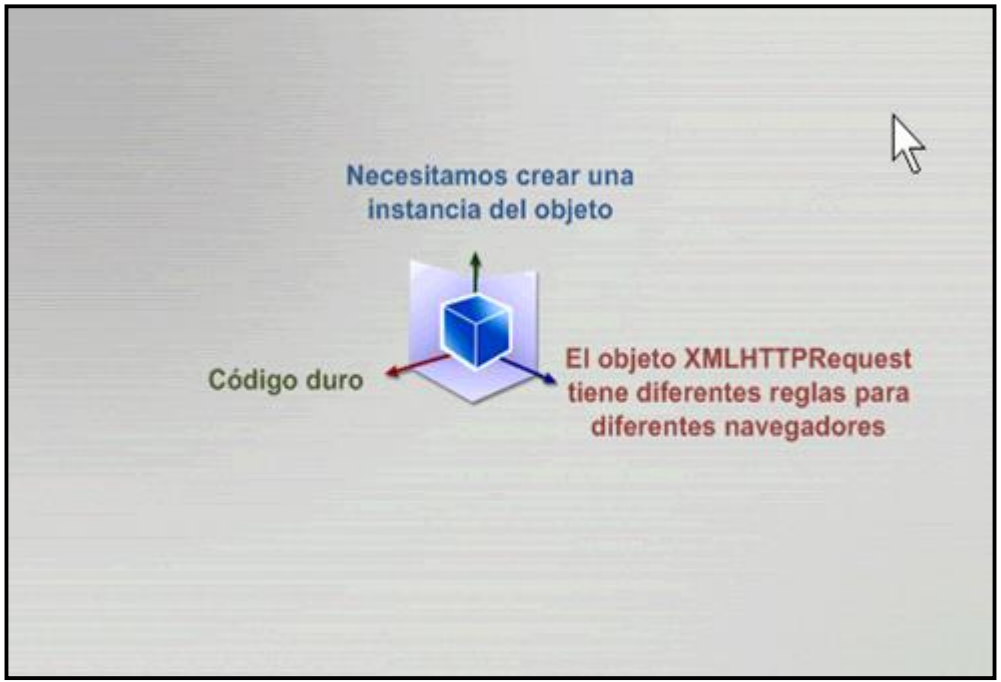
Interacción AJAX Cliente-Servidor



El objeto XMLHTTPRequest

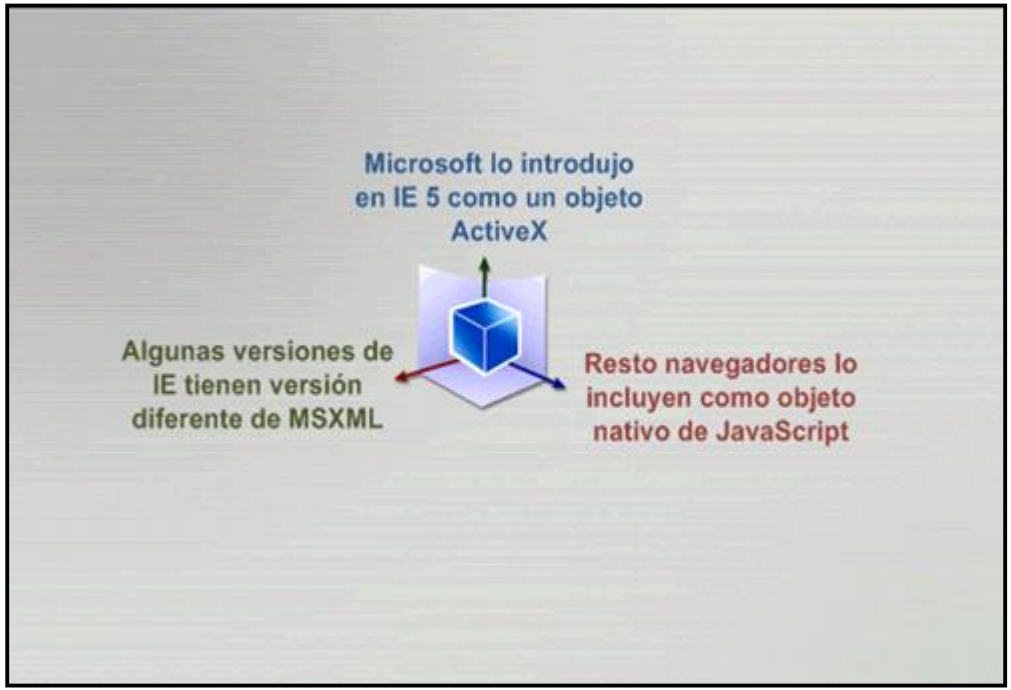


El objeto XMLHTTPRequest



El objeto XMLHTTPRequest

Diferentes reglas para diferentes navegadores.



El objeto XMLHTTPRequest

Diferentes reglas para diferentes navegadores.

El objeto XMLHTTPRequest

Diferentes reglas para diferentes navegadores.

```
function getXMLHTTPRequest() {
    var req;
    try {
        req = new XMLHttpRequest();
    } catch(err1) {
        try {
            req = new ActiveXObject("Msxml2.XMLHTTP");
        } catch (err2) {
            try {
                req = new ActiveXObject("Microsoft.XMLHTTP");
            } catch (err3) {
                req = false;
            }
        }
    }
    return req;
}
```

El objeto XMLHTTPRequest

Propiedades del objeto XMLHTTPRequest

Propiedad	Descripción
onreadystatechange	Determina el gestor de evento que será llamado cuando la propiedad readyState del objeto cambie.
readyState	Número entero que indica el estado de la petición: 0: No iniciada 1: Cargando 2: Cargado 3: Interactivo 4: Completado
responseText	Datos devueltos por el servidor como una cadena de texto.
responseXML	Datos devueltos por el servidor como un objeto XMLDocument que se puede recorrer usando las funciones de JavaScript DOM.
status	Código de estado HTTP devuelto por el servidor.
statusText	Mensaje de texto enviado por el servidor junto al código (status), para el caso de código 200 contendrá “OK”.

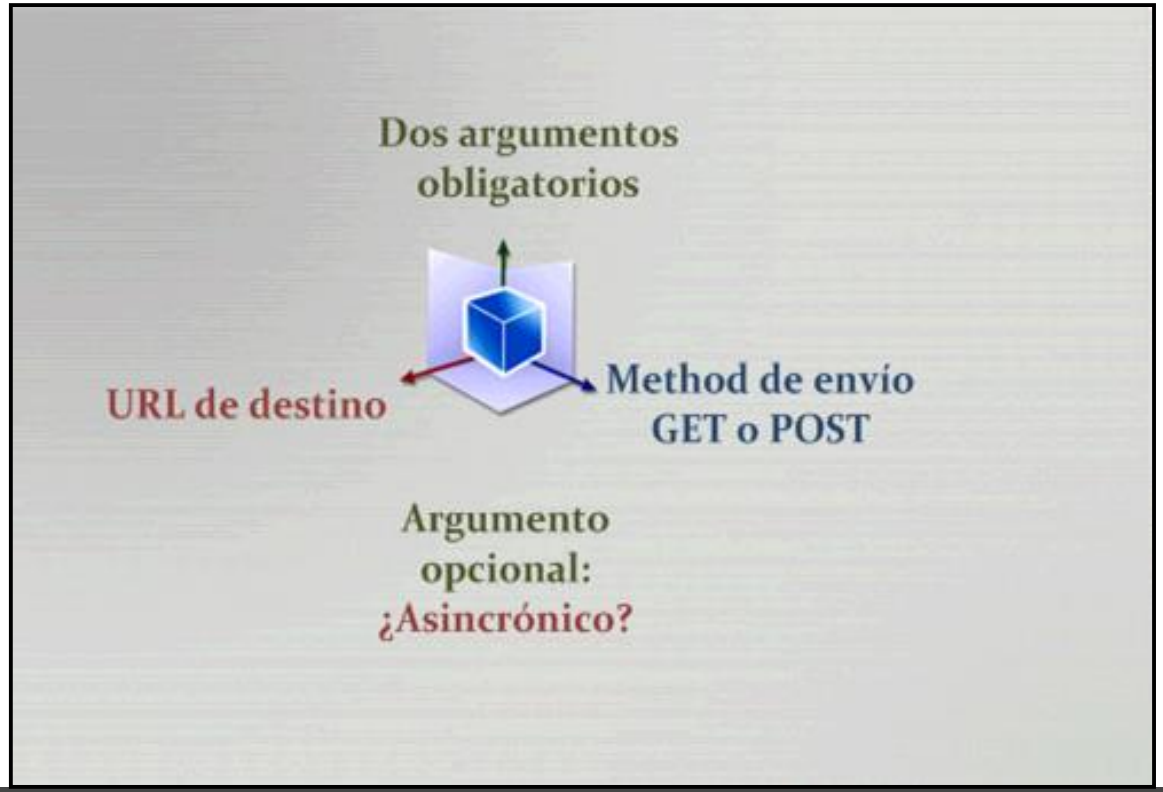
El objeto XMLHTTPRequest

Métodos del objeto XMLHTTPRequest

Propiedad	Descripción
abort()	Detiene la petición actual.
getAllResponseHeaders()	Devuelve todas las cabeceras como una cadena.
getResponseHeader(etiqueta)	Devuelve el valor de la etiqueta en las cabeceras de la respuesta.
open(método,URL,asíncrona)	Especifica un método HTTP (GET o POST), la URL objetivo, y si la petición debe ser manejada asincrónicamente (true o false).
send(contenido)	Envía la petición, opcionalmente con datos POST.
setRequestHeader(etiqueta,valor)	Establece el valor de una etiqueta de las cabeceras de petición.

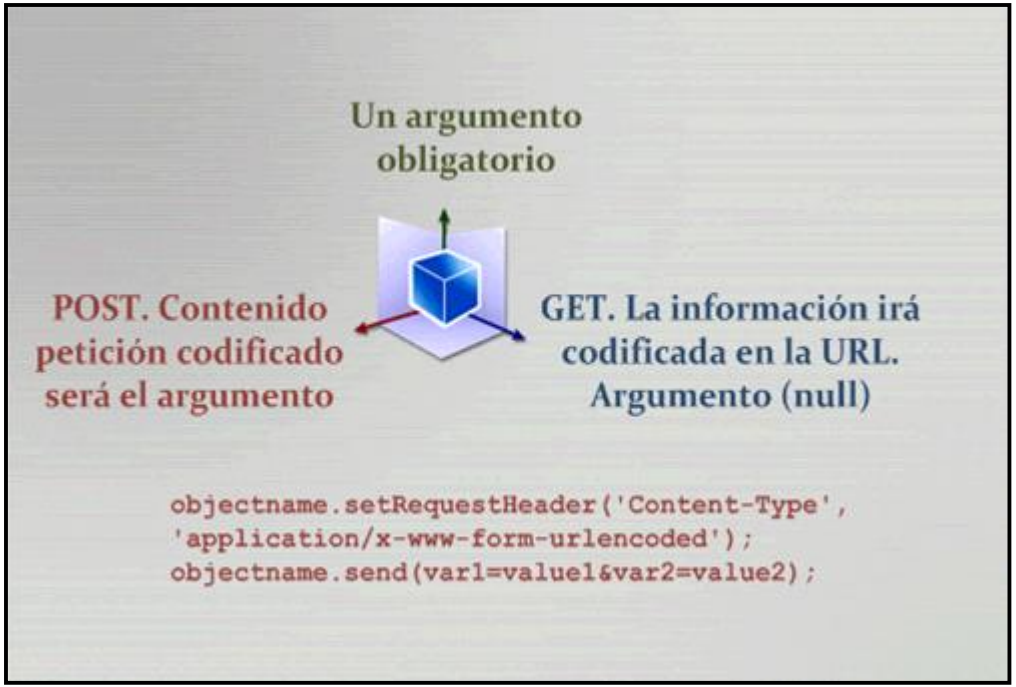
El objeto XMLHTTPRequest

- Método: open(método, URL, asíncrona)



El objeto XMLHTTPRequest

Método: send(contenido)

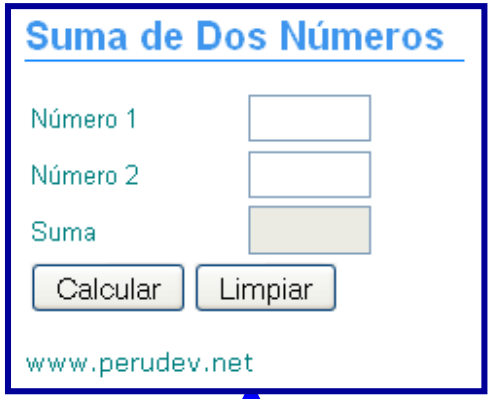


Implementación

- **Ejemplo 01**

En este ejemplo se está enviando los datos utilizando el método GET y se esta recibiendo un texto.

formulario.html



procesar.jsp

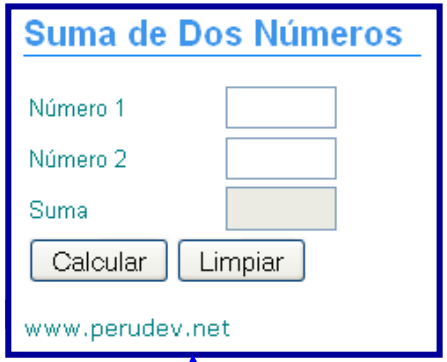
```
<%
// Datos
int n1 = Integer.parseInt( request.getParameter("n1") );
int n2 = Integer.parseInt( request.getParameter("n2") );
// Proceso
int suma = n1 + n2;
// Reporte
out.print(suma);
%>
```

Implementación

- **Ejemplo 02**

  En este ejemplo se está enviando los datos utilizando el método POST, se esta recibiendo un documento XML y la función de procesamiento se esta construyendo en línea.

formulario.html



procesar.jsp

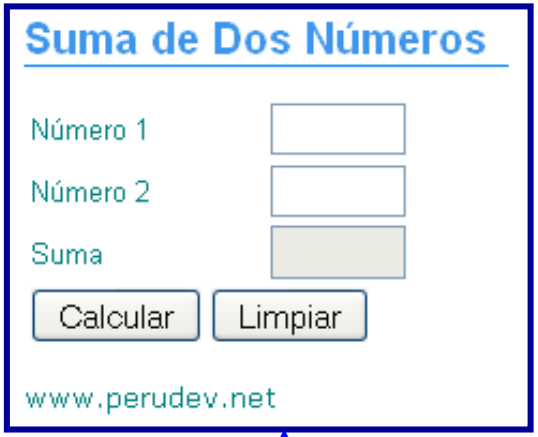
```
<%@page contentType="text/xml" %>
<%
// Datos
int n1 = Integer.parseInt( request.getParameter("n1") );
int n2 = Integer.parseInt( request.getParameter("n2") );
// Proceso
int suma = n1 + n2;
// Reporte
out.print( "<rpta><suma>" );
out.print( suma );
out.print( "</suma></rpta>" );
%>
```

Implementación

- **Ejemplo 03**

En este ejemplo se está enviando los datos utilizando el método POST, luego se retorna el resultado en formato **JSON** (Arreglo) y la función de procesamiento está utilizando la función **eval()** para convertir el resultado en un arreglo JavaScript.

formulario.html



procesar.jsp

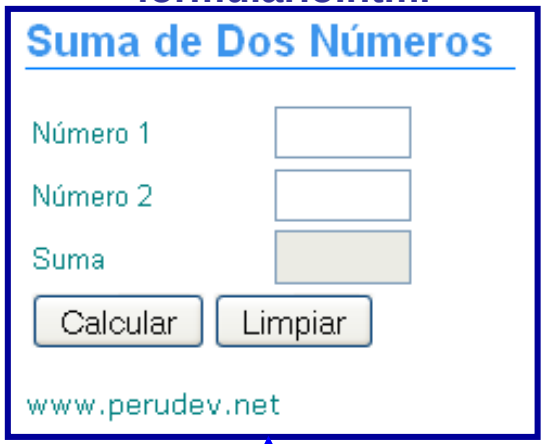
```
<%
// Datos
int n1 = Integer.parseInt( request.getParameter("n1") );
int n2 = Integer.parseInt( request.getParameter("n2") );
// Proceso
int suma = n1 + n2;
// Reporte
out.print( "[" + suma + "]" );
%>
```

Implementación

- **Ejemplo 04**

En este ejemplo se está enviando los datos utilizando el método POST, luego se retorna el resultado en formato **JSON** (Object) y la función de procesamiento está utilizando la función **eval()** para convertir el resultado en un objeto JavaScript.

formulario.html



procesar.jsp

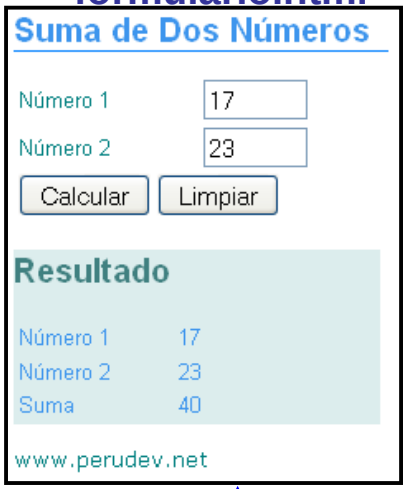
```
<%
// Datos
int n1 = Integer.parseInt( request.getParameter("n1") );
int n2 = Integer.parseInt( request.getParameter("n2") );
// Proceso
int suma = n1 + n2;
// Reporte
out.print( "{ valor:" + suma + "}" );
%>
```

Implementación

- **Ejemplo 05**

En este ejemplo se está enviando los datos utilizando el método POST y retorna una porción de código HTML que es visualizado en un **div** de nombre **divresultado**.

formulario.html



procesar.jsp

```
<%
// Datos
int n1 = Integer.parseInt(request.getParameter("n1"));
int n2 = Integer.parseInt(request.getParameter("n2"));
// Proceso
int suma = n1 + n2;
%>
<h2>Resultado</h2>
<table width="190">
    <tr>
        <td width="82">Número 1</td>
        <td width="96"><%=n1%></td>
    </tr>
    <tr>
        <td>Número 2</td>
        <td><%=n2%></td>
    </tr>
    <tr>
        <td>Suma</td>
        <td><%=suma%></td>
    </tr>
</table>
```

AJAX con JQuery

- La forma más sencilla de enviar una petición HTTP de forma asíncrona y mostrar el resultado en la página actual es utilizar la función load(). Esta se ejecuta sobre el elemento al que se va a añadir la respuesta, y se le pasa como argumento una cadena con el archivo a cargar:

```
$("#divResultado").load("productos.jsp");
```

AJAX con JQuery

- También se le puede enviar parámetros:

```
$("#divResultado").load("productos.jsp","cate=15");
```

- También se pueden utilizar los métodos `get()` y `post()` del objeto `jQuery`, en cuyo caso devolverá un resultado con los que tendrás que trabajar para generar la respuesta y mostrarla en el documento actual:

AJAX con JQuery

```
$.get("LogonController", {nombre: "gcoronelc",
pass:"secreto"}, function(data, textStatus,
XMLHttpRequest){
    $("#mensaje").html("Han devuelto: " + data);
});

$.post("LogonController", {nombre: "gcoronelc",
pass:"secreto"}, function(data, textStatus,
XMLHttpRequest){
    $("#mensaje").html("Han devuelto: " + data);
});
```

Conclusiones

- En estos momentos AJAX está revolucionando el modo en el que se programan las webs y el modo en el que se presentan. Gracias a AJAX se consigue una interfaz muy parecida a la de una aplicación de escritorio.

- La evolución de AJAX ha sido un hecho que ha marcado la tendencia en los lenguajes de programación en estos últimos años.

- Las aplicaciones de AJAX utilizan características bien documentadas presentes en todos los navegadores importantes en la mayoría de las plataformas existentes. Aunque esta situación podría cambiar en el futuro, en este momento, el uso de AJAX es efectivo entre plataformas.

- Para hacer mas productivo el desarrollo con AJAX se recomienda el uso de Framworks; como por ejemplo: Prototype, Scriptaculous, Xajax, jQuery, etc.

UNIVERSIDAD CIENTÍFICA DEL SUR

Eric Gustavo Coronel Castillo
gcoronelc.blogspot.com
egcc.ucs@gmail.com