AJAX *(Asynchronous Javascript and XML)*

¿Qué es AJAX?

Modelo “clásico” vs modelo AJAX de aplicaciones web

Algunos ejemplos de AJAX en acción

Ventajas e inconvenientes de AJAX

En resumen…

¿Qué es AJAX?

Ayax el Mayor y Ayax el Menor lucharon en Troya con Aquiles…

Ajax de Amsterdam es un equipo de fútbol…

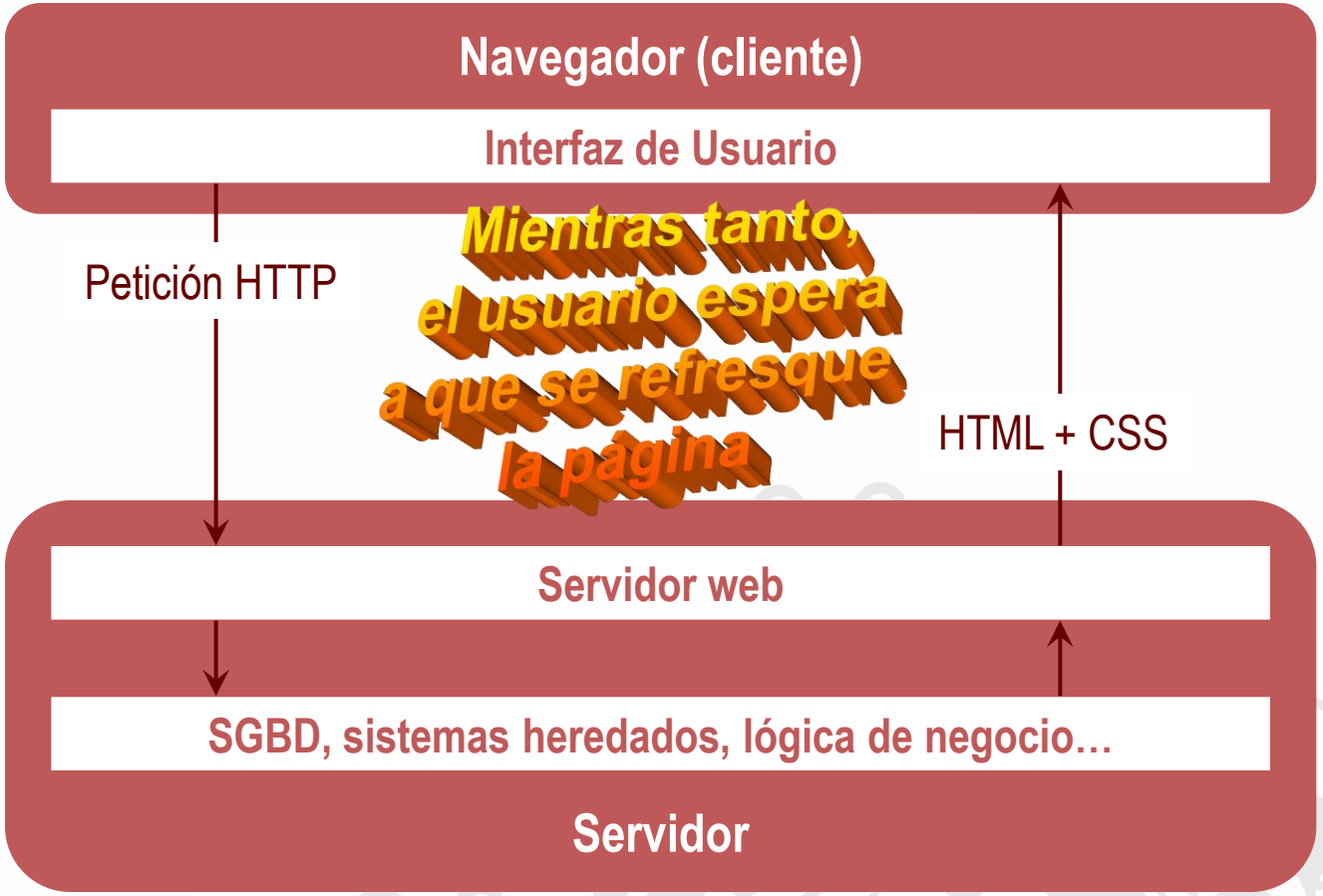
AJAX es una marca de productos de limpieza…

Un término "inventado" por Jesse James Garret para referirse a la utilización conjunta de las siguientes tecnologías:

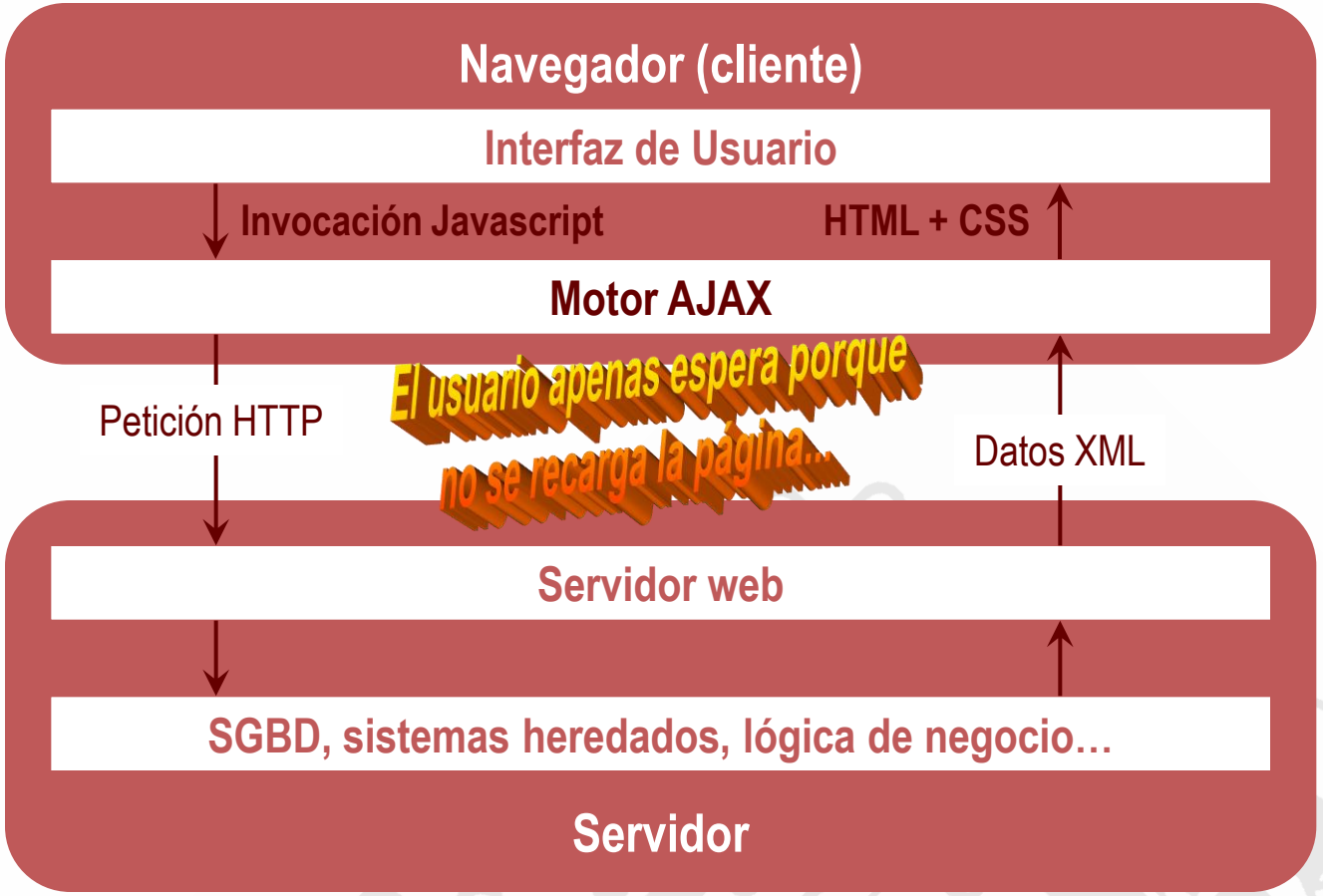
- Estándares de presentación: **XHMTL** y **CSS**
- Interacción: **DOM** *(Document Object Model)*
- Intercambio de datos: **XML** y **XSLT**
- **Interacción asíncrona con el servidor: XMLHttpRequest**
- Programación en el cliente: **Javascript**

Dicho de otro modo: aplicaciones web más **interactivas**, **rápidas** y **atractivas**.

Modelo “clásico” de aplicaciones web



Modelo AJAX de aplicaciones web



Algunos ejemplos de AJAX en acción

http://a9.com

http://labs.google.com/suggest

http://maps.google.es

http://mindset.research.yahoo.com

http://spreadsheets.google.com

http://www.gmail.com

http://www.google.com/calendar

http://gplv3.fsf.org/comments/gplv3-draft-2.html

http://www.wikimapia.com

http://www.writely.com

Ventajas de AJAX

- Interfaces de usuario más **“ricos”** ≡ experiencia de usuario próxima a la de **aplicaciones de escritorio**
- Menor consumo de **ancho de banda** (no hay necesidad de cargar una nueva página)
- Intercambio de datos XML = servicios web, RSS, etc.
- Está de **moda** ☺
- Gran variedad de **bibliotecas** para facilitar el desarrollo

Inconvenientes de AJAX

Problemas de **usabilidad**: los usuarios tienen que **"desaprender"** ciertas costumbres (p.ej. el botón para ir a la página anterior)

Problemas de **compatibilidad** con los distintos navegadores

Problemas de **IE** con el manejo de **texto "internacional"** no empaquetado como XML así como con el **"cacheado"** de las respuestas

Javascript no es un lenguaje que facilite ni el **mantenimiento** ni la **depuración**

Los **desarrolladores** tienen que **"cambiar el chip"**: se trata de **aplicaciones auténticas** no páginas web *cool*

En resumen…

El navegador contiene una aplicación, no un documento

El servidor no envía contenidos sino datos

El usuario interactúa de manera fluida y continua (muchas de las peticiones son implícitas)

Se trata de auténtico desarrollo de *software*

AJAX “crudo”

Elementos clave de AJAX

Tres patas para un banco…

Un servicio y cuatro ejemplos…

En resumen…

Elementos clave de AJAX

HTML Dinámico

Javascript: lenguaje débilmente tipado, empotrado en el navegador y empleado para interactuar con éste

Hojas de estilo (CSS): permite definir el estilo visual de la aplicación y modificarlo de manera sencilla (vía Javascript)

DOM *(Document Object Model):* modelo de objetos accesible mediante Javascript y que permite modificar la estructura de la página cargada en el navegador

Objeto XMLHttpRequest: permite que el navegador recupere datos (XML, texto plano, …) desde el servidor en segundo plano

Tres patas para un banco...

```
// Función para construir un objeto XMLHttpRequest
//
function initXMLHTTPRequest()…

// Función para realizar una petición al servidor
// empleando un objeto XMLHttpRequest//
function sendRequest(url,params,HttpMethod)…

// Función de "re-entrada" una vez el servidor
// responde al objeto XMLHttpRequest
//
function onReadyStateChange()…
```

Un servicio y cuatro ejemplos…

Pequeña “imitación” de Google Suggest

El servicio recibe una cadena de texto **`cad`** y responde con una lista de municipios asturianos que comiencen por **`cad`**

La primera versión envía texto plano y la segunda XML

Para interactuar tanto con el documento/aplicación en el lado del cliente como con el XML enviado por el servidor se utilizará el DOM

Referencias útiles

Material de consulta sobre **DOM**:

`http://www.mozilla.org/docs/dom/`

`http://developer.mozilla.org/en/docs/DOM`

Material de consulta sobre **AJAX**

`http://developer.mozilla.org/en/docs/AJAX`

`http://alexbosworth.backpackit.com/pub/67688`

Mínima referencia sobre **XMLHttpRequest**

`http://developer.mozilla.org/en/docs/XMLHttpRequest`

Dojo, *Rico* y *Yahoo! User Interface Library*

¿Qué es *dojo*?
Implementación de los ejemplos con *dojo*

¿Qué es *Rico*?
Implementación de los ejemplos con *Rico*

¿Qué es *YUI*?
Implementación de los ejemplos con *YUI*

Otros *frameworks* y bibliotecas para desarrollo AJAX
OpenAJAX Alliance

Escollos…

¿Qué es *dojo*?

Salón de entrenamiento de artes marciales dirigido por un *sensei*

Toolkit modular para el desarrollo de aplicaciones con **HTML dinámico** (y también **AJAX**)

Diferentes **perfiles**

Multinavegador: IE 5.5 o superior, Firefox 1.0 o superior, Safari 2.0, Opera 8.5, Konqueror 3.5

`http://www.dojotoolkit.org/`

`http://manual.dojotoolkit.org/`

`http://dojotoolkit.org/api/`

Implementación de los ejemplos anteriores con *dojo*

A tener en cuenta:

- Establecer la configuración de *dojo* (**`djConfig`**)
- Cargar **`dojo.js`** (incluirá el núcleo de *dojo* y los paquetes incluidos en el perfil seleccionado)
- Indicar con **`dojo.require`** qué paquetes van a utilizarse (no incluidos en el código **`dojo.js`** correspondiente al perfil)
- **`dojo.io.bind(petición)`**
- Atributos habituales en una petición:
  - **`url`**
  - **`content`**
  - **`load`**
  - **`error`**
  - **`mimetype`**
  - **`preventCache`**

¿Qué es *Rico*?

Johnny Rico es un soldado del cuerpo de Infantería Móvil en *Starship Troopers*

***Rico* es un *framework* basado en la biblioteca *Prototype* para desarrollar aplicaciones con HTML dinámico**

Dispone de gran variedad de efectos y *widgets*

Rico proporciona un objeto **`ajaxEngine`** para realizar peticiones AJAX, sin embargo…

…obliga a que el servidor envíe una respuesta XML con un formato específico ☹ que puede "envolver" XHTML o datos XML…

```
<?xml version="1.0" encoding="UTF-8"?>
<ajax-response>
     <response type="element" id="municipios">
               Allande
               Aller
               Amieva
               Avilés
     </response>
</ajax-response>
```

Implementación de los ejemplos anteriores con *Rico*

A tener en cuenta si se consume un servicio que envía XHTML envuelto en una "respuesta *Rico*":

- "Registrar" el servicio a consumir antes de nada (`ajaxEngine.registerRequest`)
- "Registrar" los elementos HTML a modificar una vez cargada la página y **antes** de realizar ninguna petición (`ajaxEngine.registerAjaxElement`)
- Realizar la petición (`ajaxEngine.sendRequest`)

A tener en cuenta si se consume un servicio que envía XML envuelto en una "respuesta *Rico*":

- "Registrar" el servicio a consumir al principio (`ajaxEngine.registerRequest`)
- Preparar y "registrar" un objeto para manipular la respuesta (al menos con los métodos `initialize` y `ajaxUpdate`)
- Realizar la petición (`ajaxEngine.sendRequest`)

¿Qué es *YUI*?

YUI es una cantautora japonesa

***Yahoo! User Interface Library*. Un conjunto de componentes y *widgets* escritos en Javascript para desarrollar aplicaciones DHTML**

También incluye un componente para hacer peticiones vía XMLHttpRequest (**`YAHOO.util.Connect`**)

`http://developer.yahoo.com/yui/`

`http://developer.yahoo.com/yui/connection/`

Implementación de los ejemplos anteriores con *YUI*

El método `YAHOO.util.Connect.asyncRequest(método, url, manejador, datos)` se emplea para las invocaciones

- `método` puede ser `GET` o `POST`
- `url` apunta al servicio a consumir
- `manejador` puede ser una función de re-entrada o un objeto que implementará los métodos `success` y `failure` para actuar en función de la respuesta recibida
- `datos` se emplea únicamente con `POST`

El manejador recibe un objeto respuesta con los siguientes atributos:

- `tId`: un identificador numérico único para las transacciones originadas en el cliente
- `status`: código de estado HTTP (p.ej. `200`, `404`, `500`, etc.)
- `statusText`: mensaje de texto asociado al código anterior (p.ej. `OK`, `Not Found`, `Internal Server Error`, etc.)
- `getResponseHeader[etiqueta]`: valor de la etiqueta de la cabecera especificada
- `responseText`: la respuesta enviada por el servidor como texto plano
- `responseXML`: un objeto `XMLDocument` si el servidor envía datos XML
- `argument`: el/los argumento/s definidos como un atributo del objeto de re-entrada

Hay más bibliotecas, *frameworks* y *toolkits* Javascript…

Escollos…

En realidad **uno**… ¡pero muy **grande**!

El objeto **`XMLHttpRequest`** no admite peticiones enviadas a ninguna otra máquina que no sea aquella de la que se descargó la página

En otras palabras: no se puede acceder (directamente) a servicios ofrecidos por terceros (p.ej. *Google Maps*, *Yahoo! Search*, *feeds RSS*, etc.)

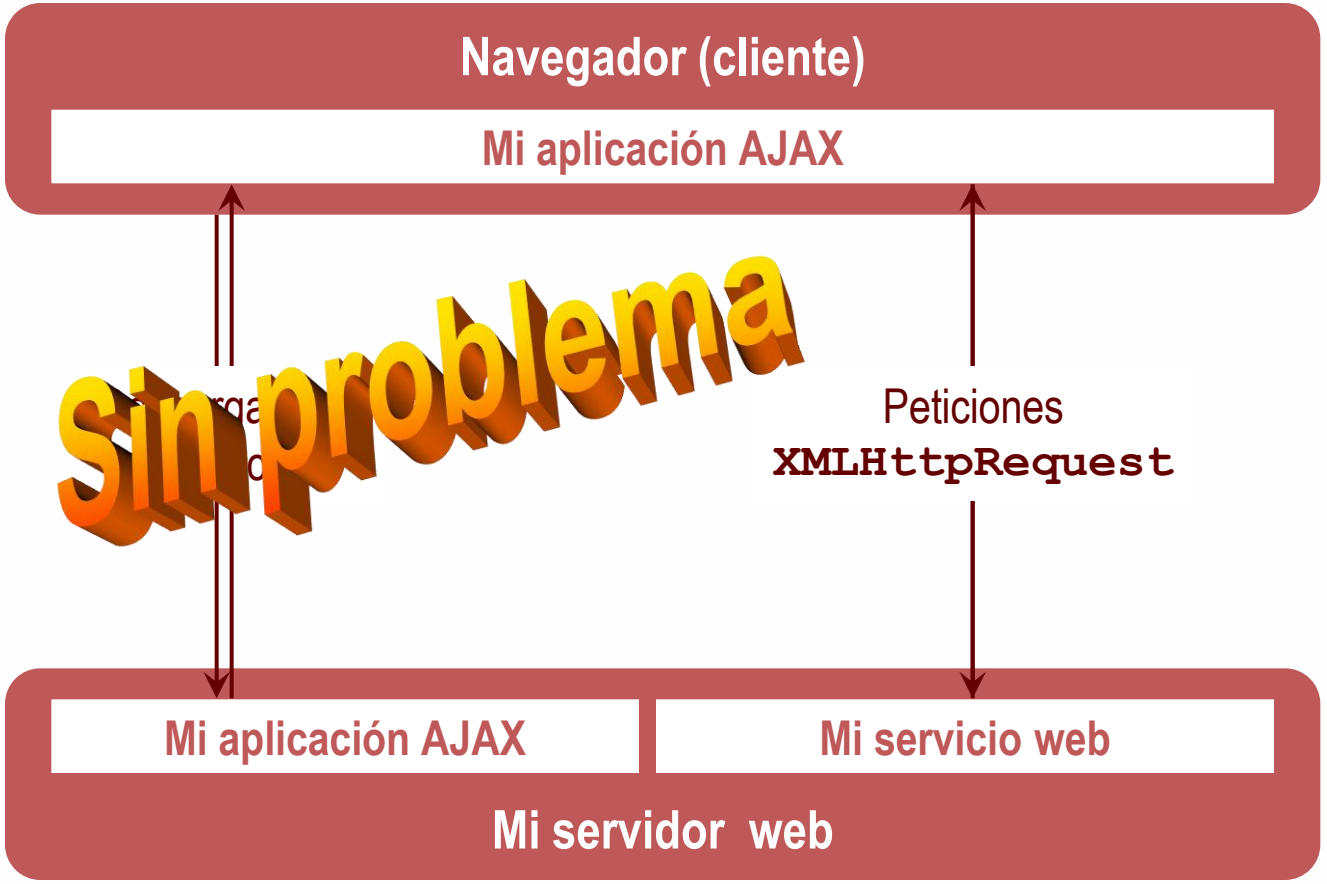
The “cross-domain” problem

Aplicaciones AJAX que consumen servicios “locales”

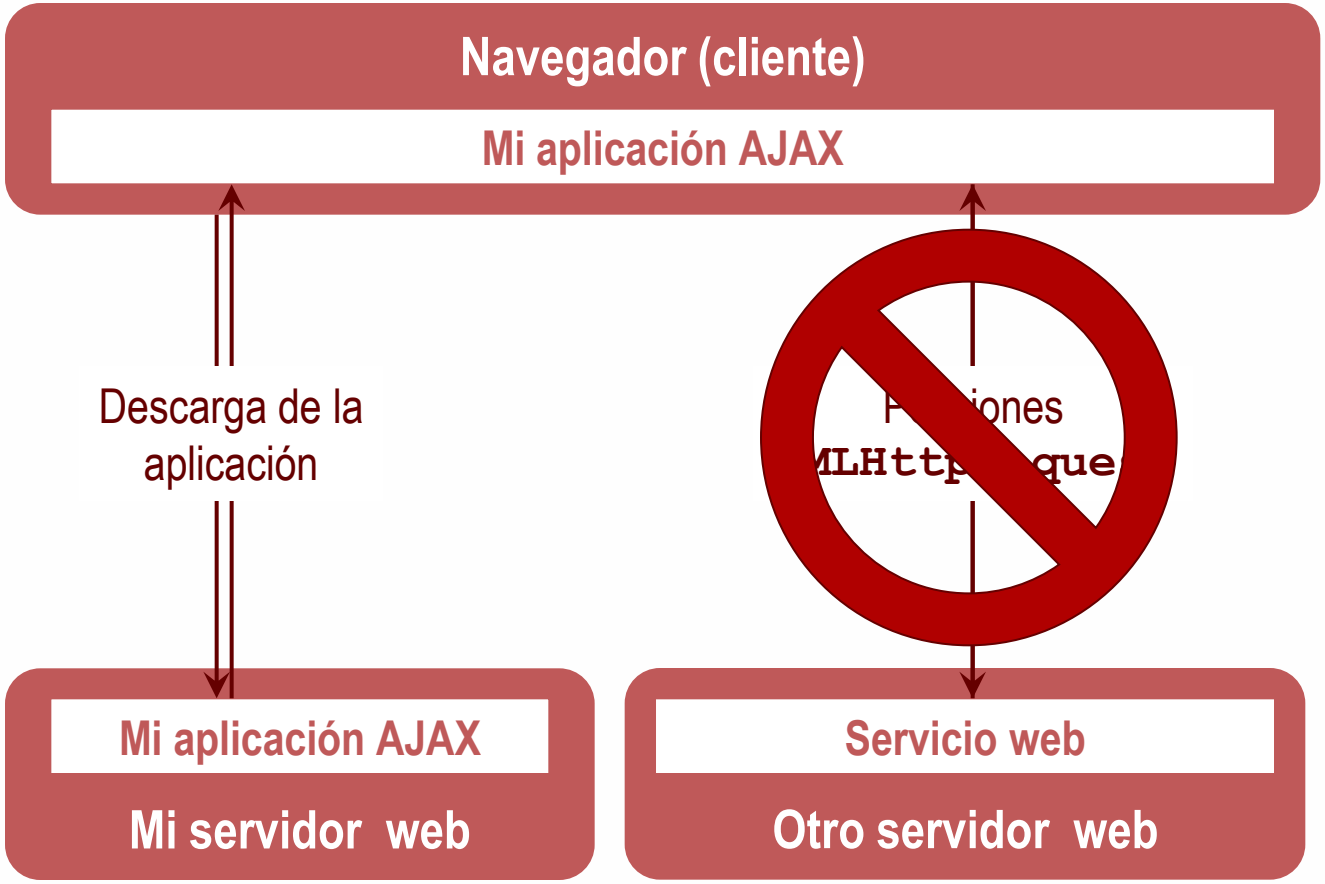
Aplicaciones AJAX que (pretenden) consumir servicios “ajenos”

Utilizando un *proxy* para consumir servicios “ajenos” con **`XMLHttpRequest`**

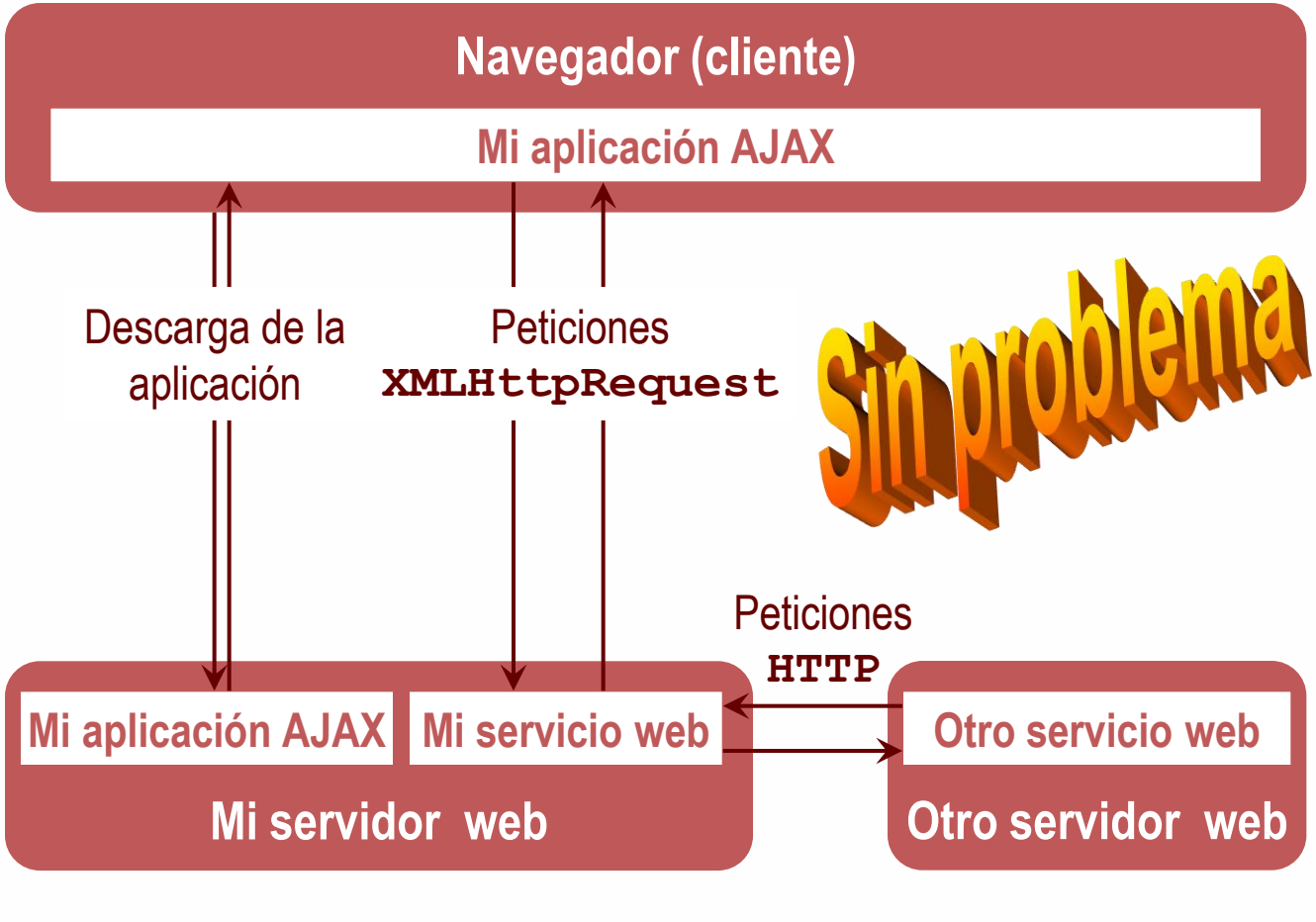
Aplicación AJAX + servicio web "local"



Aplicación AJAX + servicio web ajeno *(cross-domain problem)*



Servicio web “local” *proxy* servicio web ajeno



Servicio web "local" *proxy* servicio web ajeno

En el ejemplo se muestra un *proxy* implementado en PHP con *cURL* que permite descargar *feeds* RSS de la BBC

Más información y soluciones análogas:

http://developer.yahoo.com/javascript/howto-proxy.html

http://www.xml.com/pub/a/2005/11/09/fixing-ajax-xmlhttprequest-considered-harmful.html

http://ajaxpatterns.org/Cross-Domain_Proxy

http://developer.yahoo.com/javascript/howto-ajax.html

http://www.troywolf.com/articles/php/class_http/proxy.phps

http://www.php.net/manual/en/ref.curl.php

¿Qué es AJAX?

AJAX “crudo”

Dojo, *Rico* y *Yahoo! User Interface Library*

***HTML_AJAX* = AJAX + PHP**

“Taller” sobre Servicios Web + AJAX

HTML_AJAX = AJAX + PHP

¿Qué es *HTML_AJAX*?

Instalación de *PEAR* y *HTML_AJAX*

Aspectos básicos de *HTML_AJAX*

El ejemplo de costumbre implementado con *HTML_AJAX*

Otros *frameworks* y bibliotecas para desarrollo AJAX en PHP

Resumen…

HTML_AJAX

HTML_AJAX es una biblioteca PHP que permite desarrollar aplicaciones AJAX (tanto la parte cliente como la servidora)

Se distribuye como un paquete **PEAR**

`http://pear.php.net/package/HTML_AJAX`

`http://wiki.bluga.net/HTML_AJAX/HomePage`

PEAR

PHP Extension and Application Repository

Sistema de distribución de componentes PHP bajo la forma de “paquetes”

El objetivo es ofrecer a los desarrolladores PHP:

- Una biblioteca de código abierto bien organizada
- Un sistema para distribuir y mantener su código
- Un estilo de codificación normalizado

`http://pear.php.net`

Antes de empezar…

Instalación de *PEAR* con *AppServ 2.4.7*

Crear en `C:\AppServ\www` un directorio `PEAR` (`C:\AppServ\www\PEAR\`)
Descargar `http://go-pear.org/` en `C:\AppServ\www\PEAR\go-pear.php`
Apuntar el navegador a `http://localhost/PEAR/go-pear.php`
Configurar el *proxy* en caso necesario (en UniOvi `156.35.14.6:8888`)
¡Atención! Cambiar las barras inclinadas hacia la derecha por barras hacia la izquierda (aún así habrá algún problemilla)
Indicar la ruta **completa** hacia `php.exe` (`C:\AppServ\php\cli\php.exe`)
Marcar el *checkbox* para instalar paquetes adicionales
Instalar
Ignorar el enlace al *frontend* (incorrecto), utilizar `http://localhost/PEAR/`
Si se muestra un aviso como `WARNING: channel "pear.php.net" has updated its protocols, use "channel-update pear.php.net" to update` continuar con el siguiente paso, si no pasar a instalación de *HTML_AJAX*
Abrir una ventana *MS-DOS*
Cambiar el directorio de trabajo a `C:\AppServ\www\PEAR\`
Ejecutar `pear channel-update pear.php.net`
Debería aparecer el mensaje `Update of Channel "pear.php.net" succeeded`
Editar `C:\Windows\php.ini` y añadir `include_path = ".;c:\appserv\www\pear\pear"`
Reiniciar Apache

Antes de empezar…

Instalación de *HTML_AJAX*

Apuntar el navegador hacia **`http://localhost/PEAR/`**
Ir a la sección de configuración
Cambiar el campo **`Preferred Package State`** de **`stable`** por **`alpha`** (a fecha del curso no hay versión estable de *HTML_AJAX*)
Pulsar el botón *Go!*
En caso de mensajes de error, ignorar
Ir a la sección de gestión de paquetes
Avanzar 2 páginas (categorías 11 a 15)
Buscar *HTML_AJAX*
Añadir el paquete (icono cruz verde)
Si el paquete se ha instalado correctamente debería aparecer un *tick*
Editar un *script* PHP con el siguiente código

```
<?php
    include 'HTML/AJAX/Server.php';
    $server = new HTML_AJAX_Server();
    $server->handleRequest();
?>
```

Apuntar el navegador hacia el *script*
Si *PEAR* y *HTML_AJAX* están correctamente instalados y configurados no debería haber ningún error

Aspectos básicos de *HTML_AJAX*

HTML_AJAX permite **invocar código PHP desde Javascript**

- **Consejo**: organizar el código PHP como métodos de una clase y utilizar un objeto Javascript a modo de *proxy*

Dispone de diversos **"serializadores"** para el intercambio de los datos (*null*, ***JSON***, PHP, *urlencoded*, XML)

- El serializador XML es aún muy reciente (agosto 2006)

Genera la mayor parte del código Javascript necesario para la aplicación

El programador puede limitarse a preparar funciones Javascript para invocar los métodos remotos y procesar las respuestas

Al trabajarse "a los dos lados" (cliente y servidor) ya no hay que preocuparse por el *"cross-domain problem"* (ya tenemos un servidor donde se pueden consumir los servicios web ajenos)

Avisos a navegantes…

- Puesto que se va a trabajar con PHP hay que ser consciente de las diferencias entre PHP 4 y PHP 5 ☹
- *HTML_AJAX* está aún en versión alpha ☹
- La "documentación" es escasa ☹

Fragmentos “habituales” de código *HTML_AJAX*

En el lado del servidor (PHP):

```
include 'HTML/AJAX/Server.php';
$server = new HTML_AJAX_Server();
$server->registerClass(new Clase(),NombreClase,array(nombreMetodo));
$server->setSerializer(SERIALIZADOR);
$server->handleRequest();
// Por supuesto, código PHP para la Clase
```

En el lado del cliente (HTML+Javascript):

```
<script type="text/javascript" src="servidor.php?client=all"></script>
<script type="text/javascript" src="servidor.php?stub=all"></script>
<script>
      var codigoReentrada = {
            nombreMetodoRemoto:function (resultado) {
                  // Código
            }
      }
      …
      var servicioRemoto=new NombreClaseRemota(codigoReentrada);
</script>
servicioRemoto.nombreMetodoRemoto(argumentos); // En alguna parte…
```

Implementación del ejemplo con *HTML_AJAX*

En el lado del servidor:

Se ha utilizado el serializador XML

HTML_AJAX mapea funciones y objetos nativos PHP sobre funciones y objetos nativos Javascript por lo que no basta con elaborar una cadena que "parezca" XML, es necesario construir un objeto XML (en PHP 4 con *DOM XML*)

Al contrario que con las implementaciones "cruda", *dojo*, *Rico* y *YUI* que envían los argumentos codificados en la URL con *HTML_AJAX* (en su versión actual) al fijar una serialización XML se obliga al cliente hacer peticiones XML

En el lado del cliente:

Apenas diferencias con ejemplos anteriores

Se ha tenido que construir una petición XML

Internet Explorer: `xmlDoc = new ActiveXObject("Microsoft.XMLDOM");`

Firefox: `xmlDoc = document.implementation.createDocument("","",null);`

En ambos casos utilización del **DOM**: `createElement`, `createTextNode` y `appendChild`

Hay más bibliotecas, *frameworks* y *toolkits* PHP para AJAX…

AJASON *Ajax Agent* *AjaxAC* *Cajax* *Claw* *DutchPIPE*

Flexible Ajax Framework *Guava* ***HTML_AJAX***

HTS Web Application Framework *JPSpan* *My-BIC* *NanoAjax* *Novulo*

PAJAJ *PAJAX* *phpAjaxTags* *PHPWebBuilder* *Pipeline* *Qcodo*

Sajax *SimpleJax* *Stratos PHP Framework* *Symfony* *Tigermouse*

TinyAjax *XAJAX* *XOAD* *Zephyr*

Y ya para terminar…

A modo de resumen…

- **AJAX supone combinar varias tecnologías** (DOM, peticiones HTTP y Javascript para pegarlo todo)
- El meollo está en el objeto/componente **`XMLHttpRequest`**
- Existen multitud de **bibliotecas Javascript** para implementar aplicaciones AJAX (incluyendo fanfarrias **DHTML**…)
- Existen multitud de bibliotecas para que lenguajes de generación dinámica de páginas (p.ej. PHP) puedan usarse para desarrollar AJAX
- Ventaja de los últimos: **exponer código legado** para su invocación desde Javascript
- **`XMLHttpRequest`** no puede hacer peticiones a máquinas ajenas *(**cross-domain problem**)* pero puede solucionarse con un *proxy* en el servidor

Para profundizar…

- Sólo hemos trabajado con peticiones **`GET`**, **`POST`** es análogo pero el interfaz de usuario debe dejar claro que se han almacenado los datos
- **¿Qué es JSON?** **J**ava**S**cript **O**bject **N**otation, un formato de intercambio de datos que puede procesarse de manera trivial en Javascript con la función **`eval`**

AJAX *(Asynchronous Javascript and XML)*