COMET: UN SIGUIENTE PASO AL AJAX MOVIENDO DE LAS APLICACIONES WEB TRADICIONALES A UN NUEVO ESTILO.

Luis Enrique Oviedo Chaparro
Facultad de Ciencias y Tecnología, Universidad Católica de Asunción
Asunción, Paraguay
le.oviedo@gmail.com

RESUMEN

La Tecnología COMET consiste en la aplicación de una "vieja" técnica Web que lentamente está resucitando desde las profundidades de la historia, el cual nos permite crear aplicaciones colaborativas, interactivas y actualizadas , sacando un mejor provecho a las capacidades de los navegadores actuales.

INTRODUCCIÓN

El paradigma tradicional de la Web es *Síncrona*. Esto indica que para cada pedido desde un cliente (Navegador Web) existe una correspondiente respuesta de un servidor (Servidor Web). Cuando una página Web es visualizada, los datos contenidos en la misma son estáticos en el navegador del usuario y estos datos no son actualizados hasta que la página sea refrescada nuevamente. Sin embargo existe un número creciente de aplicaciones que necesitan una visualización de los datos más recientes, que son continuamente actualizados en tiempo real. Algunos ejemplos de estas aplicaciones son:

- Precios de Stock de los sitios de compra on-line, principalmente.
- Resultados de Competencias Deportivas que necesiten actualizar constantemente sus resultados.
- Sitios de Apuestas.
- Mensajes intercambiados a través de las comunidades virtuales.

Son pocos ejemplos de los sistemas que, por la necesidad de ofrecer lo máximo en usabilidad y calidad para el usuario, requiere una continua actualización de los datos visualizados en el navegador. En síntesis, debido a la dinámica propia del servicio, se hace crítica mantener una actualización constante de la información requerida por el usuario.
Algunas aplicaciones utilizan una técnica de *polling* donde sea requerida una actualización automática. Mediante esta técnica resolvemos parcialmente el problema, debido a que:

- La frecuencia de actualización no puede ser alta, debido a que el paradigma se mantiene *síncrono* (requerimiento/respuesta), lo que implica que también es imposible recibir datos en tiempo real.
- El ancho de banda ocupado es alto, para cada pedido por parte del navegador, es transferida la página web completa, en vez de transferirse únicamente los datos que necesitan ser actualizados.
- El impacto en los Servidores Web son enormes, porque necesita complacer una alta cantidad de pedidos de página inclusive si los usuarios están inactivos.

Para solucionar estos problemas desde la fuente, es necesario un cambio de paradigma. En otras palabras, algún sistema *push* o de *streaming* es necesario con un mecanismo que provea un continuo flujo de datos desde el servidor hasta el cliente, sin que el cliente necesite realizar un pedido de la página web entera, realizando en vez de lo anterior el manejo adecuado a una tipología de datos y solamente esperar la recepción de las actualizaciones en tiempo real de los datos, a medida que vayan ocurriendo.

EVOLUCIÓN DE LA RED

Cuando la red inició, nos encontrábamos en un entorno estático, con páginas en HTML que sufrían pocas actualizaciones y no tenían interacción con el usuario. Este estado de la red es lo que se denominó como Web 1.0.

Continuamente, la red fue creciendo debido al éxito de las páginas web. Este éxito a su vez tuvo como consecuencia la necesidad de ofrecer nuevos servicios de carácter cada vez más dinámico. Por tanto la red fue apuntando hacia servicios donde es necesaria la actualización de los datos en un tiempo cada más corto.
Así la red dio un pasó mas, y este paso consistió en webs dinámicas donde las CMS (Content Manager System, Sistema de Gestión de Contenido), que son interfaces que controlan una o varias bases de datos donde se alojan todos los contenidos del sitio en cuestión, servían páginas HTML dinámicas creadas sobre el vuelo desde una base de datos en actualización. A este paso algunos lo consideraron como Web 1.5.

Al situarnos en este estado, para las páginas web, el conseguir hits (visitas) y la estética visual eran considerados como unos factores muy importantes, por lo que los principales propulsores y desarrolladores de avances en al red fueron vislumbrando que el uso de las redes debería estar orientado a la interacción y redes sociales, que pueden servir contenido que explota los efectos de las redes con o sin crear webs interactivas y visuales. Es decir, estos sitios con estas características cumplen más bien funciones de puntos de encuentro, o webs dependientes de usuarios, que como webs tradicionales. A este estado es donde se pretende llegar al mencionar la Web 2.0.

Sintetizando, La Web 2.0 se refiere a la transición percibida en Internet desde las webs tradicionales a aplicaciones web destinadas a usuarios. La gente involucrada en esta idea espera que los servicios sustituyan a las aplicaciones de escritorio en muchos usos.

Comparación entre la Web 1.0 y la Web 2.0
La infraestructura de la Web 2.0 es compleja y evoluciona, incluye el software del servidor, sindicación de contenidos, protocolos de mensajes, navegadores basados en estándares, y aplicaciones para clientes . Una Web se puede decir que esta usando tecnología de Web 2.0 si se caracteriza por las siguientes técnicas:

- Técnicas de aplicación no intrusita (como AJAX).

- Java Web Star.
- CSS , marcado XHTML válido semánticamente.
- URLs sencillas y con significado.
- Soporte para postear en un blog.
- Algunos aspectos de redes sociales.

Existen otras técnicas, cuyo uso también implican usar tecnología de la Web 2.0, pero los en general los principios o bases a seguir son:

- El sitio no debe actuar como un “jardín cerrado”, la información debe poder introducirse y extraerse fácilmente.
- Los usuarios deben controlar su propia información.
- Basada exclusivamente en la Web. Los sitios Web 2.0 con más éxito pueden ser utilizados enteramente desde un navegador .

En el siguiente cuadro observamos una comparativa de los servicios y páginas web que utilizan características de la Web 1.0 y de Web 2.0:

WEB 1.0	WEB 2.0
DoubleClick	Google Adsense
Ofoto	Flickr
Akamai	BitTorrent
mp3.com	Napster
Britannica Online	Wikipedia
webs personales	blogging
evite	upcoming org y EVDB
especulación de nombres de dominio	optimización en máquinas de búsqueda
páginas vistas	coste por click
screen scraping	servicios web
publicar	participación
Páginas estáticas	CMS
directorios (taxonomías)	etiquetas ("folksonomy")
stickiness	sindicación

El siguiente cuadro ilustra el paso intermedio entre lo llamado Web 1.0 y la Web 2.0 y las aplicaciones webs intervinientes:

WEB 1.0	WEB 1.5	WEB 2.0
Páginas Personales	Wikis	Blogging
Email/ Grupo de Noticias	Foros de Discusión	RSS-Sindication
mp3	Napster	iTunes
Terraserver	MapQuest	Google Maps
Británica Online		Wikipedia
		Flickr

Sintetizando la filosofía de la Web 2.0, la misma no es un cambio tecnológico en su totalidad, pero sí un cambio en la filosofía con la que los usuarios y las empresas se plantean Internet. En una síntesis de la síntesis, podemos decir que:

“la Web 1.0 es la Read Only Web, mientras que la Web 2.0 es la Writable Web”

La revisión exhaustiva de las técnicas y servicios utilizados en la evolución de la Web no es el motivo de este trabajo, aunque dentro de esta nueva manera de entender la Web, están enmarcadas ciertas tecnologías que cumplen los roles fundamentales en el desarrollo del mismo, AJAX y subsecuentemente COMET.

AJAX: UN NUEVO ENFOQUE PARA LAS APLICACIONES WEB

Una de las nuevas técnicas de programación utilizadas para el desarrollo del nuevo estilo de las páginas web es AJAX:
AJAX, Es la técnica de utilizar una serie de tecnologías de forma conjunta como XML, JavaScript y objetos de cliente (MICROSOFT.XMLHTTP o XMLHttpRequest) que permite a las aplicaciones web comportarse de una manera, podría decirse similar a la de las aplicaciones de escritorio, consiguiendo para esto una navegación más ágil y rápida. Más dinámica.

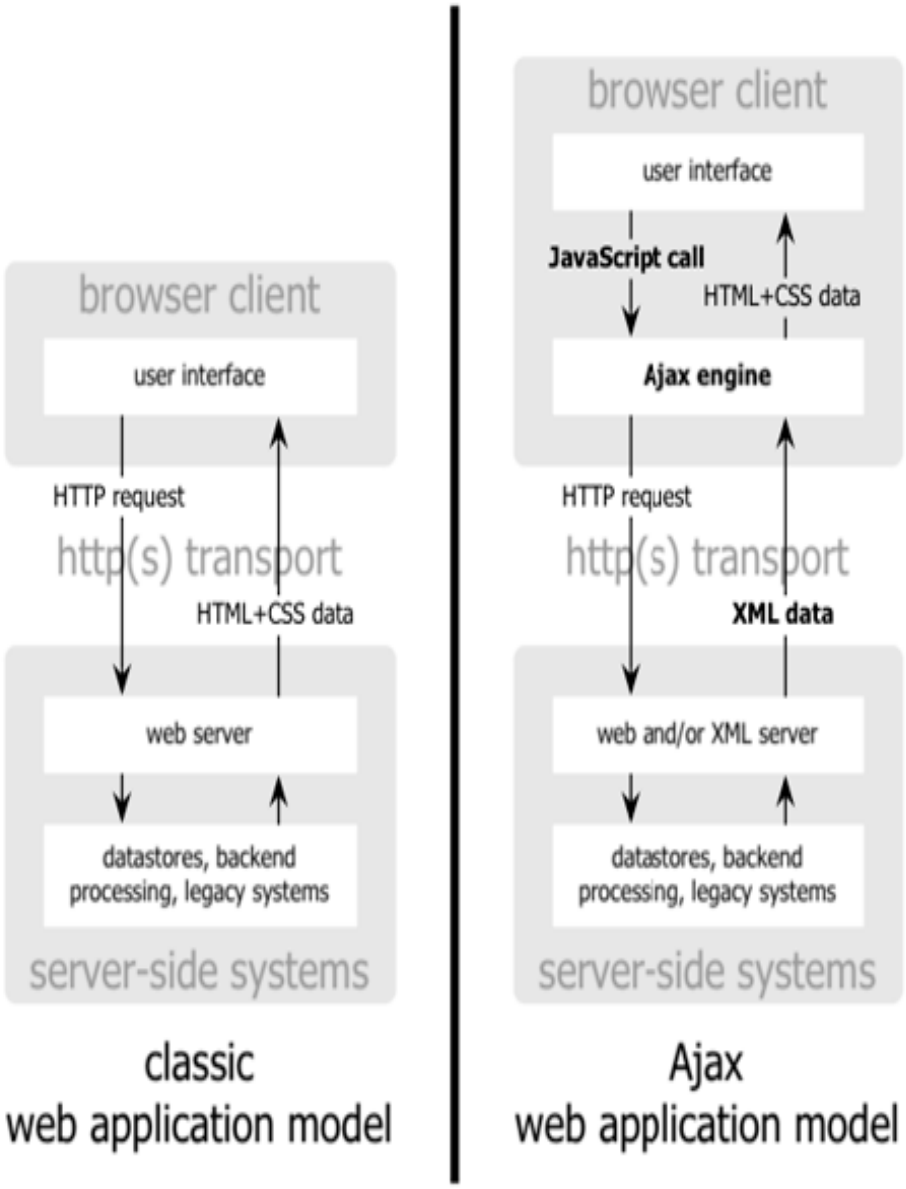
AJAX es el acrónimo de **Asynchronous Javascript And XML**:

- **Asynchronous**
  Las peticiones pueden ser síncronas o asíncronas, las asíncronas engañan más porque el cliente sigue trabajando con la aplicación mientras se resuelve la petición.
- **JavaScript**
  Lenguaje que controla las acciones del cliente
- **XML**
  Suele o puede ser el contenido de los mensajes de solicitud

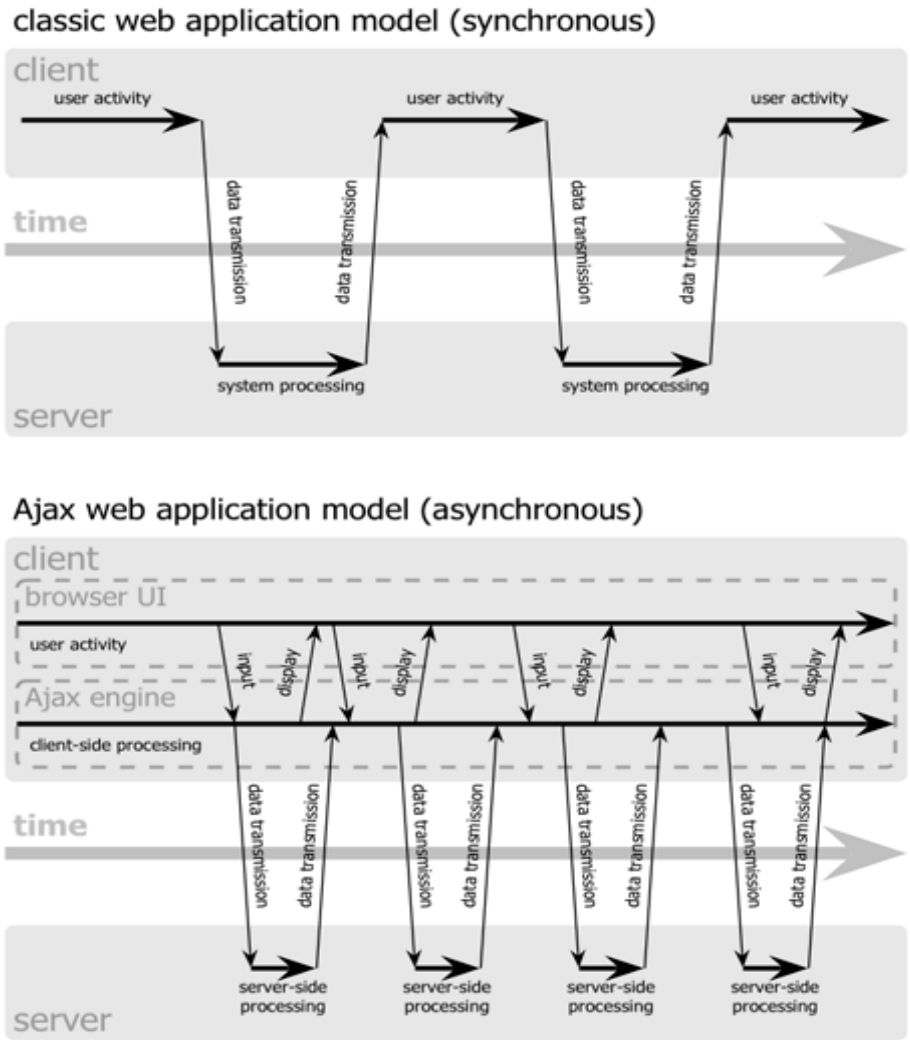
XMLHTTPREQUEST
Es una API que se encuentra implementado en el navegador que proporciona los métodos y propiedades necesarios para la comunicación con el servidor. Originalmente fue desarrollada por Microsoft como objeto ActiveX, discponible desde Internet Explorer. Esta API es utilizada por varios lenguajes de scripting, tales como JavaScript, Jscript VBScript y otros lenguajes de scripting de navegadores web. Lo fundamental de esta API es que emplea un canal de conexión independiente.

¿Cómo funciona AJAX?
Brevemente explicados estos son los pasos que se siguen en el funcionamiento de AJAX:
Inicialmente un usuario provoca un evento, luego se crea y configura un objeto XMLHttpRequest. El objeto XMLHttpRequest realiza una llamada al servidor, la petición se procesa en el servidor, el servidor retorna un documento XML que contienen el resultado, el objeto XMLHttpRequest llama a la función callback() y procesa el resultado y finalmente se actualiza el DOM de la página asociado con la petición con el resultado devuelto.

El siguiente cuadro realiza una comparación entre el modelo de aplicaciones clásico y el modelos de aplicaciones Web AJAX:



Cuadro comparativo entre el funcionamiento de las aplicaciones Web tradicionales, de manera síncrona y el funcionamiento de la tecnología AJAX, de manera síncrona:



COMPARACIÓN CON EL PARADIGMA TRADICIONAL

Ventajas de AJAX

- Mayor Interactividad
  Recuperación asíncrona de datos, reduciendo el tiempo de espera del usuario
- Facilidad de manejo del usuario
  El usuario tiene un mayor conocimiento de las aplicaciones de escritorio
- Se reduce el tamaño de la información intercambiada
- Portabilidad entre plataformas
  No requieren instalación de plugins, applets de Java, ni ningún otro elemento
- Código Público

Inconvenientes y Críticas de AJAX

- Usabilidad: Comportamiento del usuario ante la navegación.
  - Botón de volver atrás el navegador
  - Problema al agregar marcadores/favoritos en un momento determinado de la aplicación
  - Problemas al imprimir las páginas renderizadas dinámicamente

- Tiempos de Respuesta entre la petición del usuario y la respuesta del servidor
  - Empleo de feedback visual para indicar el estado de la petición al usuario.
- JavaScript
  - Requiere que los usuarios tengan el Javascript activado en el navegador
  - En el caso de Internet Explorer 6 y anteriores , que necesita tener activado el ActiveX (En Internet Explorer 7 , se implementa como JavaScript nativo).
  - Como en DHTML, debe comprobarse la compatibilidad entre navegadores y plataformas.

COMET

FUNDAMENTOS DE COMET

Actualmente, se ha acuñado un nuevo término para un aspecto variante de AJAX que esta llamando la atención más que la anterior tecnología. El termino COMET describe a las aplicaciones donde el servidor se mantiene "empujando" datos, o manteniendo un torrente continuo de datos a la aplicación cliente, en vez de tener al navegador realizando varias peticiones al sevidor para actualizar el contenido.

Esta técnica difiere fundamentalmente de AJAX, ya que:

"AJAX realiza grandes cambios principalmente en el navegador, mientras que COMET fundamentalmente cambia la naturaleza de la comunicación realizada en la arquitectura cliente – servidor."

Es en esta arquitectura en la que se difiere de la tecnología AJAX, aunque en varios sitios de discusión en Internet ven a COMET como una parte de todo lo ofrecido por AJAX, complementario a los aspectos de Interfase de Usuario de AJAX. Pero es necesario dejar en claro que COMET es el nombre de un patrón de arquitectura en la relación con AJAX

COMET se basa en que el servidor te envía datos aunque el cliente no los pida (HTTP Push), si el mismo haces una llamada, el canal se queda abierto y el servidor te va mandando información, o lo que es lo mismo, que el servidor nos va a estar devolviendo el resultado en partes. En otras palabras todas las aplicaciones de COMET utilizan conexiones HTTP de larga duración para reducir a latencia con la cuallos mensajes son pasados al servidor. En esencia no realizan pedidos ocasionalmente al servidor. En vez de eso el servidor mantiene una línea abierta de comunicación mediante la cual puede "empujar" datos hacia el cliente.

CUESTIONES FILOSÓFICAS

Para cualquier usuario, el servidor representa a otro usuario, por causa de que la red es de carácter multiusuario, simples actualización de interacción no son suficientes, usuarios que utilizan el mismo "espacio" necesitan vivir actualizados de sus propios cambios y los cambios de otros usuarios. Esta necesidad de actualización constante afecta a la disponibilidad de acciones, cuando creamos una página, o levantamos una imagen, o algo más, estamos cambiando el contexto. Los medios de conversación son definidos por la latencia, interrupciones y ancho de banda.

Los usuarios desean utilizar los medios de comunicación de alta interrupción en el menor grado posible. Las conversaciones son eventos ordenados, las interfases granulares requieren eventos granulares, las conversaciones granulares son mas inmediatos. Las aplicaciones sociales son vistas como buses de datos, Las aplicaciones web sociales simplemente amontonan los cambios necesarios hoy en día. No existen formas efectivas de suscribir los eventos al servidor. Para mejorar el contexto, necesitamos agrupar los eventos.

COMET es una técnica para empujar los datos desde el lado del servidor. En realidad es un nuevo término para una vieja tecnología. Esto se realiza mediante el establecimiento de de una conexión de larga duración en vez de utilizar varios pedidos de los datos al servidor. COMET utiliza XMLHttpRequest para la entrega de datos entre el cliente y el servidor a través del protocolo HTTP. También es conocido como Server Push o HTTP Push.

SIMILITUDES ENTRE AJAX Y COMET

Existen varias similitudes con AJAX, no existe necesidad de nuevos plugins, se utiliza http plano, asincronía, amplio soporte de navegadores.

Ejemplo de algunos sistemas que utilizan COMET:

- GMail
- GTalk
- JotLive

- Renkoo
- Meebo
- cgi:irc
- KnowNow

Debe quedar en claro que COMET no es un framework o un toolset. Sino mas bien es un concepto como lo es AJAX

DIFERENCIAS ENTRE AJAX Y COMET
A NIVEL CONCEPTUAL
En una aplicación en AJAX, el cliente maneja la interacción. El problema es que el contexto y el contenido manipulado caduca rápidamente en tiempos diferentes.
Las aplicaciones COMET evitan el retardo manteniendo una conexión HTTP y TCP/IP y una simple conexión es reutilizada, pero la mayor diferencia con AJAX es que la latencia por los pedidos realizados por el mismo, que COMET evita. La gran estrategia: transferir únicamente los datos necesarios, exactamente cuando la necesidad del mismo sea de mayor relevancia.

A NIVEL IMPLEMENTATIVO
Existen dos técnicas implementativas, la primera consiste en un pedido de larga duración donde se realiza una reconexión después de cada datagrama, esto se implementa simplemente con XMLHttpRequest.
El otro método es utilizar una técnica llamada Forever-Frame. Un forever-frame es un Iframe o navegador que recibe bloques de script y utiliza una tecnica de renderizado progresivo. Esta técnica es altamente portable y permite conexiones y subdominios. La conexión se cierra únicamente cuando existe un error en los ciclos de la conexión.

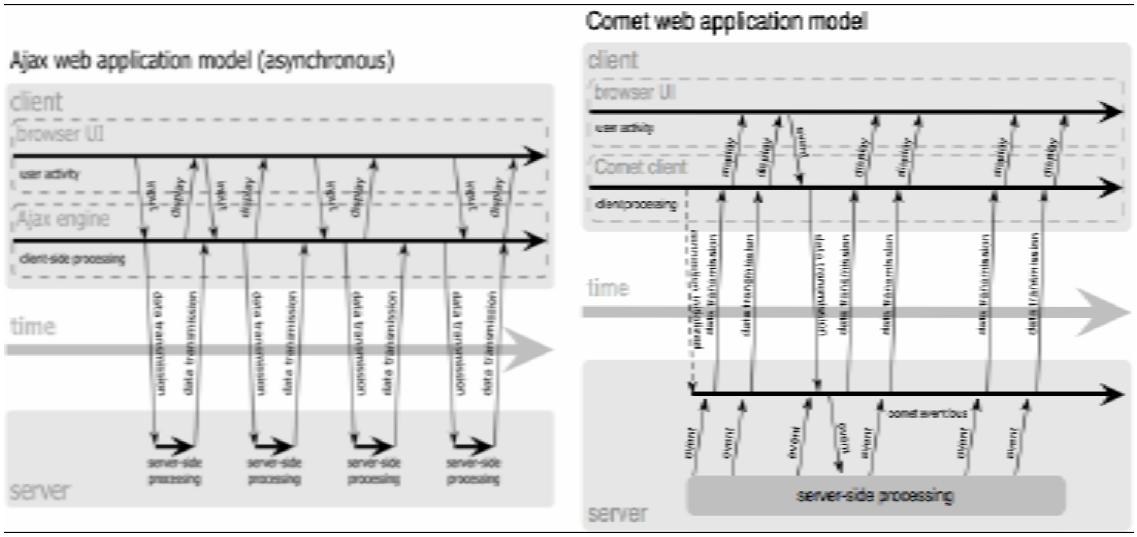
En Firefox se implementa usando *Content-type multipart/x-mixed-replace* e indicando que el canal que se abre es *multipart*, o lo que es lo mismo, que el servidor nos va a estar devolviendo el resultado en partes.
Con otros navegadores esta posibilidad no existe, y lo que se hace es abrir un canal *XMLHttpRequest*, durante cierto tiempo e ir comprobando a cada rato lo que nos llega del servidor. Para hacer esto, no es necesario AJAX, ni *XMLHttpRequest*, ni nada nuevo, simplemente un *iframe* y una función javascript que te realice llamadas cada cierto tiempo. Y ¿qué pasa si el servidor tarda más tiempo en devolvernos la salida de que tenemos la conexión abierta?, esto puede ser solucionable, pero también puede ser un problema.
¿Qué se distingue una llamada COMET a una llamada AJAX en Javascript?, tan solo en que hay que indicar que va a recibir varios objetos del servidor, para eso se usará el atributo *multipart*. Hay que indicar que es *multipart* antes de que se abra la conexión (*open*) para indicar que es multipart En el siguiente ejemplo de código Javascript se ilustra:

```
1.  function holaMundo () {
2.    var obj = document.getElementById("texto");
3.
4.    obj.innerHTML = "";
5.    comet = cometobj();
6.    comet.multipart = true;
7.    comet.open("GET", "holamundo.php", true);
8.    comet.onreadystatechange=function() {
9.  if (comet.readyState==4) {
10.   obj.innerHTML += comet.responseText;
11. }
12.   }
13.   comet.send(null);
14. }
```

SINTETIZANDO LAS DIFERENCIAS

COMET emplea una conexión persistente entre el cliente y el servidor web, por tanto la entrega de los datos actualizados se realiza desde el servidor y sin que el usuario lo haya solicitado.

El cliente no solicita los datos, pero sí envía información al servidor. El servidor no rsponde al cliente con un bloque de datos, se espera a que haya algún evento de lado del servidor para enviar la información. El siguiente cuadro demuestra las diferencias entre AJAX y COMET:



Como se ilustra arriba, las aplicaciones COMET pueden entregar los datos al cliente en cualquier timepo, no solamente en respuesta de algún input del usuario. Los datos son entregados sobre una simple conexión, previamente abierta. Este enfoque reduce la latencia de la entrega de datos significativamente.

La arquitectura se basa desde el punto de vista de los datos, el cual es orientada a eventos en ambos lados de la conexión HTTP. Aquellos que están familiarizados con los middleware que estan orientados a los mensajes pueden encontrar este diagram especialmente familiar. El unúnicao cambio sustantivo es que el punto de entrega es un navegador.

En tanto COMET es similar a AJAX en que es asíncrono , las aplicaciones que implementan el estilo COMET pueden comunicar cambios de estados pagando una pequeña latencia. Esto hace a esta tecnología adecuada para muchas clases de aplicaciones colaborativas multi – usuarios y de monitoreo que de otra forma sería muy difícil o imposible de manejar en un navegador sin la utilización de plugins.

DESVENTAJAS DE COMET

Este método podría significar un desperdicio en términos de sockets y threads, y también representaría una sobrecarga para los firewalls, los balanceadores de carga, etc…

En algunos foros de discusión se crítica esta técnica ya que al mantener abierta la conexión por un largo periodo, surgen problemas de escalabilidad como consecuencia de lo expuesto anteriormente.

Una de las cosas que hacen que las aplicaciones basadas en HTTP son que realizan pequeñas peticiones de páginas. Esto significa que se pueden manejar pedidos de un orden de magnitud o una mayor cantidad de usuarios en comparación con las técnicas de conexión de larga duración.

FRAMEWORKS ACTUALES

Existen diferentes frameworks para COMET, entre los que citamos **DojoToolkit**, **Teleport** y **Pushlets**, que implementan APIs para COMET.

CONCLUSIÓN
Es bueno COMET para los usuarios? Si todos los usuarios están tratando de hacer la misma cosa en el mismo lugar a algún pedazo de datos, entonces es lo indicado. LA actualización de los datos que se realiza mediante esta técnica es ideal, especialmente para aplicaciones de conversación. Si en el marco de la aplicación los datos pueden quedar latentes y a nadie pudiese no importarle, entonces no es necesario utilizarlo.
Por el estado actual de las redes, la tecnología COMET es apropiada y surge de vuelta como respuesta a los requerimientos actuales de las aplicaciones web.

BIBLIOGRAFIA
http://www.ajaxian.com/archives/comet-a-new-approach-to-ajax-applications
http://www.dojotoolkit.com
http://alex.dojotoolkit.com
http://phil.technometria.com
http://csshyamsundar.wordpress.com

ANEXO 1

Otras Tecnologías asíncronas para la Web

Nicolás González

ARSCIF
El **ARSCIF** (Asynchronous Remote-Script Callback Invocation Framework). Hace fácil intercambiar datos con un servidor en ECMAScript (versión estandarizada del lenguaje JavaScript, originalmente desarollada por Netscape) utilizando frames ocultos y callbacks.
Utilizar frames ocultos para comunicarse con un servidor sin recargar una pagina es conocido como remote scripting. Lo que realmente ocurre es que iniciamos asíncronamente con script remoto, luegop el script invoca un callback (posiblemente pasando datos que no estén disponibles del lado del cliente). Se utiliza *remote callback invocation* (RCI) (que no debe confundirse con *remote procedure call* ya que este denota llamadas a funciones ejecutadas remotamente de forma transparente).

Porque **ARSCIF** es mejor que otras librerías escritas para las mismas funciones

- ARSCIF puede enviar y redcibir datos arbitrariamente complejos en forma de literales ECMAScript canónicos. Uno no está limitado a tipos primitivos o arreglos.
- Soporte pleno de caracteres UTF-8 lo que permite intercambio de datos en cualquier lenguaje utilizando UNICODE, la implementación hace de este soporte dependiente únicamente del lenguaje script del servidor y así no del navegador ni en el HTTP Server.
- ARSCIF considera seriamente la concurrencia y ofrece muchas opciones para lidiar con las mismas.
- ARSCIF es sobre libre distribuido bajo la licencia de X11.

Fuente:
http://arscif.dsi.unimi.it

Java2EE
Service Activator
Provee Procesamiento asíncrono para cualquier Enterprise Beans.
En la especificación 2.0 de EJB (EJB es una de las API que forman parte del estandar de construcción de aplicaciones empresariales J2EE de Sun Microsystems), el bean manejado por mensajes es de una sesión sin estado. Utilizando el patrón Service Activator, es posible proveer aplicaciones asíncronas en todos los tipos de beans empresariales incluyendo los de sesión sin estado, sesiones de estado y los de entidad, en pocas palabras sobre cualquier servicio de negocio. Por lo tanto provee una manera de habilitar procesamiento asíncrono para clientes que o no tienen necesidad de esperar para resultados, o no quieren esperar la finalización del procesamiento. El procesamiento puede ser diferido y realizado más tarde permitiendo al cliente completar el servicio en menor timepo.

Fuente:
http://java.sun.com/blueprints/corej2eepatterns/Patterns/ServiceActivator.html