Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

UTILIZACIÓN DEL LENGUAJE CURL PARA LA IMPLENTACIÓN DE PÁGINAS *WEB*

Fernando Enrique Cáceres Monterroso

Asesorado por: Ing. Manuel López Fernández

Guatemala, mayo de 2004

FACULTAD DE INGENIERÍA

UTILIZACIÓN DEL LENGUAJE CURL PARA LA IMPLEMENTACIÓN DE PÁGINAS WEB

TRABAJO DE GRADUACIÓN

PRESENTADO A JUNTA DIRECTIVA DE LA FACULTAD DE INGENIERIA POR

FERNANDO ENRIQUE CÁCERES MONTERROSO

ASESORADO POR: ING. MANUEL LÓPEZ FERNÁNDEZ

AL CONFERÍRSELE EL TÍTULO DE INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, MAYO DE 2004

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Sydney Alexander Samuels Milson
VOCAL I	Ing. Murphy Olympo Paiz Recinos
VOCAL II	Lic. Amahán Sánchez Alvarez
VOCAL III	Ing. Julio David García Celada
VOCAL IV	Br. Kenneth Issur Estrada Ruiz
VOCAL V	Br. Elisa Yazminda Vides Leiva
SECRETARIO	Ing. Pedro Antonio Aguilar Polanco

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Sydney Alexander Samuels Milson
EXAMINADOR	Ing. Jorge Luis Álvarez
EXAMINADORA	Inga. Elizabeth Dominguez
EXAMINADOR	Ing. Ricardo Morales Prado
SECRETARIO	Ing. Pedro Antonio Aguilar Polanco

HONORABLE TRIBUNAL EXAMINADOR

Cumpliendo con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

UTILIZACIÓN DEL LENGUAJE CURL PARA LA IMPLEMENTACIÓN DE PÁGINAS WEB

Tema que me fuera asignado por la Coordinación de la Carrera de Ingeniería en Ciencias y Sistemas con fecha 30 de enero de 2002.

Fernando Enrique Cáceres Monterroso

ACTO QUE DEDICO A:

- Mi Señor Jesús.
- Mis padres, Jaime Enrique Cáceres Díaz y Maria Eugenia Monterroso de Cáceres.
- Mi hermano, Jaime Eduardo.
- Mis tíos, tías, primos, primas, abuelas y sobrinas.

AGRADECIMIENTOS

- A la razón de mi existencia, mi gran Señor Jesús, por darme las fuerzas y el amor para completar uno de mis tantos sueños.

- A mis padres, por el amor, apoyo y ejemplo que han dado a mi vida.

- A Jaime, por el apoyo y ejemplo que tengo de él.

- A Vlady y Mónica, por su incondicional amistad, aprecio y guianza.

- A mis amigos, Alberto, Carlos, Edson, Guillermo, Herbert.

- A mis amigas especiales, Arely, Brenda y Emy.

- A los miembros de Visión de Fé.

- A mis amigos de carrera Manuel, Mefiboset, Juan Manuel, Juan Carlos, Elvis y Melvin.

- Al Ing. Manuel López, por su excelente asesoría en la realización de este trabajo.

- A la Universidad de San Carlos y a todos los catedráticos por su excelente aporte a mi carrera.

- A todos mis compañeros de sistemas.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

TABLAS

GLOSARIO

Ancho de banda Es la capacidad de transmisión de una computadora o un canal de comunicación.

Applet Se le llama así a una aplicación que se está ejecutando dentro de otra, este término es utilizado para aplicaciones de red.

ASP *Active Server Page*, página activa de servidor, páginas *web* dinámicas usadas por Microsoft.

Boleano Tipo primitivo de datos utilizado para representar verdadero o falso.

Estación de trabajo Esta es una computadora que se encuentra conectada a una red.

HTML *Hyper Text Markup Language,* lenguaje de enmarcado de texto. Este lenguaje define el posicionamiento de elementos en una página como fuentes, gráficas, texto, vínculos a otros sitios.

Java Un lenguaje de programación que fue creado por *Sun MicroSystems* para poder desarrollar aplicaciones distribuidas para ser utilizadas en un navegador de Internet.

JSP *Java Server Page*, página de servidor de java,

lenguaje de programación de páginas dinámicas.

Kbps — *Kilobits per second*, es el número de bits o dígitos binarios que se transmiten cada segundo. Es utilizado como un indicador en comunicaciones para medir rangos de transmisión

Navegador — Un programa de aplicación que es utilizado para explorar recursos de internet.

ODBC — *Open DataBase Connectivity*, aplicación que permite a un programa a muchos tipos de base de datos y formatos de texto.

Parser — Programa que es utilizado para el análisis sintáctico y semántico de una gramática

PHP — *Persona Home Page*, lenguaje de programación de páginas *web* dinámicas.

Pixel — Estrucutura gráfica utilizada para representar un punto en un dibujo.

Plug-in — Un pequeño programa que se puede instalar al navegador de Internet para agregar una capacidad que originalmente no está presente. Estos por lo general son gratis

Puerto — Es número utilizado para identificar una conexión a un punto específico de una red

Repositorio	Lugar de almacenamiento de información útil para el *applet* de Curl.
SAX	*Simple Api for XML*, programa utilizado para reconocer el lenguaje XML.
Script	Un pequeño programa que se ejecuta en un tiempo en particular.
Servidor	Cualquier computadora que habilita el acceso a los archivos, impresoras, comunicaciones y otros servicios a usuarios de la red
SOAP	*Simple Object Access Protocol,* Protocolo utilizado para el acceso a servicios web, utiliza el XML para el intercambio de información
Sockets	Mecanismo de comunicación entre procesos.
Tabla *hash*	Método de representar datos de tal manera que se puedan encontrar más rápido. En esta estructura se le asigna un índice a cada pedazo de datos
TCP	*Transmisión Control Protocol,* protocolo que se utiliza en la capa de transporte del modelo OSI

UDP	*User Datagram Protocol*. Un protocolo sin conexión de la capa de transporte que es utilizado en el TCP/IP
Unicode	Código de 2 bytes para representar la mayoría de los lenguajes escritos del mundo.
URI	Identificador único de recurso.
URL	Dirección para un recurso en internet.
Web	Conexión inmensa de computadoras conectadas.
WSDL	*Web Service Description Language*, lenguaje extensible para la descripción de servicios *web.*
WWW	*World Wide Web*, es una armazón arquitectónico para accedes a documentos vinculados en miles de máquinas de toda la Internet. Posee una interfaz gráfica muy amigable.
XML	*Extensible Markup Language,* tecnología que permite que una página HTML, describa la información en término de lo que representa

RESUMEN

En la actualidad se necesita una tecnología capaz de agilizar las operaciones de las empresas con respecto a las aplicaciones. Actualmente las tecnologías existentes cumplen con su tarea pero con un alto costo para las empresas y para los usuarios finales de la aplicación. Con cada paso en que ha evolucionado el Internet y la forma en que se muestran las páginas se agrega un nuevo problema, ya sea para el cliente o para las empresas.

Los lenguajes que se han utilizado para realizar aplicaciones han sido muy útiles pero con deficiencias claras que dejan entredicho su utilidad. En este trabajo se compara el lenguaje Curl con los lenguajes que actualmente se utilizan y muestra sus ventajas sobre éstos. Se muestran tablas comparativas entre los costos del lenguaje Curl y lo que actualmente se está utilizando, páginas dinámicas hechas en el servidor.

Se tocan puntos importantes como la sintaxis del lenguaje, propiedades importantes del mismo y ejemplos útiles para que el lector tenga una introducción al lenguaje e investigue más. Se recomienda al lector que tome esta investigación como una introducción al lenguaje y no como un manual del usuario.

OBJETIVOS

General

Mostrar las ventajas del nuevo lenguaje Curl para la realización de páginas *web*

Específicos

1. Mostrar la necesidad de una nueva tecnología para las demandas de los usuarios

2. Ejemplificar los beneficios tecnológicos y económicos que tiene Curl.

3. Realizar un ejemplo básico mostrando las capacidades del lenguaje

INTRODUCCIÓN

Este trabajo muestra las ventajas de esta nueva tecnología que tendrá un efecto drástico en la navegación en el internet. Se mostrarán las ventajas de utilizar el lenguaje Curl, el cual es ideal para conexiones con una deficiencia en ancho de banda y velocidad. Con esta nueva tecnología se podrán realizar páginas *web* funcionales del lado del cliente en las cuales se realizan las operaciones las cuales no aumentan los costos que las empresas incurren en los servidores de páginas.

En esta investigación se realizan ejemplos básicos de lo que el lenguaje puede hacer, de las opciones que un desarrollador debería tomar en cuenta para utilizar este lenguaje.

El lenguaje es muy similar con los lenguajes C++ y HTML, esto facilita su aprendizaje y le da al usuario un sentido de familiaridad con el mismo.

1. NECESIDAD DE UNA NUEVA TECNOLOGÍA

1.1 Primera generación: HTML

La primera generación de HTML fue creada esencialmente para poder compartir los archivos y poder verlos desde un lugar remoto. El formato de los primeros archivos fue hecho en HTML para poder compartirlos. Aquí el hecho de compartir los archivos era suficiente mientras los desarrolladores y diseñadores aprendían este nuevo lenguaje. Después que los creadores de sitios comenzaron a realizar páginas *web* que tenían datos estáticos, la arquitectura que usaban empezó a mostrar sus limitaciones, ya que las necesidades aumentaban.

Como un ejemplo de esas limitaciones, supóngase una página que muestra el pronóstico del tiempo, los datos para esta página varían mucho, incluso en menos de un día, entonces el diseñador de la página tenía que volver a editarla para que la información estuviera actualizada. El solo hecho de actualizar los datos de la página todos los días consume mucho el tiempo del diseñador. Otro problema serio que surgió fue que los datos perdieron su significado, el HTML sólo se encarga de mostrar los datos y por esta misma razón no se puede diferenciar entre la temperatura y velocidad del viento, y mucho menos el nombre de la ciudad, el significado de los datos se perdió.

Otro ejemplo es, si el usuario necesitaba ver el pronóstico del tiempo ordenado por nombre de ciudad o por temperatura, se tenía que realizar una página diferente para cada vista; solamente este tipo de modificaciones a la página es un gran consumo de tiempo para el diseñador. No se diga si se deseaba realizar una página que mostrara la información de todo el país. Esta tecnología consumía mucho tiempo, no era muy funcional y no tenía la escalabilidad deseada para las necesidades crecientes de las empresas. Una nueva tecnología se necesitaba para poder llenar las necesidades crecientes de los medios.

1.2 Segunda generación, *server pages*

Afortunadamente, una nueva forma de hacer páginas había sido creada. Esta forma permitiría que los datos y el HTML se mezclaran cuando la página fuera solicitada, convirtiéndola en más dinámica. Lo que trajo el beneficio de guardar la información del pronóstico del tiempo en una base de datos y mostrarlos en una plantilla de una página de HTML cuando fueran solicitados. Ahora, la preocupación principal del diseñador era crear una plantilla lo suficientemente inteligente para poder interactuar con el usuario cuando éste necesitara la información ordenada de otro modo. Otra cosa que tenía que realizar el diseñador es poblar la base de datos de información para que los cambios se vieran reflejados en la página.

El cambio de presentación de la información tuvo un gran impacto en el lenguaje HTML. Este nuevo método se llama s*erver pages*, los servidores *web* se encargaban de crear las páginas en el momento que éstas eran solicitadas y, el diseñador ya no debía realizar todas las vistas que se necesitaban, lo cual generó una reducción en los costos por contratación de diseñadores. Asimismo, este método trajo un gran número de aplicaciones en el *web* las cuales podrían interactuar con bases de datos y con HTML. Al mismo tiempo eliminó las deficiencias de las páginas estáticas, permitió personalizar las páginas e interactuar con ellas de una forma en la cual no era posible con las páginas estáticas.

El *server pages* permitió a las empresas poder realizar aplicaciones para procesos internos sin la necesidad de que los usuarios se encontraran físicamente dentro de las instalaciones.

1.3 La *web* de hoy

En la *web* de hoy en día la página dinámica se envía al servidor, éste la procesa y manda una respuesta al cliente. Un ejemplo que se puede observar diariamente es cuando se elimina un correo de la cuenta o se ingresa información en una página. El proceso tarda unos cuantos segundos para poder mostrar la información solicitada por el usuario. Con este nuevo cambio los usuarios pueden realizar operaciones más complejas desde la *web* . Pero no todo en este tipo de páginas tiene ventajas, el pequeño retardo que se observa cuando se solicita una página es por dos razones importantes. La primera es el tiempo para poder mandar la página al servidor y luego bajar la página solicitada. La segunda es que el servidor tiene que ensamblar la página al momento de la solicitud.

La segunda razón del retardo es una de las más costosas, porque actualmente los servidores se encuentran saturados de peticiones las cuales toman tiempo y recursos. Los servidores que manejan páginas estáticas responden mejor que los servidores de páginas dinámicas, esto es por el procesamiento de la página. Existe otro problema con la información guardada en la base de datos, puede que la información esté desactualizada o fuera de fecha. Con el ejemplo de los pronósticos del tiempo, las personas que tienen el dato correcto del pronóstico son los meteorólogos, esa información cambia constantemente, y se puede dar el problema que la información guardada esté incorrecta.

Todos estos errores continúan hasta hoy en día. Las páginas son lentas en mostrarse, pueden contener muchos errores por la validez de la información y estos problemas no tienen una buena aceptación por parte del usuario. Los mismos se pueden eliminar cuando la tecnología evoluciona o mejora, pero este fenómeno toma tiempo en darse y mientras esto sucede la mejor solución para estos problemas es aumentar el número de servidores para las páginas y el ancho de banda para mejorar el desempeño del sitio.

Cuando se implanta este tipo de tecnología el costo es muy elevado y no todas las empresas tienen la capacidad de poder implantar este tipo de soluciones. Para poseer la información actualizada se necesita una gran inversión. Sin embargo hay una solución no muy lejana para estos problemas: los servicios *web*.

1.4 La próxima generación, servicios *web*

La idea de los servicios *web* es muy reciente. Las compañías y los sitios *web* los han estado utilizando sin darle su nombre oficial. La idea del servicio *web* es que un proceso esté escuchando y manejando las peticiones de cierta computadora. Un servicio *web* puede realizar varias tareas como por ejemplo obtener la información de una pieza en el inventario, presentar la información de la cartelera del cine y mostrar la información actualizada del pronóstico del tiempo. La idea es, pasar la responsabilidad de manejar, mostrar y guardar la información a las empresas o sitios que manejan la información, es decir cuando se esté en una página de pronóstico del tiempo, la información esté siendo mostrada directamente por los meteorólogos.

Muchas organizaciones han tenido en mente descentralizar la información y dejar que las computadoras hablen entre sí. Sin embargo, el concepto de un servicio *web* no se popularizó por la razón de que cada uno de los sitios tienen formas diferentes de trasmitir la información, lo cual representa una dificultad para el manejo de la misma. Los servicios *web* traen un nuevo estándar para poder compartir la información. WSDL, SOAP y XML son unos de los estándares que se están utilizando para poder realizar esta tarea de compartir los datos de diferentes sistemas en diferentes organizaciones. Lo que quiere decir que no importa en que lenguaje estén los datos, siempre se asegura que éstos se podrán acceder fácilmente, lo cual da la oportunidad de poder obtener el pronóstico del tiempo de varios centros de meteorología sin importar como se encuentre la información.

Una clase de estandarización de este tipo es la necesaria para poder compartir la información de diferentes compañías en diferentes partes del mundo.

Continuando con el ejemplo del pronóstico del tiempo, cada vez que se solicita la página del pronóstico, el servidor *web* crea la página con la información del servicio *web*, en lugar de estar consultando a la base de datos que puede que tenga la información desactualizada. Aquí hay una serie de pasos para demostrar este proceso, el usuario solicita la página de los pronósticos del tiempo, el servidor recibe la petición, el servidor utiliza el servicio *web* para que los meteorólogos suministren la información correcta, el servidor recibe la información de parte de los meteorólogos, el servidor construye la página y manda la página de respuesta al usuario.

Con este nuevo enfoque de los servicios *web* los diseñadores se dedican a lo que realmente es su especialidad, diseñar. Dejan los problemas de cómo presentar y almacenar los datos a las personas que realmente poseen la información, dividiendo de esta manera la tarea de la presentación de los datos. El diseño será solamente darle un espacio a los servicios *web* para que funcionen.

Al momento de utilizar esta nueva tecnología se reduce la carga de los servidores, pero el cliente sigue en la misma situación que antes, el tiempo de espera de solicitud de una página es mayor por la razón de que los datos no se encuentran a la mano.

1.5 Una solución, servicios *web* del lado del cliente

Una solución para el problema del cliente en espera de la respuesta del servidor es: se crean los servicios *web* que interactúan con el cliente directamente. El cliente ya no recibe HTML, recibe los datos en sí, o sea, que se podrá diferenciar entre la humedad relativa y la temperatura promedio. Ahora al momento de realizar servicios *web* del lado del cliente se pasa todo el procesamiento de un servidor *web* al cliente. Así que todas las operaciones que se realizaban en el servidor *web* ahora se realizan del lado del cliente pero con mayor rapidez.

La tecnología que provee la facilidad de dar un formato como el HTML y el poder de servicio de C++ y *java* es el lenguaje Curl.

El lenguaje Curl permite a los diseñadores crear páginas *web* con la facilidad del formato HTML utilizando los avances de los servicios *web*. Los *applets* de Curl son aplicaciones funcionales del lado del cliente las cuales una vez cargadas piden la información a los servicios *web*. En este lenguaje se integra la velocidad de una aplicación de lado del cliente, los beneficios de una página *web* y el poder de procesamiento de información de un servicio *web*.

1.6 Curl y otros lenguajes

Para poder comparar a Curl con otros lenguajes para aplicaciones distribuidas en un ambiente *web* se debe tomar en cuenta cuál es la aplicación del lenguaje. Primeramente se compara a Curl con los lenguajes que se ejecutan del lado del cliente, *javascript* y *vbscript*, también con las presentaciones de las páginas que han sido tradicionalmente trabajadas con HTML. Después, el lenguaje se compara con C++ y *java*, los lenguajes que dieron la idea de esta programación orientada a objetos. Para terminar se compara a Curl con los lenguajes que corren en los servidores, como ASP, PHP, JSP.

1.6.1 Comparación con HTML/CSS

Como se dijo anteriormente al inicio del capítulo, HTML fue la base para el intercambio y presentación de datos sobre la Internet, fue la base del WWW. En las primeras versiones del lenguaje, poseía limitaciones para dar formato al texto, presentación y controles de aplicación. Todas estas limitaciones fueron expuestas en el consorcio de *world wide web*, después de esto se estandarizó el uso de HTML. Cuando surgió la versión 4 del lenguaje HTML, paralelamente salió la hoja de estilos en cascada (CSS), con este nuevo avance se mejoró la presentación de las páginas y se le dio al usuario el poder de crear plantillas para sus páginas y agilizar el diseño de la presentación de las mismas.

Con el surgimiento de nuevos navegadores y plataformas, muchas de las mejoras de HTML no fueron implementadas y usadas por la incompatibilidad de las plataformas. Los desarrolladores de páginas *web*, observarán que el lenguaje Curl tiene muchas similitudes con el HTML. Se utilizan etiquetas para demarcar texto en cierta forma y color, se pueden crear tablas, colocar controles de aplicación e insertar imágenes dentro de la página. Las diferencias de lenguaje son fácilmente asimiladas. Cada etiqueta de HTML tiene su similar dentro de Curl. A diferencia de HTML existe solamente un intérprete del lenguaje, el *plug-in* de Curl, lo cual da una gran ventaja sobre HTML por la razón de que no se necesitará conocer las otras versiones del lenguaje o las incompatibilidades de los navegadores, lo que hace que el mantenimiento de las páginas sea más fácil.

Por las razones expuestas anteriormente, el lenguaje muestra gran ventaja sobre el HTML, sin mencionar el uso de un lenguaje orientado a objetos.

1.6.2 Comparación con lenguajes del lado del cliente (script)

El momento en que surgió la primera generación de páginas de HTML fue una verdadera revolución, aunque solamente se tenía la capacidad de visualizar páginas estáticas que no ofrecían iteración en lo absoluto y los desarrolladores necesitaban algo más que solamente presentación.

El lenguaje de lado del cliente, *javascript* fue introducido en 1995 por *Netscape* y *Sun MicroSystems* como una solución al problema de las páginas estáticas. Este lenguaje liviano tuvo gran popularidad por las siguientes razones

- *javascript* no requiere una precompilación
- Ofrece un modelo de objeto de documento (DOM) el cual permite la manipulación de HTML en la página.
- Permite a los desarrolladores crear procedimientos con lógica que corran del lado del cliente.

Con esta nueva ventaja los desarrolladores pueden realizar validaciones del lado del cliente y dan al usuario una sensación de iteración con la página.

Con el lanzamiento de *javascript*, *Microsoft* ofreció su propia versión de *script* del lado del cliente, *vbscript* para algunos desarrolladores las ventajas que mostraba el lenguaje daban varias razones de sacrificar la compatibilidad. Otra de las ventajas que mostraba el *vbscript* era que podía utilizar poderosos controles de *windows*, la sintaxis parecida a *visual basic*, entre otras, estas fueron las razones que se tomaron en cuenta para colocar al *vbscript* como lenguaje *script* del lado del cliente. *Microsoft* desarrolló una copia del *javascript*, el *jscript*.

El nuevo lenguaje *vbscript* no fue tomado como un estándar de la industria, en su lugar fueron tomados el *javascript*/*jscript*. *Microsoft* decidió tomar el *vbscript* para sus páginas activas ASP y para que sus servidores aprovecharan al máximo las opciones que ya se encontraban implantadas en *windows*.

Cuando fue implantado el *javascript*, tuvo la capacidad de interactuar con todo el lenguaje de HTML. Un nuevo problema se presentó para los desarrolladores, debían aprender una nueva sintaxis de programación, esto sin mencionar que las dos empresas más grandes de navegadores no establecían un estándar de *javascript.* Nuevamente fue introducido el problema de HTML de falta de compatibilidad en los navegadores.

1.6.3 Comparación con *java*/C++

Cuando se mezcla un lenguaje que da las opciones de presentación y los beneficios del *script*, se pueden crear aplicaciones distribuidas básicas. Pero, para poder realizar aplicaciones que necesitan un grado más complejo de operaciones como protocolos de red y presentaciones avanzadas de datos, se necesita un lenguaje orientado a objetos. Existe una gran cantidad de lenguajes, cada uno con ventajas y desventajas.

El C++ fue creado en 1980, derivado de C, lo cual le daba al usuario la ventaja de programar con un lenguaje orientado a objetos. El usuario puede utilizar las ventajas de la programación orientada a objetos, herencia, polimorfismo y la reutilización de código, pero después de todo el C tiene un problema, que tiene un alto grado de complicación y complejidad. Existen en la actualidad muy pocas aplicaciones que utilizan todas las ventajas del C.

Java surgió como una alternativa para el lenguaje C, siempre con las grandes ventajas de la programación en objetos y como una solución más sencilla que C. Otra de las razones de la popularidad de *java* es que se puede ejecutar desde cualquier plataforma. Con *java* se eliminaron muchos de los vestigios del C y se agregaron muchas ventajas nuevas al lenguaje. El lenguaje proporcionaba al usuario un ciclo de desarrollo más corto. Algo que fue verdaderamente un éxito fue su sistema de recolección de basura, el cual le daba al usuario una ventaja en el manejo de la memoria, por la razón de que las aplicaciones basadas en C++ sufrían del problema de un mal manejo de la memoria.

Otra mejora que fue introducida en *java* fue la generación de código binario intermedio. Este código intermedio puede ser compilado en cualquier plataforma con el compilador de *java*. Como *java* originalmente fue orientado a la programación *web*, muchos de los navegadores poseían una versión diferente del JVM (*Java Virtual Machine*). Entre los navegadores se encuentra el Internet Explorer. Muchos de lo navegadores no actualizaban su versión de JVM. Luego fueron publicados artículos sobre los privilegios de seguridad que los *applets* de *java* tenían del lado del cliente. Con estos problemas de seguridad el lenguaje se dirigió principalmente a los servidores *web*, en donde realizaban una tarea muy buena.

El lenguaje Curl fue diseñado para tomar las ventajas de la programación en objetos de C++, *java* y otros lenguajes y al mismo tiempo darle al usuario una herramienta para la presentación de páginas y el lenguaje *script*. Básicamente Curl es la unión de las mejores ventajas de *java* con las mejores ventajas de C++.

1.6.4 Comparación con los lenguajes del Servidor

Como se ha presentado hasta ahora, el lenguaje Curl fue diseñado para que se ejecute del lado del cliente. Una implementación de lenguaje de servidor de Curl no existe por el momento. En un futuro se tiene planeado colocar un *plug-in* para servidores. Con esto el lenguaje Curl se tiene que integrar sin ningún problema con los lenguajes de servidores, JSP, ASP, PHP, PERL y otros medios de comunicación como XML.

Actualmente la única forma de comunicar a un servidor *web* con un cliente es por medio de *sockets* y el protocolo http. Cuando el usuario hace clic a un vinculo se realiza una conexión entre el cliente y el servidor *web* el cual manda todos los datos de la página.

Así como el HTML se puede generar dinámicamente también se puede generar una página en Curl, el método es el mismo. A continuación un ejemplo de cómo generar una página dinámica desde ASP:

```
{ curl 1.7 applet }
{ applet license = “development”}
Este es un applet de Curl generado desde:
<%    response.write(“ASP”)  %>
```

Pero existe una gran diferencia entre las páginas *web* HTML y las páginas *web* en Curl, las páginas Curl son aplicaciones en sí, porque pueden pedir más información aun después de haber sido cargadas. Las páginas Curl pueden cambiar su estado sin necesidad de recargase. Cuando el usuario hace clic en un vínculo el proceso que se realiza a continuación es que el navegador procesa la operación, y luego recarga la página, pero cuando se

le da clic a un vínculo en un *applet* en Curl, el mismo *applet* se encarga de realizar la operación de presentación de la página.

Esto le da la oportunidad al desarrollador de crear un *applet* que se comunique con varios servidores *web*. Con esta ventaja la iteración entre las páginas HTML y el servidor puede ser eliminada y el servidor utilizado en otras operaciones.

2. BENEFICIOS TECNOLÓGICOS DE CURL

2.1 Menos y más pequeños archivos

La tecnología Curl envía pequeños paquetes de información que son procesados y compilados del lado del cliente. Así que mucho de la carga de procesamiento se pasa al cliente y el volumen de la información que se manda al cliente se reduce. Esto lleva a una página que se muestra con mayor rapidez a un costo mínimo. A continuación se mostrará un ejemplo de una página de entretenimiento de una empresa líder. Esta página genera más de dos millones de solicitudes de la misma al día, el tamaño de la imagen de fondo es de 60 KB, las respuestas de solicitud a esta página son demasiado lentas especialmente en una línea telefónica de 56kbps.

Sin embargo, una página con la misma funcionalidad y con gráficas 3-D, se puede diseñar en un tamaño menor, lo cual da la ventaja de bajar la página a una velocidad mayor de la normal. Por el hecho de que las gráficas se mezclan del lado del cliente, da una página del tamaño de 2.5 KB, esto es una reducción del 95%, sin agregarle ningún costo extra.

A continuación se muestra la tabla comparativa de los tiempos de bajada de una página con la tecnología actual y otra con la misma funcionalidad pero en Curl.

Tabla I. Comparación de tiempo entre Curl y la tecnología actual

	Tecnología actual	Tecnología Curl	Cambio
Comparación de eficiencia			
Bajado tamaño / sesión	60	2.5	-58
Comunicación cargos / sesión			
Cuota de *hosting*	$0.001	$0.000	
Cargo por el servicio Curl	---	$0.003	
TOTAL	$0.001	$0.003	$0.001
Bajado tiempo / sesión	10	0.4	-10

Fuente: http://www.curl.com

2.2 Aprovechando el ancho de banda

Cuando se incrementa el ancho de banda existente, la tecnología Curl permite a las compañías ahorrar el costo de tener que invertir en su infraestructura de la red. Por ejemplo, un banco envía su información a los clientes por miles de ramas para poder ejecutar las transacciones básicas del banco (tarjeta de crédito, ahorros, préstamos, etc.) por medio del los servidores *web*. Toda la información se manda a un cliente que puede trabajar con páginas HTML, también el cliente utiliza páginas con lenguaje de *java* para validaciones.

El problema es que cada rama es de 128 kbps, cuando varias ramas de la red están trabajando con el cliente al mismo tiempo, esto reduce el desempeño de la red, el tiempo de respuesta y agrega un costo de espera para los clientes finales. Utilizar líneas dedicadas de 256 kbps es muy costoso. Esta inversión sería millonaria, asumiendo que cada costo de línea es de $100.

La tecnología Curl permite a los clientes internos del banco, trabajar con un conjunto de formas las cuales tienen un menor tamaño que las páginas normales en HTML. Reproduciendo las mismas formas que el banco utiliza con algunas opciones extras que el usuario necesita, cada forma tiene un tamaño de 8 kb, el costo de esta alternativa es 75% menor que la otra opción, a continuación se muestran los costos de este ejemplo.

Tabla II. Costos de la propuesta de actualización

Solución propuesta: actualización de 128kbps a 256 kbps	
Número de ramas	6000
Incremento anual costo / rama	$1200
Costo anual total	$7,200,000
Alternativa: desarrollar las formas en Curl	
Número de ramas	6000
Estaciones por rama	6
Total de estaciones	36,000
Formas / Estación / Día	20
Tamaño de la forma	10 kbps
Secciones / Estación / Mes	20
Contenido Curl / mes (kb)	144,000,000
Contenido Curl / año (kb)	1,728,000,000
Costo anual total*	$1,720,000
Total de ahorros usando la tecnología Curl	$5,472,000
* Cuota de contenido Curl de $.001/kb	

Fuente: http://www.curl.com

2.3 Reduciendo la carga de los servidores

Cuando se implementa una solución del lado del servidor de Curl, el resultado es un código con un mantenimiento más simple, no se necesita código de presentación del lado del servidor por el motivo de que todo el procesamiento se pasa al lado del cliente. Entonces el código del servidor ya no se enfoca en detalles de presentación de datos, sino en los detalles de la lógica del negocio, interacción con la base de datos y la comunicación con las terceras partes.

La lógica del negocio es también colocada del lado del cliente para que el servidor no tenga mucha carga al momento de atender las solicitudes. Los recursos del servidor entonces pueden ser utilizados para otras operaciones; o en un futuro no muy lejano éstos puedan ser eliminados según la tecnología crezca. Cuando los servidores tienen un tiempo de respuesta bajo la productividad de la empresa baja por esta deficiencia de velocidad, lo cual hace que la empresa no tenga competitividad por los altos costos de mantenimiento de los servidores.

La solución que se pensó es hacer que la interfaz gráfica del usuario sea más rica, tenga capacidades de ordenación sin necesidad de recargar la página y capacidades gráficas extraordinarias. Toda esta inversión que se ahorra en los servidores bien se puede utilizar en otras áreas de la empresa que necesiten capital. A continuación se muestra una tabla de los ahorros en los servidores cuando se implementa una solución de Curl en la empresa.

Tabla III. Reducción de los servidores.

Reducción hipotética del número de servidores	5
Costo anual de renta de un servidor	$7,200
Costo administrativo anual mano de obra seguridad almacenamiento	$15,000
Total de ahorros anuales	$111,000

Fuente: http://www.curl.com

2.4 Facilidad de mantenimiento

En este momento hay una gran presión para las corporaciones (indirectamente para los desarrolladores de *software*) para modificar las aplicaciones cliente-servidor por los altos costos de mantenimiento, sin embargo el hecho de actualizar la versión de un programa puede ser muy costoso. Un usuario de Curl realizó una prueba con las 500 compañías de Fortune, estas compañías utilizan aplicaciones cliente-servidor, una aplicación a un usuario puede costar varios cientos de dólares. Cuando se implemente la solución en Curl, las empresas podrían tener su aplicación cliente-servidor en Internet y reducir los costos de mantenimiento hasta en un 90% dando la misma funcionalidad.

Tabla IV. Aplicación cliente-servidor en Curl

Aplicación cliente-servidor	
Número de usuarios	50,000
Costo de mantenimiento anual por usuario	$200
Costos anuales totales	$10,000,00
Aplicación cliente-servidor en Curl	
Tamaño de la aplicación(kb)	100
Número de usuarios	50,000
Sesiones / usuario / mes	20
Contenido Curl / mes(kb)	100,000,000
Contenido Curl / año(kb)	1,200,000,000
Costo anual total*	$1,200,000
Ahorros anuales totales	
utilizando la tecnología Curl	$8,800,000
* Cuota de contenido Curl de $.001/kb	
Fuente: http://www.curl.com	

2.5 Aumentando la productividad del desarrollador

Cuando se implementa la tecnología Curl, a los desarrolladores se les facilita el manejo de su personal, como el lenguaje Curl tiene capacidades de enmarcado de lenguaje, soporte de *scripts* y las ventajas de programación orientada a objetos, da como resultado el desarrollo de aplicaciones sólidas y la implementación de técnicas eficientes por parte del desarrollador. Para el mantenimiento de las aplicaciones se necesitará menos gente y menos tiempo.

Tabla V. Comparación de costo y tiempo para los desarrolladores.

	Java	**Curl**
Tiempo para desarrollar (horas / hombre)	4,800	600
Costo de desarrollo por hora	$250	$250
Costo total	$1,200,000	$150,000
Ahorros	88%	
Ahorros	$1,050,000	

Fuente: http://www.curl.com

3. PÁGINAS *WEB* UTILIZANDO CURL

3.1 Introducción al lenguaje Curl

3.1.1 Estructura básica del *applet* Curl

Al escribir una página de HTML tiene la siguiente estructura básica:

<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY>
</BODY>
</HTML>

En el cual se especifica el inicio, el título, el cuerpo y el fin de la página. Pero a diferencia de Curl con la siguiente sintaxis que delimita el archivo:

{curl 1.7 applet}
{applet license = “development”}

En este encabezado del archivo se especifica la versión del la API que se utiliza y de que tipo es el archivo de Curl, puede ser un paquete, applet o *script*. Cuando la primera línea se especifica se importan todas las librerías del paquete. En la segunda línea del encabezado se especifica el número de licencia que se tienen para poder hacer *applets* de Curl, para poder obtener una licencia de desarrollador se debe ir al sitio oficial de Curl para más detalles (http://www.curl.com).

3.1.2 Estilo del *applet*

Cuando se han especificado las opciones de la página y su tipo, se colocan las propiedades las cuales estarán vigentes en todo el documento. Cuando se desea definir el estilo del documento hay tres tipos diferentes

- *DefaultDocument* – este es el estilo predeterminado, cuando el estilo del documento no se especifica tiene este estilo. En este tipo de estilo, el texto se escribe de izquierda a derecha y de arriba hacia abajo.

- *TocDocument* – este tipo de estilo contiene una tabla de contenido al lado izquierdo en el cual se encuentran los encabezados del contenido en la página derecha.

- *PlainDocument* – cuando se selecciona este tipo de estilo el desarrollador tiene más control sobre la presentación del documento, este tipo de documento no tiene una barra de deslizamiento al lado derecho como es lo usual.

3.1.3 Sintaxis básica del *applet*

La sintaxis del HTML está basado en etiquetas de inicio y de fin (<HTML></HTML>), igual en Curl, pero este tiene “{“ y “}”. El *applet*, siempre debe de llevar el encabezado de que es un *applet*, la versión de la API de Curl y el número de la licencia.

3.2 Agregar texto a la página *web*

3.2.1 Escribir los encabezados de las páginas

Para poder escribir los encabezados enumerados en la página *web* se necesitan las siguientes instrucciones.

{heading level= *numero*, *contenido***}**

Donde numero*,* es un número entero el cual indica que nivel tiene el encabezado y contenido indica el texto del encabezado.

Para poder colocar encabezados con numeración por niveles se puede utilizar la siguiente instrucción:
{numbered-heading level = *numero, contenido*}

1 Nivel 1
1.1 Nivel 2
1.1.1 Nivel 3

Donde *numero*, es el número que indica el nivel del encabezado y *contenido* es el texto que el encabezado desplegará

3.2.2 Dar formato al texto

La funcionalidad de estas instrucciones es muy similar a la de HTML, las opciones del lenguaje Curl son más variadas y se pueden utilizar más expresiones. A continuación se presenta una comparación de los lenguajes, primero un ejemplo en Curl y luego su similar en HTML:

```
{text
        font-family = “verdana”,
        font-size =12pt,
        font-style = “italic”,
        color = “green”,
        text-underline? = true,
        font-weight = “bold”,
        Hola Mundo
}
```

Para poder traducir esta instrucción de Curl a HTML se necesitaría anidar todas las opciones dentro de la etiqueta “<FONT>”:

```
<FONT  face= “verdana” size = “12pt”  color = “green”>
    <I><U><B>Hola Mundo</B></U></I>
</FONT>
```

Además, en Curl, con las instrucciones básicas de texto, se tiene una gran gama de expresiones. Las instrucciones básicas de texto son las siguientes:

- {text }
- {paragraph }

Estas dos instrucciones comprenden lo que se denomina las herramientas de texto en Curl. Cada una de ellas tiene un uso muy parecido pero tienen diferencias muy marcadas.

La instrucción {text } sirve para escribir texto simple en la página, esta instrucción por sí sola no modifica el texto que se escribe, se necesita la ayuda de otras instrucciones las cuales le agregan gran utilidad.

La instrucción {paragraph}, es la instrucción <P></P> en HTML, esta instrucción es utilizada para delimitar el párrafo y colocar sus opciones. Se puede delimitar el espaciado, la justificación, tipo de párrafo, la sangría, etc.

3.2.3 Presentación de la página

El HTML utiliza para su presentación básicamente la instrucción <TABLE></TABLE> para colocar los objetos en la página. Curl utiliza una colección más variada de instrucciones para poder dar el formato a la página, entre las cuales se encuentran

- {HBox } y {VBox}, con las cuales se pueden hacer cuadros de flujos horizontales y verticales. Estos tipos de cuadros de flujo pueden contener otros cuadros de flujo, texto, imágenes u objetos. La única diferencia entre ellos es la forma en que el flujo de texto es presentado al usuario, ya sea vertical u horizontal.

- {TextFlowBox }, arregla el texto para que se justifique al tamaño de la caja de flujo. Cuando se escribe una línea de texto y el espacio donde se muestra se acaba, se corta todo el texto y se escribe el resto del texto en una nueva línea abajo.

- {Frame }, se utiliza para contener una imagen, texto, cajas verticales y horizontales, otros marcos.

- {Fill }, aquí se puede controlar la forma en la que se colocan los objetos en un marco.

4. PROPIEDADES DE LA RED DE CURL

4.1 Datos persistentes del cliente

Cuando se trabaja con el lenguaje Curl se tienen pequeños paquetes de información los cuales son guardados del lado del cliente sin la necesidad de escribir un archivo, cuando se almacena esta información en el repositorio sólo cierto tipo de información se puede guardar. Existen dos de tipos de datos persistentes que se pueden almacenar *private* y *shared* (privado y compartido). Cada *applet* tiene permisos especiales y éstos no pueden leer o escribir archivos, a menos que el usuario lo autorice o que tengan el privilegio de hacerlo. Cuando se tienen *applets* privilegiados con permisos de escritura se deben guardar datos persistentes en lugar de escribir en archivos por las siguientes razones

- No existe la necesidad de encontrar espacio suficiente para que el *applet* escriba los datos, el *plug-in* de Curl realiza esta tarea automáticamente.

- Los datos persistentes se guardan en el formato que el programador necesita sin necesidad de realizar una conversión como en el caso de los archivos.

Los datos persistentes se guardan en un repositorio en el disco del cliente. Cuando se guardan los datos persistentes se guardan en pares de nombre y valor, parecido a la utilidad de una tabla *Hash*. Lo que se guarda en el repositorio son representaciones de los objetos en memoria con un formato que se pueda guardar en el disco. Los datos persistentes tienen otras propiedades muy interesantes:

- Tienen fecha de caducidad si no se usan dentro de cierto periodo de tiempo
- Se le puede limitar el tamaño de datos que pueden escribir en el repositorio
- Pueden ser compartidos para ser consultados por otros *applets*.

Sólo cierto tipo de datos se pueden guardar con los datos persistentes, estos tipos son los soportados por el lenguaje Curl

- Los tipos de datos *null* (datos tipo nulo)
- Los tipos de datos básicos: *bool, char, double, flota, int, int8, uint8, int16, uint16, int32 e int64*.
- Otros datos que especifican una característica como: ángulos, distancias y frecuencias
- Cadenas
- *Array-of, FastArray-of y HashTable-of* cada uno de ellos son tipos contenedores los cuales pueden almacenar otros tipos de datos antes mencionados.

Cuando se utilizan los datos persistentes privados éstos sólo pueden ser consultados por el *applet* que creó los datos. Para realizar esta tarea tan básica se utiliza un identificador único para cada *applet* llamado URI.

Para poder definir un repositorio de datos persistentes privado se necesita escribir la instrucción *persistent-data* antes de cualquier línea de código, esta instrucción debe de aparecer antes de cualquier intento de leer o escribir del repositorio

{persistent-data comentario: String, duración: Time=180days, tamano-max*:* int=max-int}: void

Esta instrucción tiene varios parámetros:

- Comentario: se guarda en el repositorio para darle al campo una pequeña descripción de lo que se guarda.

- Duración: este dato es guardado en el repositorio para poder darle un tiempo de caducidad al dato, este tiempo se toma desde la última vez que el usuario utilizó el dato guardado antes de eliminarlo. Se utiliza para no guardar datos que ya no tienen validez y para que no ocupen espacio en el cliente. El tiempo debe ser calculado dependiendo de cuán importante sea el dato y qué tan frecuentemente se utiliza el *applet*.

- Tamano-max: este es el número máximo de bytes que se pueden guardar en este repositorio.

Para poder leer y escribir datos del repositorio se utilizan dos procedimientos para definir los datos. Primeramente para guardar datos en el repositorio se tiene que utilizar la siguiente instrucción *set-persistent-data*, mandando todos los parámetros que anteriormente se describieron para poder guardar el objeto. Después de agregar todos los datos en el repositorio se debe de llamar a la instrucción *commit-persistent-data*, para poder guardar los datos del repositorio en el disco. Para poder llamar a los datos que fueron almacenados en el repositorio se necesita llamar a la siguiente instrucción *get-persistent-data*, la cual devuelve un objeto del tipo *any* el cual puede contener tipos primitivos de datos, como por ejemplo enteros, caracteres, cadenas, etc. Si el dato pedido no existe en el repositorio se levanta una excepción de error la cual se utiliza para dar el mensaje.

Cuando se tienen varios *applets* en una página que desean leer datos persistentes del repositorio, se necesita que estos datos estén compartidos para que varios *applets* puedan leer de ellos. Por lo general la información que se desea pública son las preferencias del usuario que visitó la página. Existen ciertas limitaciones para poder compartir los datos del repositorio. A continuación se describen cuáles son aquellas limitaciones de seguridad de los *applets*

- Los *applets* privilegiados pueden tener acceso a todos los repositorios de datos compartidos.

- Applets que son cargados en el sistema local no pueden compartir los repositorios de datos, por la razón de que no hoy forma de determinar si los *applets* están relacionado entre sí y si deberían compartir los datos.

- Los *applets* son cargados desde un lugar remoto si y sólo si están en el mismo directorio padre o hijo directamente del *applet* que creo el repositorio.

Cada uno de los *applets* tiene limitaciones para el tamaño máximo de información que puede guardar además de las limitaciones de lo que pueden tener acceso. Cada *applet* tiene la limitación de abrir como máximo 20 repositorios de datos y los repositorios de datos están limitados a 16K.

4.2 Configuración del *plug-in*

Cuando se utilizan los *applets* se pueden mezclar con páginas Curl cuando está contenido en una página ésta se queda en el área destinada de la pantalla. Cuando el *applet* es cargado directamente, éste toma toda el área visual del explorador de Internet.

4.2.1 Mezclando un *applet* con una página HTML

Para poder mezclar un *applet* en una página de HTML, se debe de utilizar la instrucción <EMBED ...> esto hace que el *applet* se muestre en un área rectangular dentro de la página. Si se desea que el *applet* tome el control de toda la pantalla en lugar de que se muestre en un área rectangular, se debe de cargar el *applet* directamente. Cuando se navega por varios *applets* en el explorador, éstos se guardan en el historial del explorador, con esta característica se utilizan las barras del historial de atrás y adelante.

4.2.2 Las clases *applets*

Para poder comunicar el *applet* con las acciones que el usuario ejecuta se necesita establecer una interfase, para esto se utiliza la clase *applet*. Esta interfase se comunica con el usuario y con el navegador que lo contiene. Este *applet* puede ser cargado desde cualquier lugar. Puede cambiar el título de la página que lo contiene y otras opciones para facilitar el uso con el usuario.

4.3 Configuración del servidor *web*

Una vez que se han desarrollado todas las páginas en Curl se necesita configurar el servidor *web* para que pueda entregar las páginas a los clientes sin ningún problema.

Para que se puedan servir los *applets* de Curl desde el sitio se necesita configurar los tipos *Multipurpose Internet Mail Extensions* (MIME), en la configuración del servidor *web*. Se necesitan agregar dos tipos a las extensiones MIME del servidor:

Tipo de archivo	Extensión	Tipo MIME
Archivo de *applet* Curl	.curl	text/vnd.curl
Paquete	.pcurl	application/vnd.curl.pcurl

4.4 Soporte de red y cliente-servidor y seguridad

El *plug-in* de Curl puede leer los archivos remotos de *web* de la misma manera que lee un archivo o recurso local. La única diferencia entre leer de un archivo local y uno que se encuentre en un servidor *web* es el URI usado para poder leer el archivo.

Los *applets* no pueden cargar archivos desde otro lugar que no sea el mismo del cual han sido cargados, esto se puede realizar si el *applet* se encuentra con los permisos y privilegios correspondientes. Tienen una capacidad restringida de crear conexiones por los puertos, los *applets* solamente pueden

- Realizar conexiones salientes. Estos no pueden aceptar conexiones entrantes, es decir, si un agente externo solicita una conexión será denegada.

- Los *applets* sólo pueden conectarse con el equipo del cual fueron bajados, hasta que el equipo remoto permite que la conexión esté activa.

- Abrir conexiones a ciertos puertos, los *applets* no pueden abrir conexiones con puertos que comprometan la seguridad del sistema.

Para el protocolo TCP (*Transmission Control Protocol*) existen restricciones en los puertos que utilizan los *applets*..

A continuación se presenta una tabla con los puertos menores del 1024 donde se pueden utilizar los *applets*:

Tabla VI. Puertos menores de 1024 que se pueden usar en TCP

Puerto	Descripción
70	Gopher
80	http
119	NetNews
194	IRC
443	HTTPS

A continuación se presenta la lista de los puertos mayores de 1024 que no se pueden utilizar:

Tabla VII. Puertos mayores al 1024 que no se pueden usar en TCP

Puerto	Descripción
1352	Lotus Notes®
1524	Ingres
1525,1527,1528 y 1529	Oracle®
2401	CVS client/*server*
2053	Kerberos de-multiplexor
2105	Kerberos encrypted login
3306	MySQL
6000-6003	X11
7000-7009	AFS
10081-10083	Amanda

Cuando los *applets* tratan de utilizar el protocolo UDP (*User Datagram Protocol*) será negada la petición de conexión de todos los puertos abajo del 1024 con excepción de los siguientes:

Tabla VIII. Puertos menores del 1024 que pueden usar UDP.

Puerto	Descripción
70	Gopher
80	http
194	IRC
443	HTTPS

Todos los puertos mayores del 1024 se pueden utilizar con el protocolo UDP con excepción de los siguientes puertos:

Tabla IX. Puertos mayores del 1024 que no pueden utilizar UDP.

Puerto	Descripción
1352	Lotus Notes®
1524	Ingres
1525,1527,1528 y 1529	Oracle®
2401	CVS client/*server*
3306	MySQL
7000-7009	AFS
10081-10083	Amanda

4.5 Utilizando XML con SAX

La herramienta SAX, es una interfase para interactuar con la API de XML, esta herramienta sirve para todos los *parsers* que utilizan XML, como un ODBC que interactúan con varias bases de datos relacionales, lo mismo es SAX, interactúa con varios *parsers* de XML.

La versión de SAX API que se utiliza en el lenguaje Curl es la Java-SAX 2.0 API, esta versión de la API no da soporte a las clases de SAX 1, que fueron desaprobadas en la versión 2. Los nombres de las clases utilizados aquí son los mismos que se utilizan en Java-SAX. Para poder utilizar la herramienta SAX se importa al *applet* el siguiente paquete CURL.XML.SAX.PARSER

Con este *parser* se puede leer un archivo XML para poder separar todas las etiquetas y los campos que se especifican.

4.6 Soporte SOAP

El lenguaje Curl soporta el protocolo SOAP 1.1 sobre HTTP para poder utilizar los servicios *web*. Este soporte tiene varias características

- Soporte automático de los esquemas de XML con los tipos básicos de Curl.
- La creación de descriptores del lenguaje correspondientes a llamadas a procedimientos remotos.
- Definir los tipos de XML.
- Mapeo de los errores de SOAP a excepciones de Curl.

Cuando se agrega un servicio *web* a una página, todo el código está en el cliente, pero periódicamente se necesita ir al servidor para datos. En el *applet* se necesita colocar el código para la petición de los datos.

El protocolo SOAP está basado en XML para el intercambio de información en un ambiente distribuido. Curl soporta las peticiones de SOAP sobre HTTP porque éste convierte los tipos de XML a los tipos de Curl. El protocolo SOAP consiste además de

- Un componente que utiliza el cliente para llamar operaciones de servicios *web* descritas por un documento W*eb Services Description Language* (WSDL).

- Un componente del lado del servidor que verifica y transfiere las llamadas a un servicio *web* utilizando un objeto COM descrito por archivos *WSDL*. Este archivo es necesario para la implementación de SOAP.

A continuación se especifican los pasos para que desde el lenguaje Curl se realicen llamadas a servicios de SOAP sobre HTTP, se deben de seguir los siguientes pasos

- Recolectar la información necesaria sobre la interfaz del servicio SOAP. Esto incluye el URL, el encabezado de *SOAPAction* HTTP, el nombre de la operación, los nombres y los tipos de entrada y salida de los argumentos. Toda esta información necesaria se encuentra en el WSDL.

- Como cada uno de los tipos del protocolo SOAP son tipos de XML tienen que ser convertidos a tipos de Curl. El lenguaje Curl puede convertir entre los tipos de XML y los de Curl, también da los medios para realizar la conversión. Si todos los tipos son compatibles no hay necesidad de realizar una función.

- Con la información de los pasos descritos anteriormente, se debe de inicializar la instancia de S*oap-1-1-HttpOperation* e instancias de *Soap-1-1-StandardArgumentDescriptors.* Se crea una instancia de *Soap-1-1-StandardArgumentDescriptors* por cada argumento, ya sea de entrada o de salida.

- Llamar al servicio SOAP utilizando los métodos de llamada que se encuentran en la instancia de *Soap-1-1-HttpOperation* que se inicializó en el paso anterior.

Cada llamada a un servicio SOAP tiene las mismas restricciones de seguridad que las de un *applet*.

4.7 Soporte de archivos y datos

Cuando se trabaja con recursos, como archivos y datos, el lenguaje Curl desea hacer esta tarea lo más simple al usuario como se pueda y que éste se concentre en la lógica del programa y no en como poder acceder a la información.

El lenguaje Curl le permite escribir y leer datos de los recursos. Además, le da la opción de poder eliminar los archivos del sistema de archivos. Para poder realizar estas tareas se necesita identificar la ubicación del recurso.

En Curl se identifica a los recursos, ya sean archivos en el sistema de archivos local o una página *web* en un servidor remoto, con un *Universal Resource Identifier* (URI). Este término es muy similar a los que se utilizan en los buscadores de Internet, llamados URL.

A continuación se muestran unos ejemplos de URIs

- www.mipagina.com/consultas/datos.asp. Este es un archivo que se encuentra en un servidor *web* remoto.
- file:///Archivos/locales/datos/clientes.doc. Este es un ejemplo de un archivo que se encuentra en el sistema de archivos local.

Una de las tareas más básicas de la programación es leer y escribir los archivos. Cuando el usuario ha trabajado en cierta tarea que desea continuar más tarde, se necesita guardar este trabajo, por esta razón, se necesita la opción de poder guardar los archivos.

El proceso para poder leer o escribir datos a un recurso tiene los siguientes pasos:

- Identificar el recurso por medio de un URL u otro objeto.

- Abrir un flujo de datos al recurso declarando una instancia de flujo de datos.

- Leer o escribir datos en el recurso utilizando los métodos del flujo de datos.

- Cuando ser termina la operación de leer o escribir del recurso se debe de llamar al método *close* de la instancia para poder cerrar el recurso y grabar los cambios que se realizaron.

Leer datos desde un archivo

La forma más sencilla de poder leer un archivo es abrirlo de modo texto. Para poder leer datos desde un archivo de texto se necesita declarar un objeto tipo *TextInputStream*. Este es un objeto de flujo de datos que representa una secuencia de datos de un recurso.

El método de leer datos desde un archivo que se encuentra en el disco local hasta un archivo que se encuentra en la red local es el mismo. A continuación se muestra un ejemplo de cómo leer datos desde un archivo local.

```
{curl 1.7 applet}
{applet license=”development”}

|| Se debe de obtener la dirección donde se encuentra el recurso
{value
   let buffer:TextFlowBox = {TextFlowBox width=6in, border-width=1pt}
   {try
      let myinput:#TextInputStream=
      {read-open {url "file://D:/CURL/prueba.txt"}}
        {until myinput.end-of-stream? do || lee datos del archivo hasta que
        se acaba

          || cada linea que se lee del archivo se agrega a la caja de texto
          {buffer.add {myinput.read-one-line}}}
      {myinput.close}                  ||cierra el archivo
    catch err:MissingFileException do
      {error "No se puede encontrar el archivo"}  }
   {value buffer}}
```

Para poder profundizar más en los errores e instrucciones que se pueden utilizar al momento de abrir un archivo favor de utilizar la ayuda que trae el *plug-in* de Curl.

A continuación se presenta un ejemplo el cual escribe un archivo al disco duro.

```
{curl 1.7 applet}
{applet license = "development"}
Este es un applet que se encarga de escribir a un
archivo

{let Name:Url = {url "file://D:/CURL/salida.txt"}} ||el nombre del archivo
{let salida:TextOutputStream = {write-open Name, error-if-exists?=false}}

{let numout:int= {salida.write-one-string "Hola este es una cadena que se
excribe en un archivo"}}

{salida.close}
```

4.8 Seguridad en los *applets*

Los *applets* de Curl están restringidos por la seguridad de los usuarios, para que otras personas no deseadas puedan aprovecharse de las opciones de Curl de leer y escribir archivos. Cuando un *applet* quiere leer o escribir un archivo necesita la aprobación por parte del usuario para que se realice la operación.

Las restricciones en los *applets* tienen dos objetivos. Primero proteger el sistema del usuario de *applets* maliciosos y de *applets* que han sido programados deficientemente, esto incluye proteger el sistema del usuario y el uso de los recursos del sistema. Segundo, prevenir que información valiosa del usuario sea violada.

Un *applet* tiene restricciones sobre otras operaciones como:

- Incluir controles *ActiveX*
- Manipular la tabla de procesos de ninguna forma
- Realizar operaciones de transferencia de archivos a un nivel bajo
- Acceder a variables de ambiente
- Acceder a preferencias del sistema
- Acceder a las entradas del registro de *windows*.

Cuando un *applet* es cargado desde una red local se toman en cuenta ciertas restricciones: el *applet* solamente se puede conectar al sitio de donde fue cargado, si éste se lo permite, los *applets* no pueden aceptar conexiones de redes remotas ni locales.

Al momento de que un *applet* quiera acceder a un archivo local, el *plug-in* de Curl da una excepción de error, el cual se puede leer como cualquier mensaje de error. Los *applets* no tienen permiso de leer un archivo del sistema local de archivos, estos deben de ser *applets* privilegiados para poder realizar esta tarea. También los *applets* pueden utilizar la ventaja de los datos persistentes para grabar la información localmente, toda la información grabada en los datos persistentes no puede ser transmitida a otra máquina.

CONCLUSIONES

1. Este trabajo expone la necesidad de los clientes por una nueva tecnología que resuelva muchos de los problemas que la tecnología actual mantiene. Con las ventajas que se expresan en el lenguaje Curl, se pueden realizar cambios importantes a la estructura de los sistemas de informática que los usuarios utilizan.

2 Curl por ser un lenguaje *script* que se ejecuta del lado del cliente, ayuda a bajar la carga de trabajo de los servidores *web*.

3 Por tener su base en la tecnología de páginas *web*, la distribución de la aplicación se realiza de una forma más simplificada

4 Curl toma las ventajas de varias tecnologías y lenguajes, como la tecnología de objetos de C++ y la lógica de etiquetas de HTML, para crear un lenguaje que puede darle un apoyo importante a la tecnología *web* que actualmente se utiliza.

RECOMENDACIONES

1. A las empresas que tienen aplicaciones en páginas *web* tomar en cuenta las ventajas de Curl para las aplicaciones que actualmente se tienen desarrolladas.

2. Aprovechar la capacidad de procesamiento de datos de los clientes para no saturar a los servidores *web* y revisar la infraestructura con la que se cuenta.

3. Análisar exhaustivamente las ventajas y desventajas que se tienen actualmente al utilizar la tecnología de páginas *web* dinámicas y realizar un estudio para determinar si las nuevas tecnologías aportan una solución a las limitaciones que se tienen actualmente.

BIBLIOGRAFÍA

1. http://www.curl.com, Estados Unidos, 2002, 15 de mayo de 2004

2. http://www.curlbreaker.com, Estados Unidos, 2002, 15 de mayo de 2004

3. http://www.curlexamples.com, Estados Unidos, 2002, 15 de mayo de 2004

4. Gordon Michael; Ullman Chris y Joly James. ***Early adopter Curl*.** Estados Unidos: Wrox Press Ltd. 2001, 279pp.

APÉNDICE

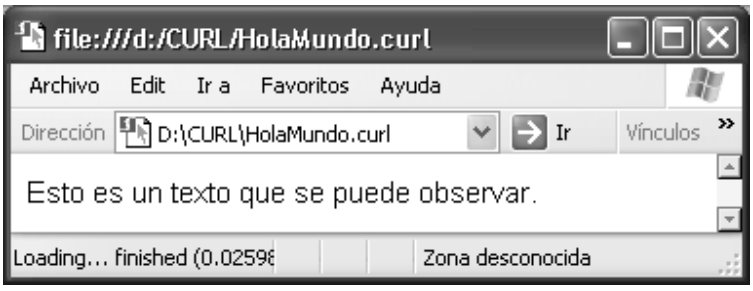
Instrucciones más útiles en el lenguaje Curl

A continuación se muestra las instrucciones más útiles del lenguaje Curl para la creación de páginas *web*. Se le muestra al usuario las instrucciones para mostrar texto en la página, los tipos de datos que se pueden utilizar en el lenguaje, declaración de variables y el uso de ellas. Se muestra también el uso de los ciclos y los procedimientos.

Desplegando texto

Una de las operaciones básicas que se encuentran en cualquier lenguaje de programación es el de desplegar texto. Para esto se tienen varias instrucciones que realizan esta tarea, *text* y *paragraph*.

```
{curl 1.7 applet}
{applet license=”development”}
Esto es un texto que se puede observar.
```



Un ejemplo similar al anterior pero utilizando las instrucciones de texto es

{curl 1.7 applet}
{applet license = “development”}
{text Esto es un texto que se puede observar}

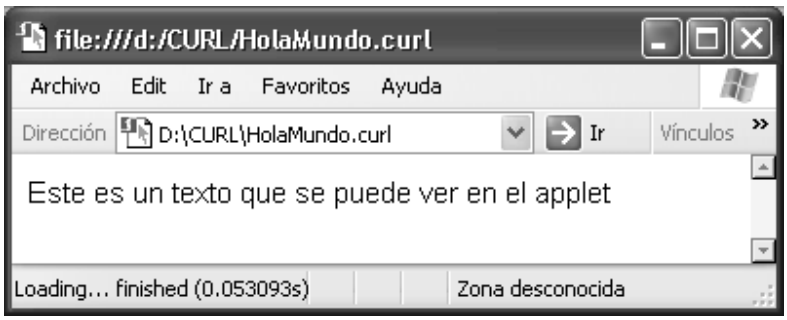
Ó

{curl 1.7 applet}
{applet license = “development”}
{paragraph Esto es un texto que se puede observar}

Comentarios

Como en todo lenguaje de programación, es necesaria una documentación interna del código para una mejor comprensión del mismo. Por eso se pueden agregar comentarios en el código del *applet*.

{curl 1.7 applet}
{applet license = “development”}
Este es un texto que se puede ver en el *applet* || Este es un comentario no se muestra.



Cuando existe la necesidad de comentar varias líneas de código, se puede utilizar como a continuación se muestra

```
{curl 1.7 applet}
{applet license = “development”}
|#
        Este es un comentario de varias líneas
        Que no se mostrará en el applet
#| Pero esto sí se muestra
```

Opciones de texto

Las opciones de texto adicionales para texto son aquellas que dan más opciones de presentación al usuario. Ya sean de tipo de letra, tamaño, color, alineación, etc. Entre estas opciones se encuentran

- *color,* cambia el color de la fuente, los colores disponibles son los que se pueden utilizar en Curl.

- *font-family,* especifica el tipo de fuente que se utiliza en el texto, dependiendo del tipo de sistema que se encuentre.

- *font-size,* aquí se especifica el tamaño de la fuente. Este tamaño puede ser medido en puntos (pt), en centímetros (cm), pulgadas (in), y en porcentaje del tamaño de la fuente predeterminada (em)
- *font-style,* se puede especificar si la fuente es itálica u oblicua.

- *font-weight,* define si la fuente se muestra en negritas o no.

- *text-underline?,* define si el texto se subraya o no, los valores son verdadero o falso.
- *text-line-through?,* define si el texto que se muestra tiene una línea que lo atraviesa.
- *text-breakable?,* un valor boleano que define si el texto se puede dividir en varias líneas.

A continuación se muestra un ejemplo utilizando las opciones de texto que anteriormente se describieron:

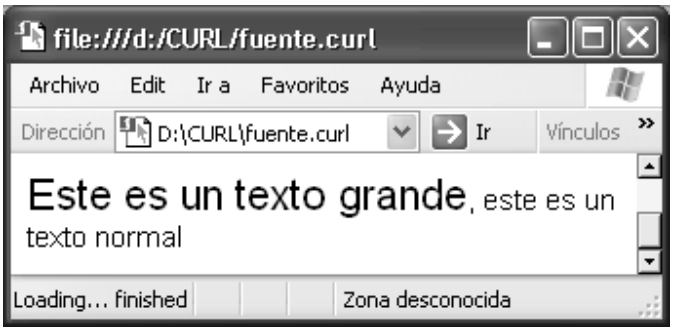
{curl 1.7 applet}
{applet license = “development”}
{text color=”green”, font-size=16pt, text-underline?=true, Este es un texto verde, de tamaño 16 y subrayado}

Formateado del texto

Existen otras formas de presentar texto además de las opciones de presentación de texto. A continuación se presentan otras opciones de instrucciones para desplegar texto con formato

- *tiny,* equivale a {text font-size=”0.6em”...}
- *small,* equivale a {text font-size=”0.8em”…}
- *big,* equivale a {text font-size=”1.5em”…}
- *huge,* equivale a {text font-size=”2.0em”…}
- *italic,* equivale a {text font-style=”italic”…}
- *bold,* equivale a {text font-weight=”bold”…}
- *underline,* equivale a {text text-underline?=true…}

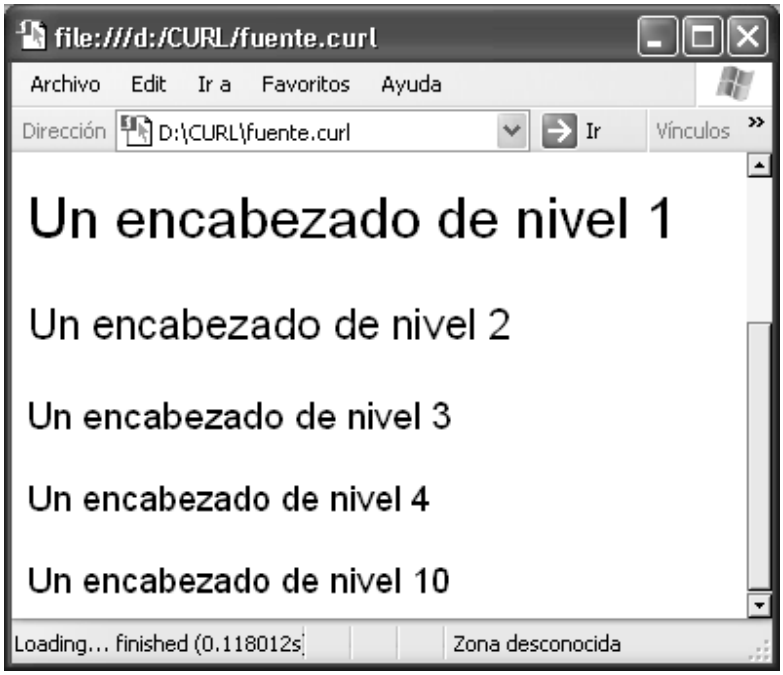
{curl 1.7 applet}
{applet license = “development”}
{big Este es un texto grande}, este es un texto normal



Encabezados

Esta instrucción tiene su equivalente en HTML desde <h1> hasta <h6>. En Curl se encuentra hasta el nivel 4. Después del nivel 4 todos los niveles son iguales.

```
{curl 1.7 applet}
{applet license = “development”}
{heading level=1, Un encabezado de nivel 1}
{heading level=2, Un encabezado de nivel 2}
{heading level=3, Un encabezado de nivel 3}
{heading level=4, Un encabezado de nivel 4}
{heading level=10, Un encabezado de nivel 10}
```



Tablas

Las tablas tienen un nivel de complejidad más avanzado que las opciones para formatear texto. Las tablas tienen las siguientes partes

- *row,* para cada una de las filas de la tabla
- *header-row,* para el encabezado de la tabla
- *cell,* para cada celda de la tabla
- *header-cell,* para las primeras celdas de la primera fila.

Para poder crear una tabla básica se puede escribir el siguiente código:

```
{curl 1.7 applet}
{applet license = “development”}
{table
        {row
                {cell Perro}
                {cell Gato}
        }
        {row
                {cell Pez}
                {cell Tiburón}
        }
}
```

Cuando se utilizan los encabezados de tabla, las instrucciones *header-row* y *header-cell* ayudan a resaltar los encabezados.

```
{curl 1.7 applet}
{applet license = “development”}
{table
        {header-row
                {header-cell Amigables}
                {header-cell No amigables}
        }
        {row
                {cell Perro}
                {cell Gato}
        }
        {row
                {cell Pez}
                {cell Tiburón}
        }
}
```

Imágenes

En Curl las imágenes pueden ser insertadas desde una dirección URL o desde un archivo del disco duro. A continuación se muestra como insertar la imagen:

{curl 1.7 applet}
{applet license ="development"}
Esta es una imagen
{image source= {url "file:///D:/CURL/Muestra.jpg"}}

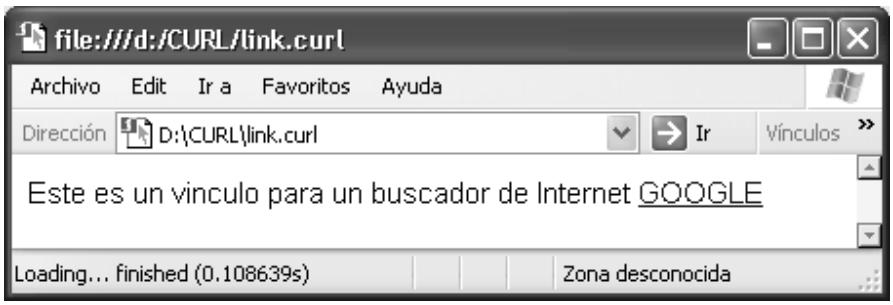
Vínculos

Los vínculos son utilizados para poder llamar a otros archivos que se encuentran en el sitio de Internet. La sintaxis de esta instrucción es la siguiente:

{link href = {url “vinculo”}}

Ejemplo:
{curl 1.7 applet}
{applet license = “development”}

Este es un vínculo para un buscador de Internet
{link href = {url “http://www.google.com”}, GOOGLE}



Tipos y sintaxis

Lo que hasta ahora se ha visto es el desplegar texto en la página, lo cual tiene una dificultad mínima. Lo que se tiene que tener en cuenta es la versión de la API que se utiliza en el lenguaje. Las versiones que hay en Curl son la 1.5, 1.6 y actualmente se utiliza la 1.7 (los ejemplos de este apéndice están hechos en la versión 1.7).

En Curl hay dos clases de tipos, los tipos por valor y los tipos por referencia, los tipos por valor son aquello que se utilizan para guardar un valor o grupos de valores, ejemplo, el radio, las coordenadas del centro de un círculo.

Los tipos por valor contienen:

- Tipos primitivos (*int, char, double*, y demás)
- Cantidades
- Tipos enumerados

Los tipos por referencia incluyen a los siguientes:

- Clases

- Interfaces
- *Strings*
- Colecciones
- Y cualquier tipo implementado por clases

TABLA X. Tabla de tipos primitivos

Nombre	Descripción	Valor Predeterminado	Nota
bool	Valor boleano	Falso	*true* o *false*, no se puede cambiar por 0 ó 1
byte	Entero de 8 bits sin signo	0	Rango entre 0-255
char	Caracter simple	'\u0000'	Un carácter de 16 bits
double	Número con punto flotante doble	0.0	
float	Número con punto flotante simple	0.0f	
int	Entero de 32 bits con signo	0	
int8	Entero de 8 bits con signo	0	
int16	Entero de 16 bits con signo	0	
int32	Entero de 32 bits con signo	0	Es como el *int*
int64	Entero de 64 bits con signo	0	

uint16	Entero de 16 bits sin signo	0	
uint8	Entero de 8 bits sin signo	0	

TABLA XI. Operadores soportados por Curl

Categoría	Operador
Operadores unitarios	Menos unitario -
Aritméticos	+ - * / div mod rem
Lógicos	and, or, not
Concatenación de cadenas	&
Relacionales	= = != < <= > >= isa
Conversión	Asa

Precedencia de los operadores

A continuación se muestra el orden descendente en que los operadores son evaluados

Grupo	Operadores
	( )
Operadores unitarios	- (Negación unitaria)
Multiplicación	* / div mod rem
Suma	+ -
Conversión	Asa
Cadenas	&
Relacionales	= = != < > <= >= isa
NOT lógico	Not
AND lógico	And
OR lógico	Or
Asignación	=

Creación e inicialización de variables

Al momento de la declaración de variables, se tiene que tomar en cuenta de que tipo son las variables a declarar, si son de valor o de referencia, pero en ambos casos se usa la instrucción *let*. A continuación se muestra un ejemplo para la declaración de variables

```
{curl 1.7 applet}
{applet license = “development”}
{let i1:int}                    ||    i1 : inicializada con el valor predeterminado, 0
{let i2:int64 = 12}             ||    i2 : tipo int64 inicializado con 12
{let f1:float}                  ||    f1 : inicializado con el valor predeterminado,
0.0
{let f2:float= 3.1415f}         ||    f2 : tipo float inicializado con 3.1415f
{let d1:double}                 ||    d1 : inicializado con el valor predeterminado, 0
{let d2:double=3.1415}          ||    d2 : tipo double inicializado con 3.1415
{let b1:bool}                   ||    b1 : inicializado con falso
{let b2:bool=true}              ||    b2 : tipo bool inicializado con el valor true.
{let c1:char,                   ||    c1 : inicializada con el valor ‘\u0000’
      c2:char=’Z’,              ||    c2 : inicializada con el valor ‘Z’
      c3:char=’\u20AC’}         ||    c3 : inicializada con el valor ‘\u20AC’
```

Para la declaración de las variables hay reglas a seguir:

- Los identificadores de las variables son sensibles a las mayúsculas y minúsculas, no es lo mismo miclase que MiClase.
- Un identificador debe de comenzar con una letra o con el carácter de subrayado.
- Los siguientes caracteres pueden ser letras, dígitos, símbolo de subrayado, signo de interrogación, signo de menos.
- Los espacios no se pueden utilizar al momento de declarar una variable.
- Los caracteres utilizados deben de existir en el estándar *Unicode*.

Cuando a una variable se le quiere cambiar el valor con el cual fue inicializado se utiliza la siguiente instrucción:

```
{set variable =valor}


{curl 1.7 applet}
{applet license = “development”}

{let i1:int=0,
        c1:char=’a’,
        b1:bool=true,
        str1:String= {String “Una prueba…”},
        str2:#String}
{let any1=1}
{set i1=123}
{set c1=’s’}
{set b1=false}
{set any1=str1,
        str1= {String “…Para probar las propiedades de Curl”},
        str2=str1}
```

Cantidades

A las variables se les puede asignar un valor de medida. Entre la cantidad y la medida, no debe de existir un espacio en blanco. A continuación se muestran las medidas que soporta el lenguaje Curl

Tipo de medida	Unidad predeterminada	Abreviación
Aceleración	M/s^2	-
Ángulo	radian	rad
Distancia	metros	m

Fracción	-	-
Frecuencia	1/s	-
Intensidad	candela	cd
Masa	gramos	g
Porcentaje	-	-
Resolución	1/m	-
Velocidad	M/s	-
Tiempo	segundos	S

Las variables de medidas, tal como las variables normales se pueden declarar y asignarles nuevos valores.

```
||cantidades.curl
{curl .17 applet}
{applet license="development"}

{let
        d1: Distance,                    || Usa un valor predeterminado
        d2: Distance=7.5m,               || Asignándole valor nuevo
        m1: Mass=125grams,               || se usa grams en lugar de g
        m2: Mass=2kg,                    || se aplica el prefijo k de kilo
        t1: Time=12hr,                   || se utiliza una medida compatible
        f1: FloatAcceleration=2f (m/s^2) || se utiliza un float en lugar de double
}
d1 = {value d1}{br}          ||muestra d1 = 0m
d2 = {value d2}{br}          ||muestra d2 = 7.5m
m1 = {value m1}{br}          ||muestra m1 = 125g
m2 = {value m1}{br}          ||muestra m2 = 2000g
t1 = {value t1}{br}          ||muestra t1 = 43200s
```

```
f1 = {value f1}                    ||muestra f1 = 2(m*s^-2)
```

Constantes

A una variable se le puede asignar un valor que no cambie con la palabra reservada *constant*. Por definición, las constantes no pueden cambiar, así que se les debe de asignar un valor cuando se declaran.

```
{let constant i1:int}              ||constante con el valor 0
{let constant
        c1:char = 'c',
        s1:String = "Una cadena Constante"     || los valores de c1 y s1 no
cambian
}
```

La expresión *if*

La expresión *if* que se utiliza en Curl tiene la misma función de evaluar que la de cualquier lenguaje conocido. Lo que realiza *if* es evaluar si la expresión es verdadera o falsa.

```
||if.curl
{curl 1.7 applet}
{applet license = “development}

{let d1:Distance=1m,
        d2:Distance=100cm}

||vamos a ver si 1m es mayor que 100cm

{ if d1 < d2 then
        {value “1 metro es menor que 100cm”}
        else if d1>d2 then
                {value “1 metro es mayor que 100cm”}
        else {value “1 metro es igual a 100cm”}
}
```

La expresión *unless*

La expresión *unless* es muy similar a la expresión *if*, pero realiza todo lo contrario, la expresión *if* ejecuta el bloque de código si la evaluación es

verdadera, en cambio la expresión *unless* ejecuta el bloque de código si la evaluación es falsa.

```
||unless.curl
{curl 1.7 applet}
{applet license = “development”}

{let d = 3}      ||cuando se declara es tipo any, cuando se corre es tipo int

{let box:Vbox={VBox}}

{unless {type-of d} == int do
        {box.add “Raro. La variable d debería ser entero”}
}
{value box}
```

El ejemplo anterior no despliega el mensaje de texto.

La expresión *switch*

La expresión *switch* es comúnmente utilizada cuando se necesita realizar varias validaciones. Cuando la expresión no cumple ninguna de las validaciones se ejecuta la opción que se encuentra en el bloque de *else*.

```
||switch.curl
{curl 1.7 applet}
{applet license = “development”}

{ let d1:Distance=1meter}
{switch d1
```

```
        case 98cm, 99cm do              ||cuando se tiene varias opciones se
separa por ,
                {value “Hay 98 o 99 cm en un metro”}
        case 100cm do
                {value “Hay 100cm en un metro”}
        case 100cm do       ||esta expresión, aunque cierta, no se cumple, porque
                            ||ya fue hecha verdadera
                {value “Realmente hay 100cm en un metro”}
        case 101cm do
                {value “Hay 101cm en un metro”}
        else
                {value “Se creía que había 100cm en un metro, pero no es cierto”}
}
```

La expresión *type-switch*

La expresión *type-switch* es un caso especial del *switch*, en donde lo que se evalúa es el tipo de la expresión. A continuación se muestra un ejemplo del mismo

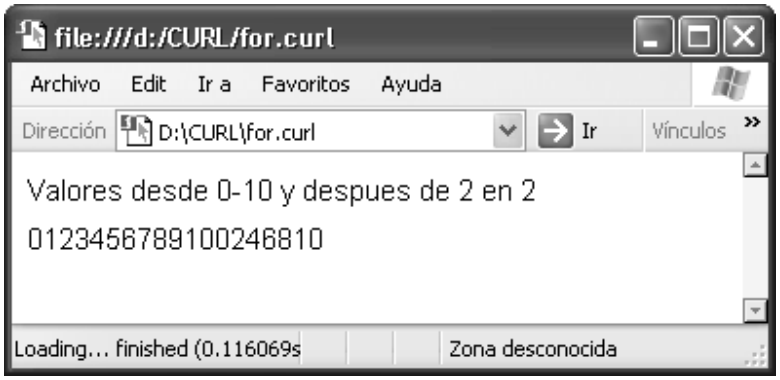
```
{type-switch(2+2)
```

```
        case i:int do
                {value i & " es un entero"}
        case f:float do
                {value f & " es un float"}
        case x:AlgunTipoDelCliente do
                {value x & " es un tipo definido por el usuario"}
        else
                {value "No se sabe de que tipo es"}
}
```

La expresión *for*

Se utiliza la expresión *for* cuando se necesita repetir un bloque de código. Esta expresión necesita un valor de inicio y uno de fin.

```
||for.curl
{curl 1.7 applet}
{applet license = “development”}

{let box:HBox  = {HBox}}
Valores desde 0-10 y despues de 2 en 2

{for i:int = 0 to 10 do
        {box.add i}             ||agrega los numeros del 0-10 en la caja
}
{value box}
||un ciclo que va de dos en dos
{for i:int = 0 to 10 step 2 do
        {box.add i}
}

{value box}
```


file:///d:/CURL/for.curl
Archivo Edit Ir a Favoritos Ayuda
Dirección D:\CURL\for.curl
Ir
Vínculos
Valores desde 0-10 y despues de 2 en 2
0123456789100246810
Loading... finished (0.116069s
Zona desconocida

La expresión *while*

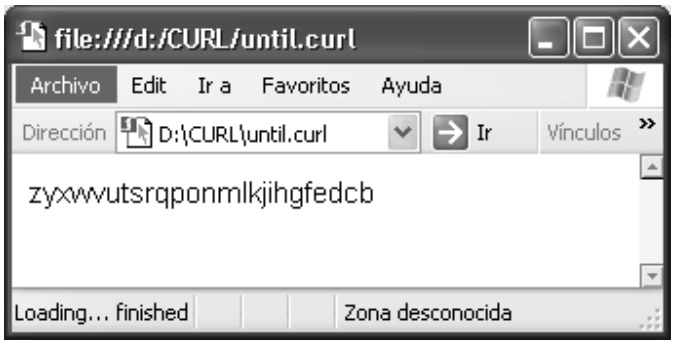
El ciclo *while* de Curl funciona de la misma manera que el de C++ y C#, este ciclo se ejecuta hasta que la expresión evaluada es falsa.

```
||while.curl
{curl 1.7 applet}
{applet license = “development”}

{let box:HBox = {HBox}}
{let count:char = ‘a’}

{value
        {while count< ‘z’ do
                {box.add count}
                set count = (count + 1) asa char
        }
        box
}
```

La expresión *until*

Esta expresión complementa al ciclo *while,* por la razón de que este ciclo se repite una o más veces hasta que la expresión es verdadera.

```
||until.curl
{curl 1.7 applet}
{applet license = “development”}

{let box:HBox={HBox}}

{let count:char = ‘z’}

{value
        {until count <= ‘a’ do
                {box.add count}
                set count = (count -1 ) asa char
        }
        box
}
```

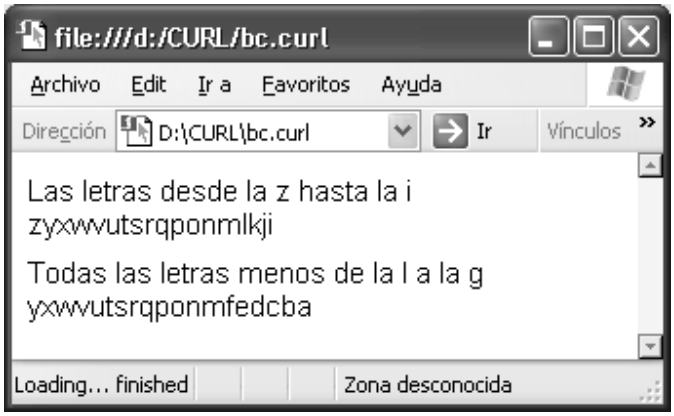


Las expresiones *break* y *continue*

Estas dos expresiones son utilizadas para poder interactuar con los ciclos. La expresión *break* se utiliza para salir de un ciclo, *for, while, until.* La expresión *continue* se utiliza para finalizar con la interacción actual y continuar con la siguiente, así que se debe de tener cuidado cuando se llama a esta expresión porque se puede dar un ciclo infinito.

```
||usando break y continue   bc.curl
{curl 1.7 applet}
{applet license = “development”}

{let box1:HBox= {HBox}}

{let count:char=’z’}
Las letras desde la z hasta la i
{value
        {while count>’a’ do
                {if count ==’h’ then
                        {break}
                }
                {box.add count}
                set count = (count – 1 ) asa char
}
box
}

{set count = ‘z’}
Todas las letras menos de la l a la g
```

```
{value
        {while count > ‘a’ do
                {set count = (count – 1 ) asa char
                {if ( count >=’g’) and ( count <=’l’) then
                        {continue}
                }
                {box.add count}
        }
        box
}
```



Procedimientos

En Curl existen cuatro tipos de procedimientos:

- Procedimiento de alto nivel
- Procedimientos anónimos
- Procedimientos de clases
- Métodos

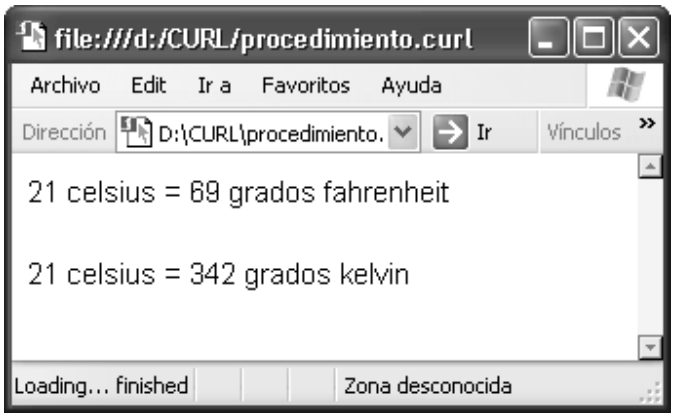
Procedimientos de alto nivel

Los procedimientos, como en *pascal* y *visual basic*, hay que declararlos antes de poder llamarlos, o mostrará un error de que el identificador es desconocido. Todos los procedimientos deben de retornar un valor, aunque éste sea vacío. Si no se especifica el tipo de retorno del procedimiento, se asume que este tipo es *any*.

```
||procedimiento.curl

{curl 1.7 applet}
{applet license = “development”}

{let celsius : int = 21,
        fahrenheit : int,
        kelvin : int
}
{define-proc {c2f celsius_in : int} : int
        let f : int = (celsius_in * (9/5) + 32) asa int
        {return f}
}

{define-proc {f2k fahrenheit_in : int} : int
        let k: int = (fahrenheit_in+273)
        {return k}
}

{set fahrenheit = {c2f celsius}}
```

{value “21 celsius = “ & fahrenheit & “ grados fahrenheit”}{br}

{set kelvin= {f2k fahrenheit}}
{value “21 celsius = “ & kelvin & “ grados kelvin” }



Los procedimientos en Curl pueden devolver más de un valor. A continuación se muestra como se puede utilizar esta gran ventaja:

```
{define-proc {c2fandk Celsius_in : int} : (int,int)
        let f:int = {c2f celcius_in}
        let k:int = {f2k f}
        {return f, k}
}

{set (fahrenheit, kelvin) = {c2fandk 45}}
```

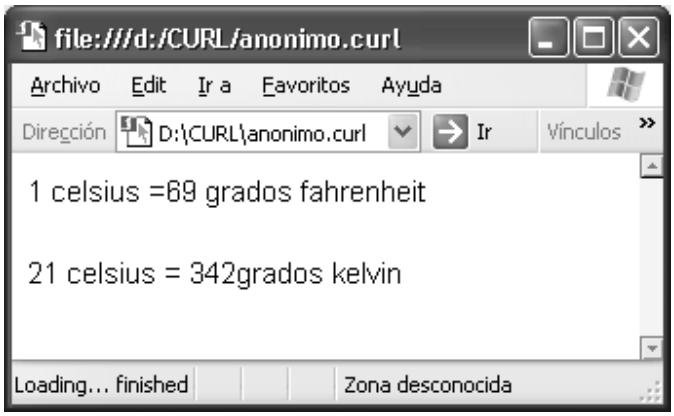
```
{value “45 celsius = “ & fahrenheit & “ grados fahrenheit o “ & kelvin & “ kelvin”}
```

Procedimientos anónimos

Los procedimientos anónimos son tratados como variables que se les puede asignar un nuevo valor.

```
||anónimos.curl
{curl 1.7 applet}
{applet license = “development”}

{let celsius:int = 21,
        fahrenheit:int,
        kelvin:int,
        changing-proc: {proc-type {int}:int}
}

||primeramente se le dice al procedimiento que convierta de celsius a fahrenheit
{set changing-proc = {proc {celsius_in:int}:int
        {return ((celsius*1.8)+32) asa int}
        }
}

||luego se llama
{set fahrenheit = {changing-proc celsius}}
{value “21 celsius =“ & fahrenheit & “ grados fahrenheit “}{br}

||luego se cambia el procedimiento, sin cambiar en nombre
```

```
{set changing-proc = {proc {fahrenheit_in : int}:int
                let k:int = (fahrenheit+273)
                {return k}
        }
}

||se vuelve a llamar al procedimiento

{set kelvin = {changing-proc fahrenheit}}
{value “21 celsius = “ & kelvin & “ grados kelvin”}
```



Los procedimientos anónimos tienen mucho uso por las siguientes razones:

- Se pueden declarar los procedimientos anónimos en cualquier bloque de código o expresión, por ejemplo, en una llamada o como un valor que se guarda en una estructura de datos.

- Los procedimientos anónimos tienen acceso a las variables declaradas dentro del mismo bloque de código en el cual fueron definidos. Además

de todas las variables globales y locales, y aquellas que son pasadas por argumento.

-
- Se utilizan los procedimientos anónimos para sobrecargar los procedimientos.

Ejemplo de pagina en el lenguaje Curl

```
{curl 1.7 applet}
{applet license = "development"}
{document-style  TocDocument}    || el estilo del documento
|#
        este estilo de documento coloca una barra de navegación en el lado
        izquierdo de la página para navegar en los títulos
#|

{heading Los Continentes del Mundo}    ||Encabezado de la página
{heading color="red",
   level= 1, América                   ||primer nivel
}

{paragraph paragraph-justify= "left",      ||la orientación del texto


   paragraph-line-spacing=3,               ||espaciado en el párrafo
   paragraph-first-line-offset=30,         ||sangría en la primera línea
   Los Continentes de América y Europa}

{heading color="green",                    ||coloca el color verde en el título
   level= 2, Guatemala
}
{paragraph paragraph-justify= "left",      ||justificado a la izquierda
   paragraph-line-spacing=3,               ||espaciado del párrafo
   paragraph-first-line-offset=30,
   Guatemala.                              ||Nombre del título
```

```
}
{heading
   level =3, Izabal
}
{paragraph paragraph-justify= "left",
   paragraph-line-spacing=3,
   paragraph-first-line-offset=30,
   El Departamento de Izabal
}
{heading
   level=4, Puerto Barrios
}
{paragraph paragraph-justify= "left",
   paragraph-line-spacing=3,
   paragraph-first-line-offset=30,
   El municipio de Izabal
}
{heading
   level=3, Chiquimula
}
{paragraph paragraph-justify= "left",
   paragraph-line-spacing=3,
   paragraph-first-line-offset=30,
   El Departamento de Chiquimula
}
{heading
   level=4, Chiquimula
}
{paragraph paragraph-justify= "left",
```

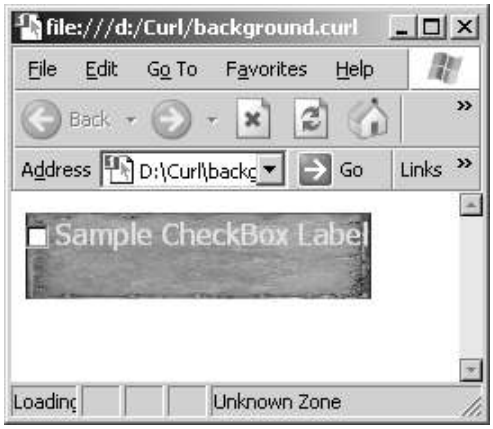
```
    paragraph-line-spacing=3,
    paragraph-first-line-offset=30,
    El municipio de Chiquimula
}

{heading
    level=3, Zacapa
}
{paragraph paragraph-justify= "left",
    paragraph-line-spacing=3,
    paragraph-first-line-offset=30,
    El departamento de Zacapa
}
{heading
    level=4, Zacapa
}
{paragraph paragraph-justify= "left",
    paragraph-line-spacing=3,
    paragraph-first-line-offset=30,
    El municipio de Zacapa
}
{heading color="red",
    level =1,Europa
}
{paragraph paragraph-justify= "left",
    paragraph-line-spacing=3,
    paragraph-first-line-offset=30,
    El continente de Europa
}
```

```
{heading color="green",
   level=2, España
}
{paragraph paragraph-justify= "left",
   paragraph-line-spacing=3,
   paragraph-first-line-offset=30,

   El país de España:
}
{paragraph
   paragraph-justify="left",
   paragraph-line-spacing=3,
   paragraph-first-line-offset=30,
   color="blue",
   En el país de España hay varias ciudades,
   a continuación se muestran unos ejemplos de las ciudades:
}
{itemize                          ||listas
   {item Madrid}                  ||cada uno de los elementos se coloca de esta
   {item Barcelona}               ||manera
   {item Cantabria}
   {item Salamanca}
}

{br}                              ||este es un separador, es una nueva línea
{image source=                    ||se inserta una imagen en la página
   {url "file://D:/CURL/Muestra.jpg" }      ||se especifica la dirección de la imagen
}
```

Aplicaciones de CURL en gráficas

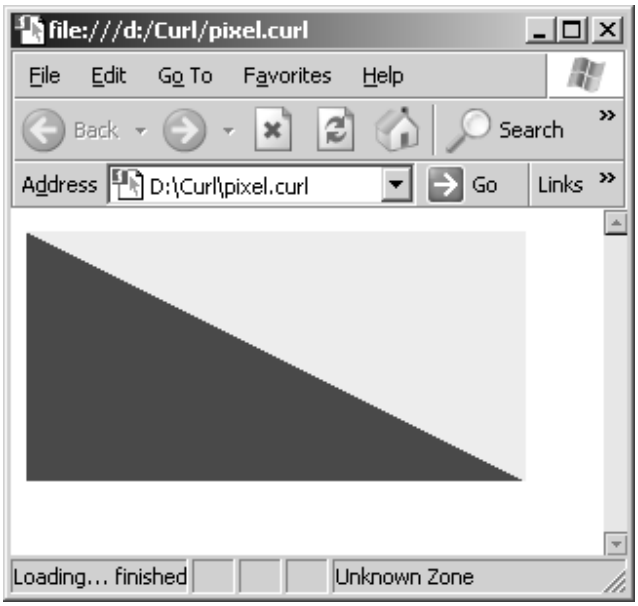
A continuación se mostrarán los aspectos importantes de Curl en el aspecto de las gráficas de 2 dimensiones (2d). Toda la arquitectura del sistema de gráficas 2d puede ser consultado en los archivos de ayuda de Curl. La arquitectura del sistema de gráficas de Curl consiste en varias capas de interfaces creando flexibilidad y complejidad. Las gráficas en Curl tienen unos ejes de referencia (x,y) los cuales tienen su inicio en la parte izquierda superior de la pantalla. A continuación se muestra un ejemplo de gráficas que se pueden desplegar en Curl.

```
|| background.curl
{curl 1.7 applet}
{applet license = "development"}

{let copper-panel= { url "curl://source/Textures/copper-panel.jpg"}
}

{CheckButton
        height = 0.5in,
        label = {text color = "wheat",
                                {big Sample CheckBox Label }
                        },
        background = copper-panel
}
```


file:///d:/Curl/background.curl
File Edit Go To Favorites Help
Back
Address D:\Curl\backg Go Links
Sample CheckBox Label
Loading
Unknown Zone

En Curl existe el manejo de los píxeles. A continuación se muestra un ejemplo de cómo poder manipular los píxeles de la pantalla predeterminada del navegador.

```
|| pixel .curl

{curl 1.7 applet}
{applet license = "development"}

{let sample-pixmap:Pixmap =
        {Pixmap 256, 128,
                initial-value = {{Color.from-rgb 1.0,1.0,0.0}.to-Pixel},
                empty-value = {{Color.from-rgb 1.0,0.0,1.0}.to-Pixel}
        }
}

{for-pixel p at x, y in sample-pixmap do
        {if x < (y*2) then
                set p = sample-pixmap.empty-value
        }
}

{Fill width=256pixel, height=128pixel,
        background = sample-pixmap
}
```


file:///d:/Curl/pixel.curl
File Edit Go To Favorites Help
Back
Search
Address D:\Curl\pixel.curl
Go
Links
Loading... finished
Unknown Zone

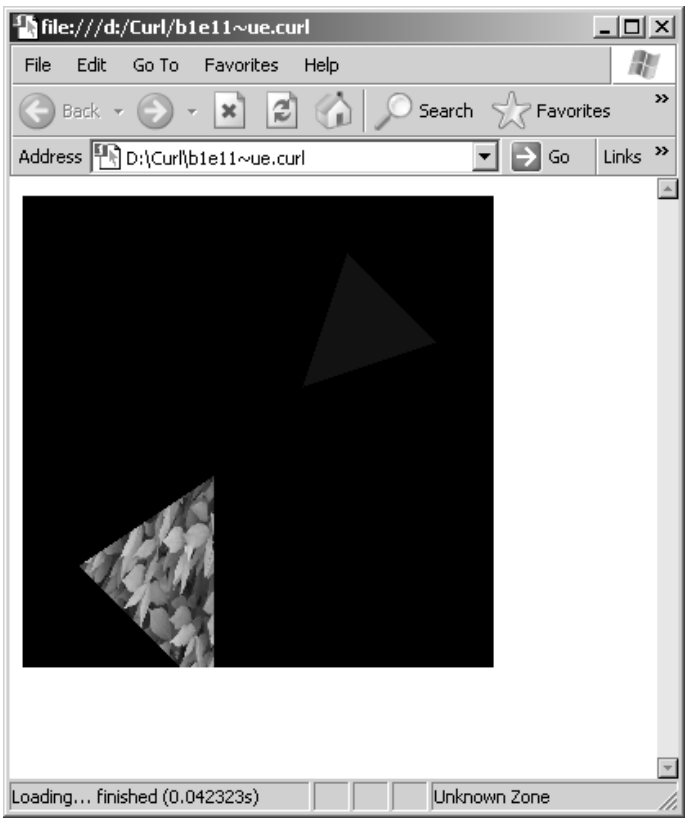
Gráficas de 3D en Curl

La generación de las gráficas en 3D es similar en uso a las gráficas de 2D. El sistema de coordenadas utilizado en las gráficas 3D es similar a las coordenadas de 2D, con la única diferencia que se tiene el sistema de medidas en “z”, las cuales se miden hacia dentro de la pantalla.

La generación de las gráficas 3D que se utilizan en Curl va incluida dentro del código del *applet*. Lo que significa que las gráficas son generadas del lado del cliente. A continuación se muestra un ejemplo de unas gráficas 3D desde Curl.

```
|| triangles.curl

{curl 1.7 applet }
{applet license="development"}

{import * from CURL.GRAPHICS.SCENE}

{SceneGraphic
        width  = 3in,
        height = 3in,
        camera-motion-axis= CameraMotionAxis.none,
        {Scene
                {Triangle
                        {Distance3d 1in,1in,0in},
                        {Distance3d 4in,2in,0in},
                        {Distance3d 2in,4in,0in},
                        fill-pattern = "blue"
```

```
        },
        {Triangle
                {Distance3d -1in,-1in,1in},
                {Distance3d -4in,-3in,1in},
                {Distance3d -1in, -6in,1in},
                fill-pattern = {url "curl://source/Textures/leaves.jpg"}
        }
}}
```



También las gráficas de 3D en Curl pueden interactuar con los eventos generados por el ratón. Para poder realizar esta tarea se le debe de asignar al objeto de la escena un manejador de eventos del ratón. A continuación se muestra un ejemplo de la interacción del ratón con las gráficas 3D de Curl.

Empresas que actualmente utilizan Curl

Las empresas que actualmente utilizan Curl muestran hermetismo en el momento de la investigación, la empresa que actualmente ha implementado Curl como lenguaje de programación es Siemens, empresa alemana de telecomunicaciones (**http://www.siemes.de**). Recientemente (septiembre-2002) otra empresa decidió utilizar Curl como su lenguaje interno de programación, Japan Telecom..