

El Kit de Construcción de Mozilla

Programación XUL

De todas las invenciones de los últimos 50 años, solo unos pocos productos realmente innovadores han llegado a un punto en el cual todos los usamos habitualmente todos los días de nuestra vida. A pesar de ello no tenemos por qué quedarnos estancados en sus defectos iniciales de diseño.

POR JONO BACON



A pesar de que joyas como el Sinclair C5, la moto tipo Chopper, el refresco Tab Clear o Microsoft FoxPro se hayan quedado en el camino, hay algunos productos y tecnologías que se han mantenido. Una de esas tecnologías es la Web. Hay pocas dudas al respecto de que la Web ha causado un gran impacto en la forma en que nos comunicamos, compramos o hacemos

otras cosas. Al margen de las políticas de las guerras de navegadores, la incompetente implementación de estándares y los intentos de censura por parte de ciertos políticos americanos, la Web ha demostrado ser un medio atrayente al que el acceso se supone. Hay veces que puedo oír por ahí la frase “¿Cómo que no tienes Internet?” (enunciada en tono de asombro). Al margen de la popularización de la Web, sus limitaciones son evidentes. La más visible es el hecho de que la interfaz entera debe ser reinventada cada vez que se desarrolla un sitio web. Además, la interfaz debe ser recargada cada vez que se realiza un cambio en la página por muy pequeño que sea. Esto es ineficiente no solo por el hecho de que HTML redundante necesita ser enviado y recibido una y otra vez entre el servidor y el navegador, si no porque

crea un entorno donde los cambios más dinámicos de la página son los más difíciles de realizar, dándole a la Web una clara sensación de “pesada”.

Introducción a XUL

La dependencia de HTML y su esperada funcionalidad de búsqueda es realmente una piedra de toque del problema que acabamos de describir. A pesar de que tecnologías como el HTML Dinámico (DHTML) y el Modelo de Objetos de Documentos (DOM) han aparecido para atacar el problema, éstas necesitan un poco de mimo para hacerlas funcionar conjuntamente. Los desarrolladores del popular buscador Mozilla tuvieron una idea diferente. Con grandes dosis de discusiones y diseño, los programadores trabajaron juntos hasta crear el Lenguaje de Interfaz del Usuario de XML (XUL)

EL AUTOR

Jono Bacon is a writer/journalist, consultant and developer based in England. Jono has been actively involved with Linux since 1998 and has worked on a number of different projects including KDE, KDE::Enterprise, KDE Usability Study, Kafka and Linux UK. You can find his website at <http://www.jonobacon.org>.



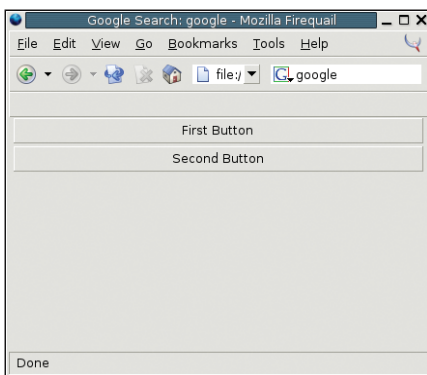


Figura 1: No es exactamente el código XUL más claro, pero es un inicio.

resolviendo el problema parcialmente. Pronunciado “zool” e inspirado por los Cazafantasmas (Ghostbusters), el lenguaje XUL en esencia busca recrear las características de la interfaz de usuario típicamente asociadas a paquetes de herramientas gráficas normales como Qt y GTK. Características como botones, barras de desplazamiento, etiquetas o menús están disponibles en el paquete XUL. Muchos de estos simplemente no están disponibles en formatos normales HTML. Cada una de estas funciones se usa dentro de XUL simplemente escribiendo (no es de extrañar) archivos XML. Para los que sienten un agudo dolor de cabeza cuando leen las letras XML les voy a dar un breve repaso de lo que es. XML es un conjunto de reglas y convenciones que nos permiten crear lenguajes que parecen similares al HTML. Estos lenguajes incluyen etiquetas, atributos, contenidos y otros conceptos que también se hallan en HTML (de hecho las versiones más modernas de HTML son “dialectos” XML). La tecnología XML nos permite crear nuestras propias etiquetas y lenguaje que pueden ser usados para nuestros propios fines. Por ejemplo, si quisiésemos almacenar una dirección en XML crearíamos:

```
<contacto>
<nombre>Pepe</nombre>
<apellidos>López</apellidos>
<direccion>13, Rue del Percebe, 3
33333 Villaalgo</direccion>
<telefono>020 344 5443</telefono>
</contacto>
```

Aquí usamos etiquetas específicas para marcar datos diferentes de información

específica, pudiendo entonces escribir software que lea esas etiquetas y las use en el contexto que deseemos. XUL sigue el mismo concepto, pero las etiquetas son usadas para crear elementos de interfaz específicos. La magia no comienza, no obstante, hasta que Mozilla lee las etiquetas y crea los elementos de la interfaz por nosotros. El archivo XML es una forma simple de especificar lo que queremos como interfaz y donde.

Comencemos

Esta es la primera parte de una serie, donde voy a mostrar como modelar una interfaz basada en XUL. Esta interfaz abarcará algunos de los diferentes componentes XUL disponibles, y si bien no veremos como realizar tareas hasta el próximo capítulo, si sentaremos las bases de cómo crear nuestra interfaz visual. Para comenzar crearemos un archivo XUL simple que contenga 2 botones. Para hacer esto, crearemos un archivo llamado `xul1.xul` al que le añadiremos el siguiente código (ver Listado 1). Tras añadir el código, usaremos *Archivo | Abrir Archivo* para localizarlo y abrirlo en Mozilla. Deberíamos ver algo similar a la figura 1. Cualquier archivo XML, independientemente de su función, debe tener algunas líneas al principio que indican la versión de XML y una hoja de estilo si es oportuno. En nuestro ejemplo tenemos las siguientes 2 líneas:

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://
/global/skin/" type="text/css"?>
```

Como podemos observar, tenemos una línea de versión que indica que la versión XML es la 1.0. La segunda línea indica que nuestra hoja de estilo está en la ruta `chrome`. La ruta `chrome` contiene algunas utilidades internas de Mozilla que gestionan habitualmente las interfaces de usuario de Mozilla. Las siguientes líneas contienen nuestras primeras etiquetas:

```
<window
id="firstwindow"
title="First XUL Window"
xmlns="http://www.mozilla.org/
keymaster/gatekeeper/there.is.only.xul">
```

Si bien tenemos 4 líneas de código, esto es realmente una única etiqueta que he subdividido en una serie de líneas para hacerla más fácil de leer. De hecho, no importa donde hagamos separaciones de línea, siempre y cuando no rompamos etiquetas.

Cada página XUL que creemos necesita una etiqueta `<window>` que pueda ser usadas para contener los componentes que forman nuestra interfaz. Dentro de esta etiqueta hemos usado 3 atributos. El primero (*id*) es una referencia única que apunta a esta etiqueta en XML. El atributo *id* es esencial para ser capaces de comunicarnos con etiquetas y actualizarlas con cambios e información. Esto estará más claro cuando usemos DOM para actualizar y referenciar realmente etiquetas. La segunda etiqueta, *title* (título), contiene una serie de caracteres legibles por humanos que se muestra en la barra de título cuando lanzamos el archivo XUL en la ventana oportuna. Si cargamos el archivo en el navegador como hemos hecho, este texto es ignorado. El último atributo es la parte `xmlns`. Este valor especifica el *namespace* (espacio de nombres) en el que la etiqueta `<window>` y todas las etiquetas dentro de ella están basadas. Un *namespace* es como un grupo especial que podemos especificar para determinar de donde viene una etiqueta. Esto ayuda en situaciones en las que tengamos una etiqueta `<window>` de otro lenguaje XML y una etiqueta `<window>` del lenguaje XUL: *namespace* las diferencia. Ahora estamos listos para poner algo en nuestra pantalla. En nuestro ejemplo hemos creado dos botones con nuestro código:

Listado 1: Xul1.xul

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://
global/skin/" type="text/css"?>
<window id="firstwindow"
title="Primera Ventana XUL"
xmlns="http://www.mozilla.org/
keymaster/gatekeeper/
there.is.only.xul">
<button id="button1"
label="Primer Botón"/>
<button id="button2"
label="Segundo Botón"/>
</window>
```

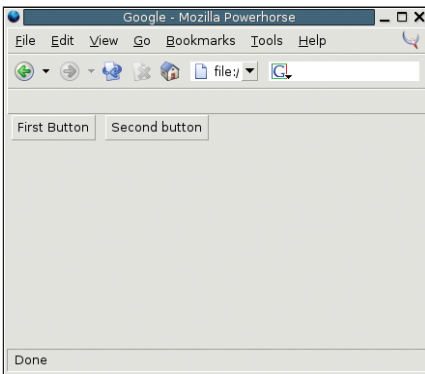


Figura 2: Gestor de diseño horizontal.

```
<button id="button1" label=BB
"Primer Botón"/>
<button id="button2" label=BB
"Segundo Botón"/>
```

Las líneas del código son muy similares, pues solo los atributos *id* y *label* (etiqueta) tienen diferentes contenidos. El atributo *id* se comporta de la misma manera que el equivalente *<window>* que usaremos para referenciar la etiqueta más tarde. El atributo etiqueta contiene el texto que realmente aparece en el Botón. Alguno se preguntará que está haciendo la barra invertida (/) en la etiqueta. Al contrario de lo que ocurre en algunas formas de HTML, donde podemos dejar sueltas etiquetas por ahí, XML es muy estricto sobre el marcaje

correcto. En el caso de nuestras etiquetas *<button>*, se ha de incluir una etiqueta de cierre *</button>* para mantener las reglas de XML. La barra invertida al final de nuestra etiqueta *<button>* es un atajo para incluir la etiqueta *</button>*. Verán que este tipo de atajos se usan habitualmente en XML. Para finalizar nuestro archivo incluimos la etiqueta de cierre *</window>*.

Gestión de la composición

En nuestro primer ejemplo que el segundo Botón está debajo del primero. Este es el comportamiento predeterminado de los componentes cuya distribución no está especificada. Si bien esto es válido para páginas sencillas, este método de posicionar componentes no es suficientemente flexible. Aquí es cuando se necesita usar un gestor de distribución. La gestión de la distribución es algo común en la mayoría de los componentes GUI como Qt y GTK. El único requisito es que pongamos nuestros componentes dentro de unos controles invisibles que nos coloquen nuestro controles visibles de una manera determinada. La mayoría de herramientas contienen formas horizontales y verticales de composición. Esta forma estandarizada de gestionar las formas se ha transferido a XUL y consecuentemente tenemos las etiquetas *<hbox>* y *<vbox>*. Veamos

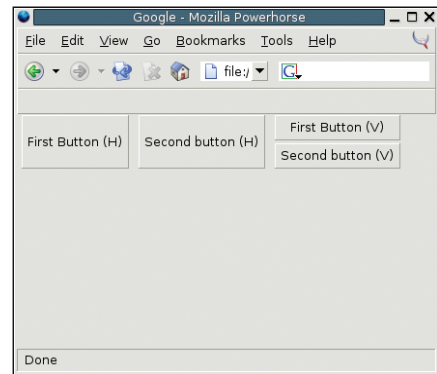


Figura 3: Combinación de gestión del diseño horizontal y vertical.

a continuación un ejemplo de gestión que nos permite disponer nuestros botones:

```
<hbox>
<button id="button1" label=Z
"Primer Botón"/>
<button id="button2" label=Z
"Segundo Botón"/>
</hbox>
```

La forma en que funciona la etiqueta *<hbox>* es colocando horizontalmente los componentes entre las etiquetas *<hbox>* y *</hbox>* más cercanas. Los resultados de este código los podemos ver en la figura 2.

El otro tipo de gestión de la composición es la etiqueta *<vbox>*. Esta etique-

Componentes tipo HTML

Hasta ahora, en nuestra exploración de XUL solo hemos hecho uso de gestores de composición y botones de pulsación. Hay otros muchos componentes que podemos usar, mirando primero los componentes HTML más comunes. Aprenderemos estas etiquetas ejecutando un poco de código:

```
01 <vbox>
02 <hbox>
03 <label value="CuadroBB
04 de verificación"/>
05 <vbox>
06 <checkbox id="check1" label=Z
07 "Primer"/>
08 <checkbox id="check2" label=Z
09 "Segundo"/>
10 </vbox>
11 </hbox>
12 <hbox>
13 <label value="Botones de
```

```
Radio"/>
14 <vbox>
15 <radio id="radio1" label=Z
16 "Primer"/>
17 <radio id="radio2" label=Z
18 "Segundo"/>
19 </vbox>
20 </hbox>
21 <hbox>
22 <label value="Caja de Texto"/>
23 <vbox>
24 <textbox id="textbox"/>
25 </vbox>
26 </hbox>
27 <hbox>
28 <label value="Caja deZ
29 Texto Multilínea"/>
30 <textbox id="multitextbox"Z
31 multiline="true"/>
32 </hbox>
33 </vbox>
```

Con este código estamos componiendo una serie de descripciones de componentes y su correspondiente componentes. Estamos usando la etiqueta *<label>* para indicar texto en nuestra interfaz XUL. Usamos el atributo *<of>* de esta etiqueta para contener el texto que queremos mostrar en la etiqueta.

Nuestro primer tipo de componentes en una cuadro de verificación. La creamos con la etiqueta *<checkbox>* y usamos la etiqueta atributo para indicar el texto al lado de la caja. El segundo componente que usamos es el botón radio, usando la etiqueta *<radio>* de la misma manera que para crear el componente anterior. Ahora creamos una caja de edición de una sola línea. No hay etiqueta asociada a esta caja por lo que simplemente fijamos el atributo *id* dentro de la etiqueta *<textbox>*. Finalmente creamos una caja de texto multilínea. Con este fin simplemente fijamos la variable *multiline = "true"* (verdadero) a una caja de texto normal.

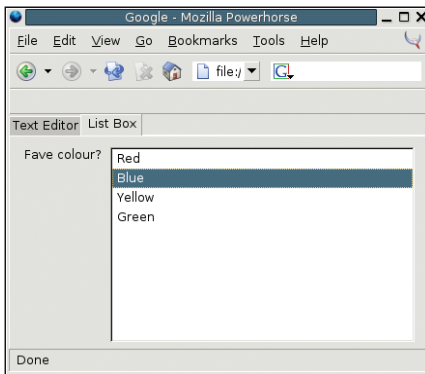


Figura 4: Uso de etiquetas y listados.

ta se comporta de la misma forma que la etiqueta `<hbox>` pero muestra sus componentes hijos (los componentes entre las etiquetas `<vbox>` y `</vbox>`) verticalmente. Si ponemos un bloque similar debajo de nuestro bloque `<hbox>` podemos ver como funciona. (H) en los botones gestionados horizontalmente y (V) en los verticales:

```
01 <hbox>
02 <button id="button1" label="Primer Botón (H)"/>
03 "Primer Botón (H)"/>
04 <button id="button2" label="Segundo Botón (H)"/>
05 "Segundo Botón (H)"/>
06 </hbox>
07 <vbox>
08 <button id="button1" label="Primer Botón (V)"/>
09 "Primer Botón (V)"/>
10 <button id="button2" label="Segundo Botón (V)"/>
11 "Segundo Botón (V)"/>
12 </vbox>
```

Las cosas se ponen realmente interesantes cuando intentamos combinar un tipo de gestor de composiciones dentro de otro gestor. Miremos el siguiente código por ejemplo:

```
01 <hbox>
02 <button id="button1" label="Primer Botón (H)"/>
03 "Primer Botón (H)"/>
04 <button id="button2" label="Segundo Botón (H)"/>
05 "Segundo Botón (H)"/>
06 <vbox>
07 <button id="button1" label="Primer Botón (V)"/>
08 "Primer Botón (V)"/>
09 <button id="button2" label="Segundo Botón (V)"/>
10 "Segundo Botón (V)"/>
11 </vbox>
12 </hbox>
```

Aquí hemos puesto el gestor vertical dentro de los botones gestionados hori-

zontalmente. Prestemos atención a como hemos dispuesto el bloque vertical de botones tras los componentes del bloque horizontal. Debido a la posición de nuestros componentes deberíamos ver algo parecido a lo que muestra la figura 3. Una cosa a la que tenemos que estar atentos cuando posicionemos nuestros componentes es cómo el gestor maneja el espacio. En nuestro último ejemplo se ajustó el ancho de los botones horizontales para acomodar el espacio necesario para los botones verticales. Esto es debido a que el gestor vertical fue anidado dentro del gestor horizontal afectando los componentes horizontales.

Componentes Exóticos

El verdadero poder de XUL reside en la forma en que puede eclipsar a HTML en la forma de conseguir información del usuario. Esta capacidad reside en la forma en que podemos usar aplicaciones de control GUI normales dentro del concepto de la web. En el siguiente ejemplo vamos a crear dos etiquetas, una con un componente de edición de texto multilinea y la otra con un cajetín con una lista desplegable. En este ejemplo, las etiquetas y los cajetines no son típicamente usados en un entorno web. Iremos avanzando paso a paso por este ejemplo y escribiendo el código a medida que progresamos.

Primero creamos un archivo nuevo con la versión de XML, página de estilo y etiquetas `<window>`, para luego añadir las siguientes líneas:

```
<tabbox>
<tabs>
<tab label="Editor Textos"/>
<tab label="Lista"/>
</tabs>
```

Aquí hemos empezado creando un nuevo componente que contiene solapas (`<tabbox>`). Una caja de solapas contendrá un número de solapas que a su vez puede contener otros componentes. Entonces abrimos la etiqueta `<tabs>` para especificar los nombres de las solapas en nuestra interfaz. Para cada solapa usaremos la etiqueta `<tab>` para especificar que etiqueta debe ser definida. Nuestros solapas serán añadidos desde la izquierda hacia la derecha en el orden que los especifiquemos en XUL.

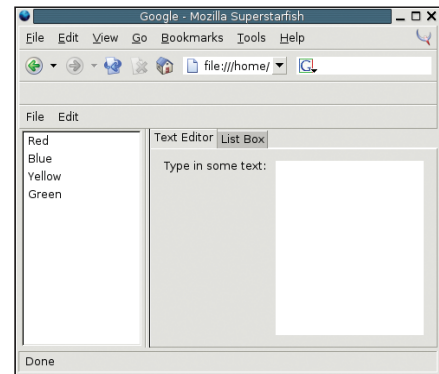


Figura 5: Un interfaz XUL completa.

En este caso, la pestaña "Editor Textos" será la pestaña de la izquierda y la pestaña "Lista" la de la derecha. Ahora tenemos que crear los paneles de las pestañas. Lo hacemos creando primero una etiqueta de paneles de pestañas generales:

```
<tabpanel>
```

Ahora creamos cada panel por turnos. Primero creamos el panel de la caja de texto. Para ello utilizamos la etiqueta `<tabpanel>` para crear cada panel y luego la rellenamos con otros componentes:

```
<tabpanel id="text">
<label value="Escribe un poco de texto:"/>
<textbox id="textbox" multiline="true" flex="1"/>
</tabpanel>
```

El lector avisado se habrá dado cuenta del atributo `flex` que se ha colado en el código. Cuando este tiene un valor de 1 el componente se ajustará para ocupar todo el espacio disponible. Para nuestro segundo panel crearemos nuestra lista de texto dentro del mismo panel. Usaremos la etiqueta `<listbox>` para crear la caja principal y luego usaremos la etiqueta `<listitem>` para añadir cada cosa a la caja.

```
<tabpanel id="listbox">
<label value="¿Color Favorito?"/>
<listbox flex="1">
<listitem label="Rojo"/>
<listitem label="Azul"/>
<listitem label="Amarillo"/>
<listitem label="Verde"/>
```

```
</listbox>
</tabpanel>
```

Finalmente cerramos el panel de etiquetas y la caja de pestañas general:

```
</tabpanel>
</tabbox>
```

Podemos ver el interfaz completa en la figura 4.

Interfaces completos

Para acabar la primera entrega de programación XUL miraremos un interfaz XUL de ejemplo completo. Este interfaz incluirá parte del código que ya hemos repasado al igual que algunos menús y divisores reajustables. Iremos paso a paso por todas las líneas del código para asegurar que entendemos todo lo que hemos tratado. Primero comenzaremos con la definición de etiquetas en XML y la creación de una ventana principal (ver listado 2). Los primeros componentes que añadiremos son algunos menús. Su creación sigue el mismo principio que hemos estado usando con anterioridad creando etiquetas dentro de otras etiquetas para construir los distintos elementos. Primero creamos una barra de menú (la barra en la que el menú se asienta) con una etiqueta `<menubar>`.

```
<menubar id="menubar">
```

Luego añadiremos un menú completo. En este caso, el menú Archivo:

```
01 <menu id="filemenu" >
02 label="Archivo">
03 <menupopup id="file-popup">
```

Listado 2: Ventana principal.

```
01 <?xml version="1.0"?>
02 <?xml-stylesheet
03 href="chrome://global/skin/"
04 type="text/css"?>
05 <window
06 id="complete"
07 title="Ejemplo Completo"
08
09 xmlns="http://www.mozilla.org/
10 keymaster/gatekeeper/there.is.
11 only.xul">
```

```
04 <menuitem label="Nuevo"/>
05 <menuitem label="Abrir"/>
06 <menuitem label="Guardar"/>
07 <menuseparator/>
08 <menuitem label="Salir"/>
09 </menupopup>
10 </menu>
```

Para crear el menú, primero usamos la etiqueta `<menu>` para crear las entradas del mismo y luego usamos `<menupopup>` para crear el área desplegable. Finalmente añadimos una serie de elementos con etiquetas `<menuitem>`. Ahora usamos el mismo concepto para crear el menú *Editar*:

```
<menu id="editmenu" >
label="Editar">
<menupopup id="editpopup">
<menuitem label="Deshacer"/>
<menuitem label="Rehacer"/>
</menupopup>
</menu>
</menubar>
```

Con nuestros menús creados estamos listos para crear el área del interfaz principal. Si miramos la figura 5 podemos observar el resultado de nuestro interfaz y como está construida. Tenemos un listado a la izquierda de la pantalla y las pestañas a la derecha. Para gestionar esta disposición primero necesitamos abrir una etiqueta `<hbox>` y luego crear el listado:

```
<hbox>
<listbox flex="1">
<listitem label="Rojo"/>
<listitem label="Azul"/>
<listitem label="Amarillo"/>
<listitem label="Verde"/>
</listbox>
```

Lo siguiente que vamos a hacer es usar un componente especial llamado *splitter* para añadir una barra reajutable que permita al usuario ajustar el componente a la izquierda y a la derecha del divisor. Usamos la etiqueta `<splitter/>` para crear este componente:

```
<splitter/>
```

El siguiente montón de código es nuestra familiar caja de etiquetas que contienen etiquetas de edición de texto y listados

(ver listado 3). El código no es diferente del ejemplo anterior. Finalmente cerramos el gestor horizontal y la ventana:

```
</hbox>
</window>
```

Conclusión

En la primera parte de nuestra serie hemos cubierto bastante terreno. No solo nos hemos lanzado pateando y gritando al mundo de los programas XML y XUL, si no que además hemos podido estudiar las etiquetas estilo HTML, las etiquetas especiales, los gestores de diseño, cajas de etiquetas, menús y muchas cosas más. Con el conocimiento que hemos desarrollado hasta ahora tenemos la posibilidad de crear interfaces XUL más o menos grandes. Desde luego que hay muchos más componentes que cubriremos en otros números. El mes que viene partiremos del conocimiento de este número y lo haremos funcional. Usaremos las capacidades de JavaScript de Mozilla y las uniremos con XUL para conseguir que nuestras interfaces interactúen con el usuario. ■

Listado 3: Tab box

```
01 <tabbox>
02 <tabs>
03 <tab label="Editor Textos"/>
04 <tab label="Lista"/>
05 </tabs>
06 <tabpanel>
07 <tabpanel id="text">
08 <label value="Escribe algo de
09 texto:"/>
10 <textbox id="textbox"
11 multiline="true" flex="1"/>
12 </tabpanel>
13 <tabpanel id="listbox">
14 <label value="¿Color
15 Favorito?"/>
16 <listbox flex="1">
17 <listitem label="Rojo"/>
18 <listitem label="Azul"/>
19 <listitem label="Amarillo"/>
20 <listitem label="Verde"/>
21 </listbox>
22 </tabpanel>
23 </tabpanel>
24 </tabbox>
```