TRABAJO DE GRADO

LICENCIATURA EN INFORMATICA

TEMA: DESARROLLO DE UN BROWSER OFF LINE

Mayo - 2000

Alumna:
Laura Grandjean

Director:
Lic. Javier Díaz

INDICE

1. BROWSERS OFF-LINE - INTRODUCCION

Los browsers off line son programas clientes que actúan como interfaz entre el usuario e Internet. Muestran páginas web y permiten seguir los links para acceder a otras páginas web sin necesidad de estar conectado a Internet. Las páginas a recorrer son almacenadas en el disco rígido para su posterior navegación.

El browsers off line:

- se contacta con un servidor web y envía un pedido de información
- recibe la información y la almacena en el disco rígido
- muestra la información en el browser que utiliza el usuario

Los browsers off-line permiten navegar en la World Wide Web, sin estar conectado a Internet. Para ello, se baja una porción completa de la WWW al disco rígido. Se le debe indicar una o varias URLs de páginas y el browser off-line seguirá tantos links desde estas páginas como uno le indique. Todas las páginas que encuentre, se bajan al disco, incluyendo los archivos que las componen (gráficos, sonido, etc).

Todos los programas de off-line browsing trabajan esencialmente de la misma manera, es decir, generando un caché propio donde se almacenan las páginas solicitadas previamente en el momento de configuración. Todos ellos, permiten utilizar el browser con el que normalmente opera el usuario.

El objetivo es entonces, visualizar las páginas cuando no se está conectado y que el trabajo de captura de la información, se realice en los horarios cuando la congestión sea menor, para leer la información bajada no sólo en horarios más cómodos, sino a la velocidad de lectura del disco rígido.

Se utilizó como punto de partida un análisis comparativo de distintos browsers off line, para determinar ciertos parámetros a considerar (beneficiosos y perjudiciales) tanto en su funcionalidad como en su interfaz, entre ellos:

2. Grab a Site
3. WebWhacker
4. Go-Get-It
5. Anawave Web Snake
6. NetAttaché Pro

2. ANALISIS DE DISTINTOS BROWSERS OFF LINE

2.1. Grab-A-Site

Nombre:

Grab-A-Site

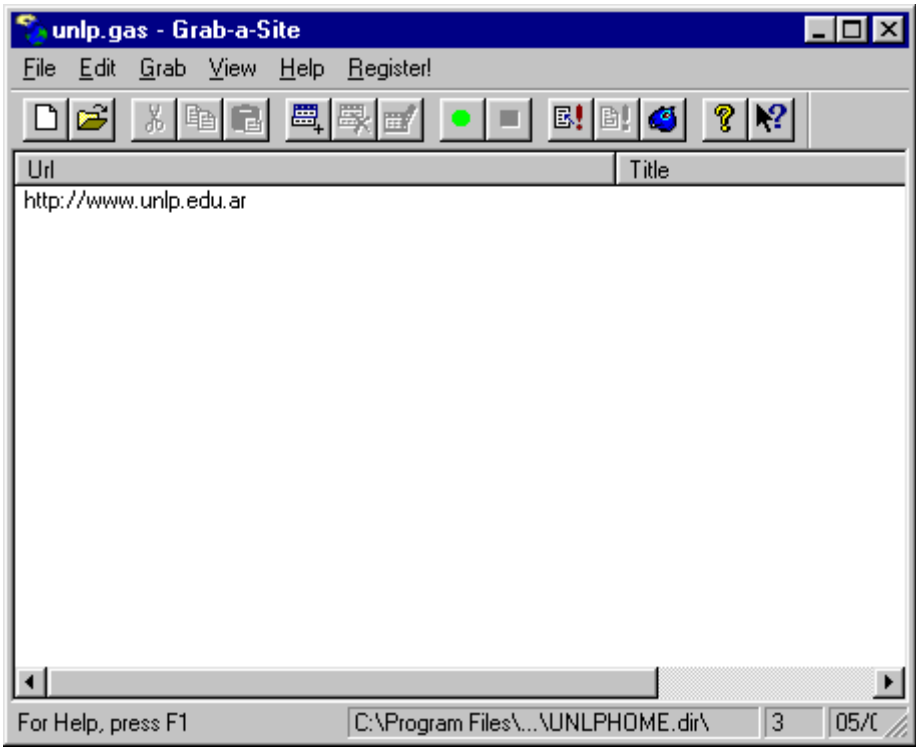
Versión 2.12

Desarrolladores:

ForeFront Group Inc

Lugar donde se encuentra :

http://www.ffg.com



Características :

- Se organiza en grupos. Un grupo contiene una o más URLs
- Simplicidad de manejo
- Permite setear las preferencias: máxima cantidad de bytes bajados, tiempo de búsqueda, documentos grabados, cantidad niveles de links a recorrer, tipo de archivos a bajar, browser a utilizar.

Desventajas :

∇ No permite programar el momento para obtener la información

∇ No posee tutorial ni wizard

∇ No detecta páginas modificadas

∇ Los seteos son generales para todas las URLs del grupo

∇ Disponible sólo para Windows

1.2. WebWhacker

Nombre:

WebWhacker

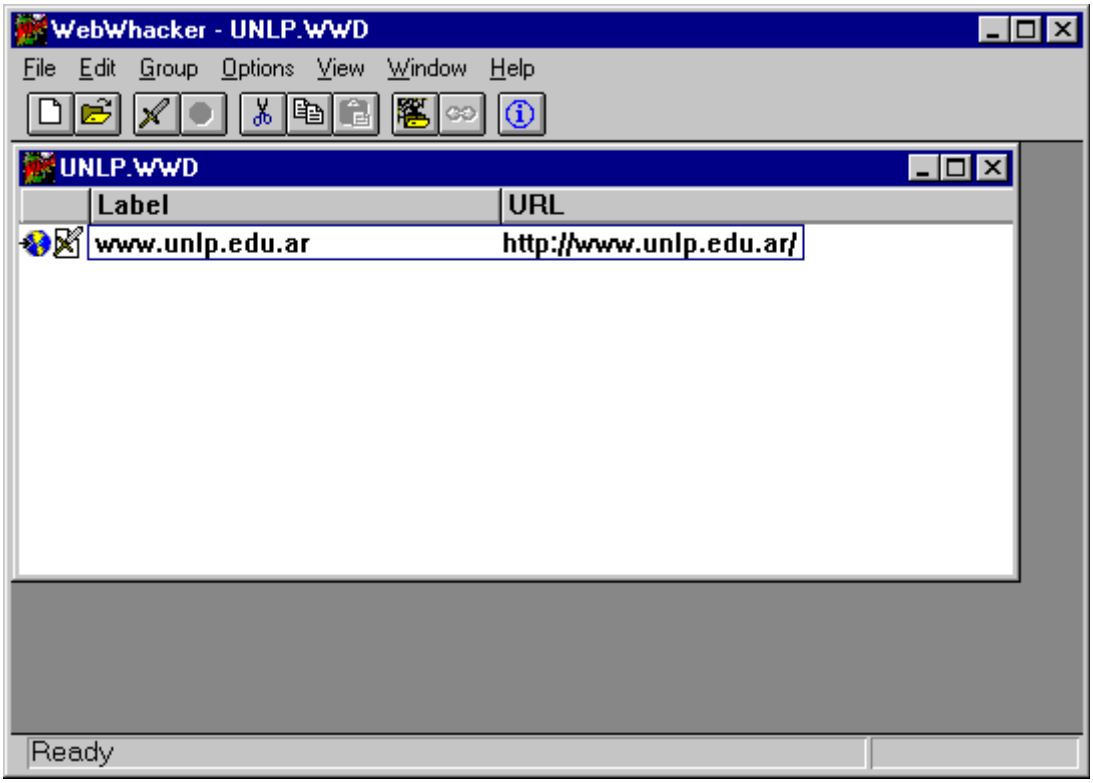
Versión 1.01

Desarrolladores:

ForeFront Group Inc

Lugar donde se encuentra :

http://www.ffg.com



Grab-A-Site y WebWhacker fueron desarrollados por la misma empresa. Si bien son similares en cuanto a su funcionalidad, WebWhacker presenta algunas ventajas que Grab-A-Site no posee.

Características :

- ♦ Se organiza en grupos. Un grupo contiene una o más URLs
- ♦ Para cada URL del grupo permite setear la cantidad de niveles a recorrer
- ♦ Permite setear para el grupo los tipos de archivos y el máximo tamaño de los archivos a bajar
- ♦ Permite programar el momento para obtener la información
- ♦ Detecta páginas modificadas
- ♦ Posee tutorial y wizard

Desventajas :

- ∇ No permite setear la máxima cantidad de archivos a bajar
- ∇ Disponible sólo para Windows y Mac, con aplicaciones diferentes.

2.3 **Go-Get-It**

Nombre:

Go-Get-It

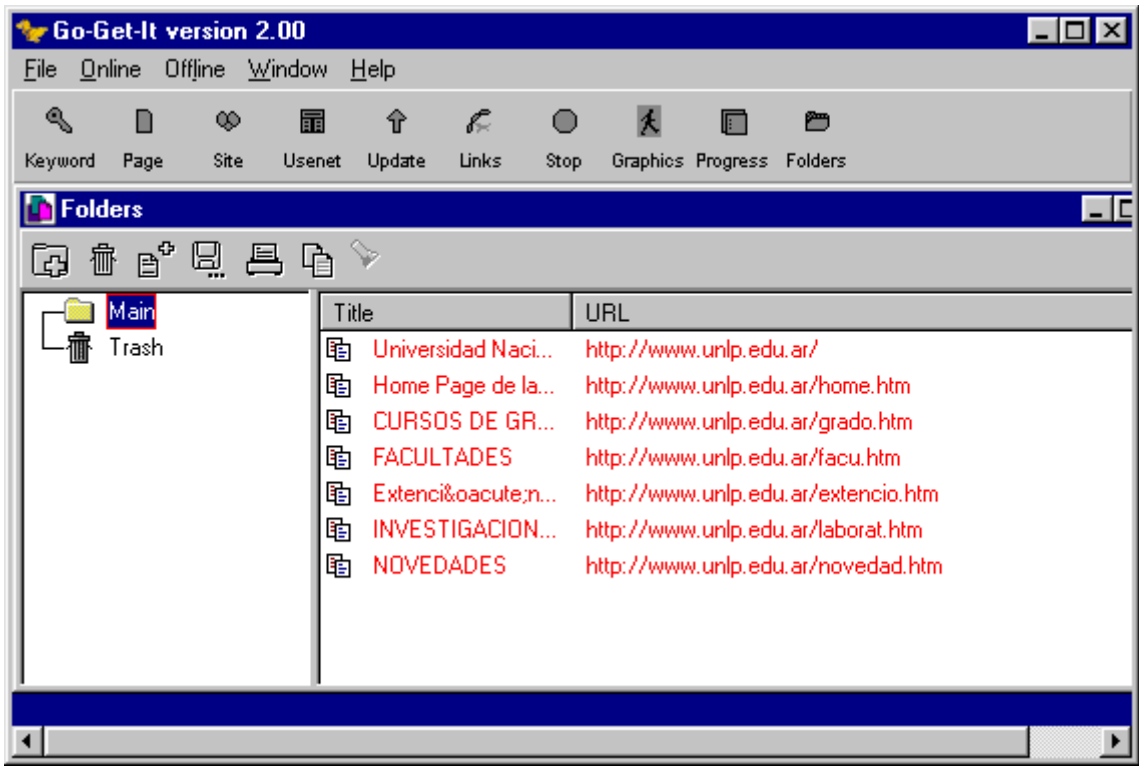
Versión 2.00

Desarrolladores:

Home Page Press, Inc.

Lugar donde se encuentra :

http://www.hpp.com



Características :

- ♦ Se organiza en carpetas . Una carpeta contiene una o más URLs
- ♦ Permite ver las páginas con un browser propio.
- ♦ Detecta páginas modificadas

Desventajas :

- ∇ No permite setear cantidad de niveles de links a recorrer, tipo de archivos a bajar, cantidad de bytes, tiempo.
- ∇ No permite programar el momento para obtener la información
- ∇ Disponible sólo para Windows

2.4. Web Snake

Nombre:

Web Snake

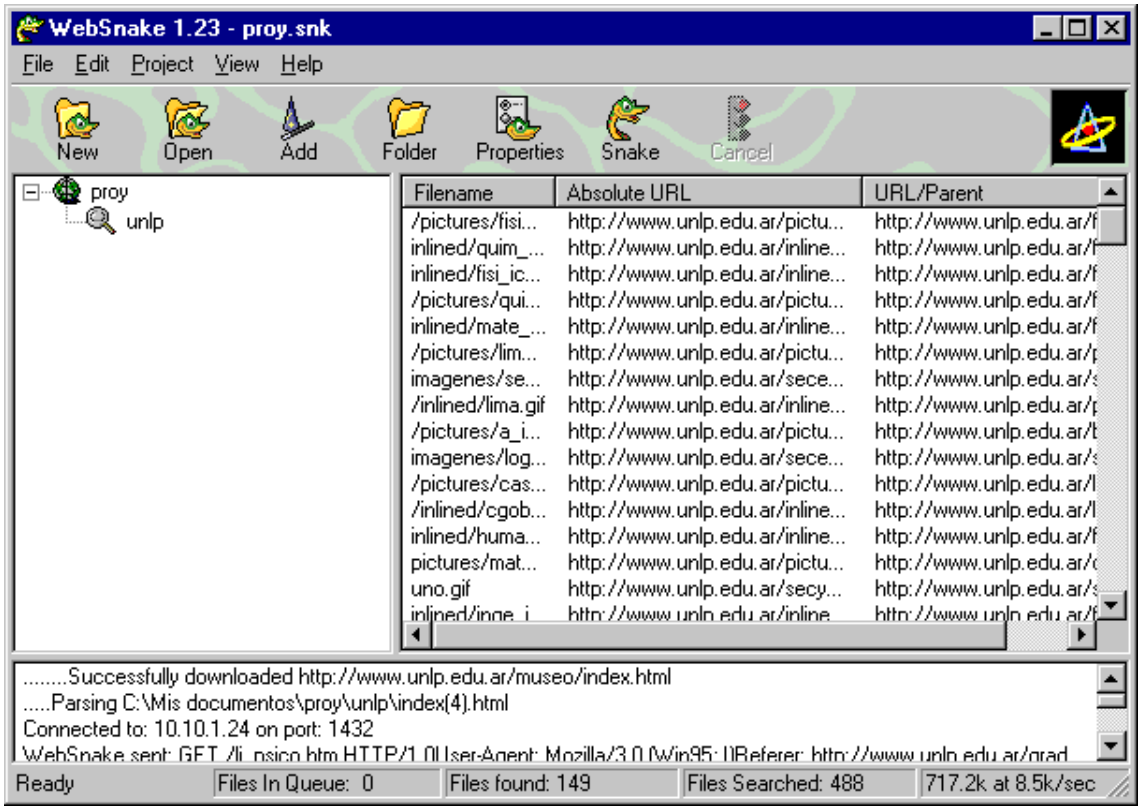
Versión 1.23

Desarrolladores:

Anawave

Lugar donde se encuentra :

http://www.anawave.com



Permite bajar páginas para la navegación off line. Tiene las siguientes características :

- Permite duplicar un sitio (incluyendo la estructura de directorios)
- Se organiza en proyectos y sesiones. Una sesión contiene uno o más proyectos. Un proyecto contiene todo lo necesario para bajar información desde una URL.
- Posee un tutorial que explica paso a paso el proceso para crear una sesión con sus proyectos.
- Permite realizar distintos seteos: máxima cantidad de bytes a bajar, mínimo espacio en disco, cantidad de archivos bajados, no bajar aquellos archivos que excedan una cantidad determinada de bytes, tipo de archivos a bajar y cronograma.
- Posee tutorial y wizard

Desventajas :

∇ No permite bajar desde más de una URL

∇ No es simple el manejo de las sesiones y los proyectos.

∇ No permite programar la fecha y hora para bajar la información

∇ Disponible sólo para Windows

1.5. NetAttache

Nombre:

NetAttache Pro

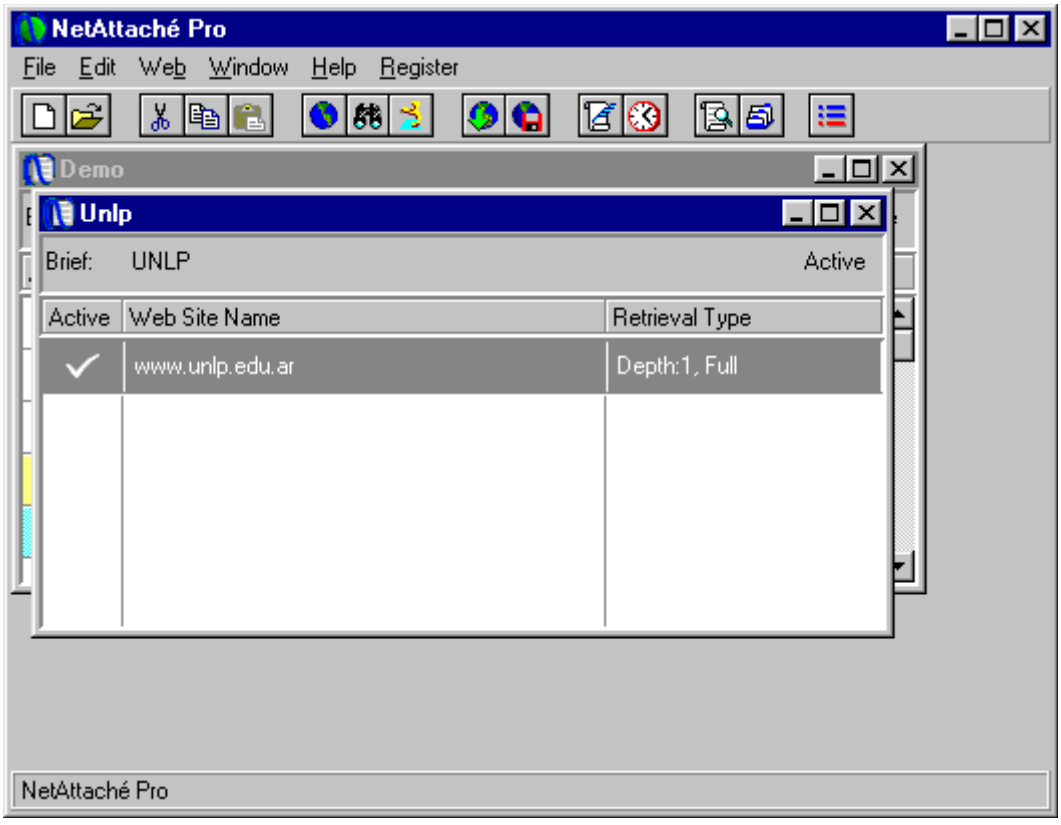
Versión 2.1 b – 16 bit

Desarrolladores:

Tympani Development Inc

Lugar donde se encuentra:

http://www.tympani.com



Características :

- Se organiza en grupos llamados briefs. Un brief contiene una o más URLs
- Permite distinguir diferencias cuando se vuelve a bajar la información.

- Permite setear: mínimo de espacio en disco, tipos de archivos a bajar, máxima cantidad de tiempo, niveles a recorrer.

Desventajas :

∇ No soporta protocolo FTP

∇ Disponible sólo para Windows

3. COMPARACION DE LOS BROWSERS OFF LINE INVESTIGADOS

Característica	Grab-A-Site	Web Whacker	Go Get It	Net Attache	Web Snake
Detecta páginas modificadas	No	Si	Si	Si	Si
Setear tipos de archivos a bajar	Si	Si	No	Si	Si
Setear máxima cantidad de archivos a bajar	Si	No	No	No	Si
Partir desde más de una URL	Si	Si	Si	Si	No
Soporta protocolo FTP	Si	Si	No	No	Si
Soporta múltiples plataformas	No	No	No	No	No
Browser Propio	No	No	Si	No	No
Tutorial	No	Si	No	Si	Si
Wizard	No	Si	No	No	Si

4. OBJETIVO DEL PROYECTO

El objetivo de este proyecto es el desarrollo de una aplicación que permita navegar off line por la World Wide Web, sin necesidad de estar conectado a Internet, haciendo más rápida la navegación.

Características:

- ✓ Organización por grupos de URLs, a partir de las cuales se grabarán las páginas.
- ✓ Seteo de las preferencias de las páginas a recuperar: cantidad máxima de archivos a recuperar, cantidad máxima de bytes a bajar, tipos de archivos, niveles de links a recorrer, opción de grabar URLs con protocolo ftp, opción de grabar archivos de otros dominios, opción de recuperar sólo aquellas páginas que han sido modificadas.
- ✓ Permite recuperar la información en el momento o setear para que se realice automáticamente en un horario dado.
- ✓ Genera un directorio exclusivo donde se almacenan las páginas solicitadas previamente en el momento de configuración.
- ✓ Utiliza el browser con el que opera el usuario por defecto.

Ventajas:

- ✓ Acceder a una página sin necesidad de estar conectado a Internet.
- ✓ Reducir los costos de conexión a Internet (cuando se trata de conexión telefónica)
- ✓ Como las páginas se almacenan en el disco rígido, se puede navegar rápidamente sin los retrasos de una conexión lenta. Se accede a la velocidad del disco rígido. No hay pérdida de tiempo esperando que se baje la información de la red.
- ✓ La información está disponible en todo momento.

5. FUNCIONALIDAD

La aplicación desarrollada permite bajar páginas enteras de la Web o grupos de páginas, incluyendo texto, imágenes y otros objetos, almacenándolos en el disco local. Las páginas bajadas serán relinkeadas localmente para navegar luego sin conexión a Internet. La información grabada se puede ver navegando localmente con cualquier browser.

La información a grabar se organiza en proyectos, los cuales contienen una o más URLs con los seteos correspondientes. Desde estas URLs va a partir nuestra aplicación para empezar la grabación. Una vez que se haya bajado la información, el proyecto contendrá todas las páginas que se bajaron, las imágenes y otros objetos a los que hacen referencia estas páginas.

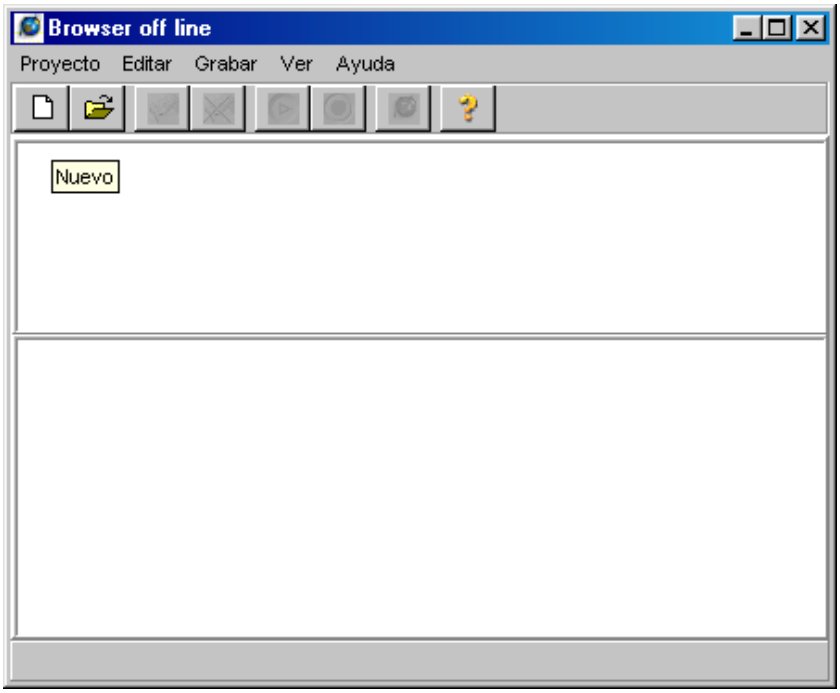
Dentro de los seteos que se pueden realizar están:

- niveles de links a recorrer
- tipos de archivos
- opción de grabar URLs con protocolo ftp
- opción de grabar archivos de otros dominios
- opción de recuperar sólo aquellas páginas que han sido modificadas
- cantidad máxima de bytes a bajar
- cantidad máxima de archivos a bajar (imágenes, sonido, etc.)
- cronograma: hora a la que se quiere bajar la información

Cuando se crea un proyecto, se guarda en el disco rígido un archivo de tipo txt, que contendrá las URLs de las que partirá la grabación y los seteos que el usuario determine. Al grabar, todos los archivos bajados se guardarán en un directorio exclusivo para este proyecto. También se creará un archivo que contendrá todos los archivos bajados, la relación entre éstos, fecha de última grabación.

6. INTERFAZ

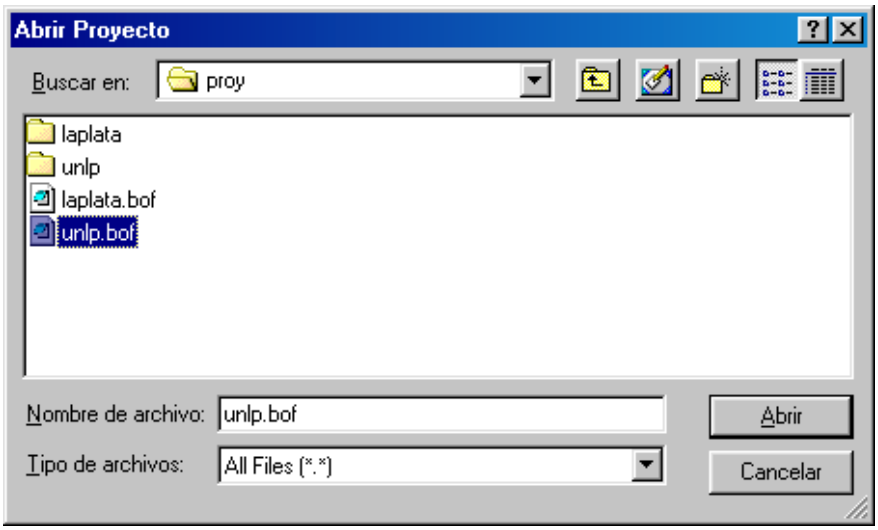
Al ingresar al sistema se despliega la siguiente ventana



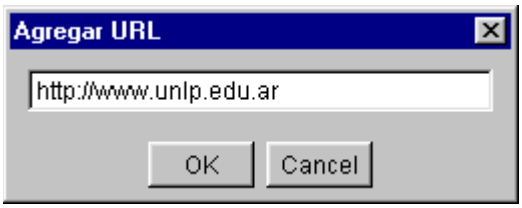
Esta ventana está compuesta por un menú, una barra de herramientas, una barra de estado y dos paneles. El panel superior se utiliza para visualizar la URLs que se van a grabar y el panel inferior para mostrar los archivos que se van grabando a partir de dichas URLs.

Opciones del menú

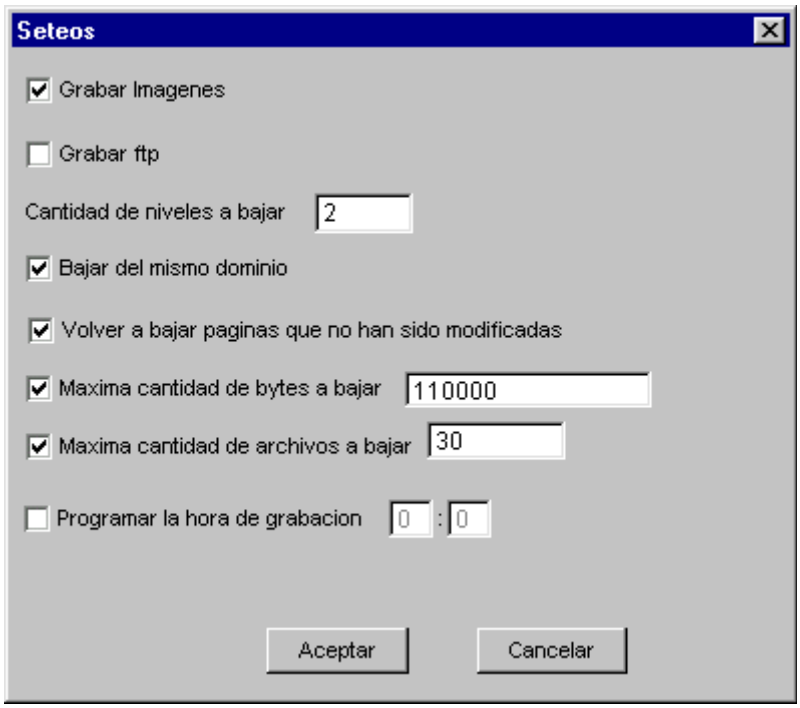
1. Proyecto : Desde este menú, se puede crear y abrir proyectos. Dichas opciones también se encuentran disponibles a través de la barra de herramientas haciendo click en los botones  y  .

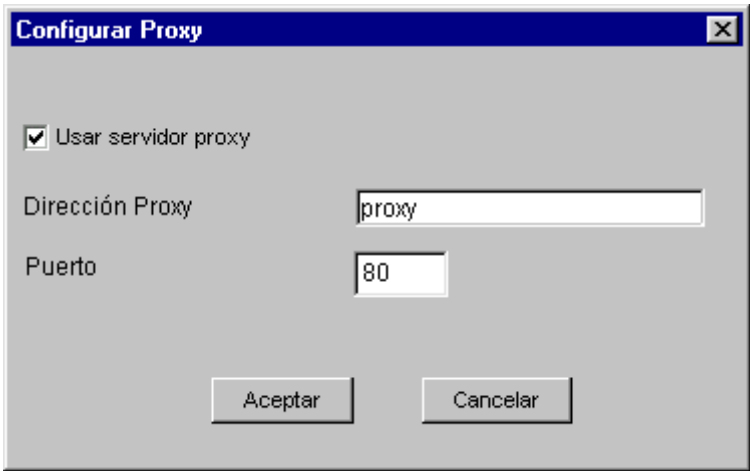


2. Edición : Permite agregar o quitar las URLs a grabar del proyecto. Además se puede acceder a estas opciones haciendo click en los botones de la barra de herramientas  y  .



Se pueden editar los seteos para el proyecto actual y configurar el proxy.





3. Grabar : Permite empezar o detener la grabación. También se puede acceder a estas opciones desde los botones de la barra de herramientas  y  .

4. Ver: permite ver con el browser por defecto las páginas grabadas o haciendo click en el botón de la barra de herramientas  .

5. Ayuda : Contiene un índice con los temas de ayuda, un tutorial y un wizard para manejar un proyecto (quedan como pendientes de desarrollo el tutorial y el wizard).

7. LENGUAJE DE DESARROLLO: JAVA

7.1. Introducción

La aplicación fue desarrollada en Java. Java es un lenguaje de programación que fue diseñado e implementado por un grupo de gente de la empresa Sun Microsystems. Fue presentado en sociedad en Mayo de 1995.

Es de uso general, y su sintaxis y semántica son muy completas para poder abarcar programas de todo tipo. No es necesariamente un lenguaje para Internet, sino que puede ser usado en el desarrollo de un sistema a ejecutarse simplemente en una computadora.

Sun es quien ha desarrollado el lenguaje Java, en un intento de resolver simultáneamente todos los problemas que se le plantean a los desarrolladores de software por la proliferación de arquitecturas incompatibles, tanto entre las diferentes máquinas como entre los diversos sistemas operativos y sistemas de ventanas que funcionaban sobre una misma máquina, añadiendo la dificultad de crear aplicaciones distribuidas en una red como Internet.

7.2. Características

Sun describe al lenguaje Java de la siguiente manera: simple, orientado a objetos, distribuido, robusto, de arquitectura neutral, seguro, portable, interpretado, multihilo y dinámico.

Sun admite totalmente que lo dicho anteriormente es una cadena de halagos por parte suya, pero el hecho es que todo ello describe al lenguaje Java.

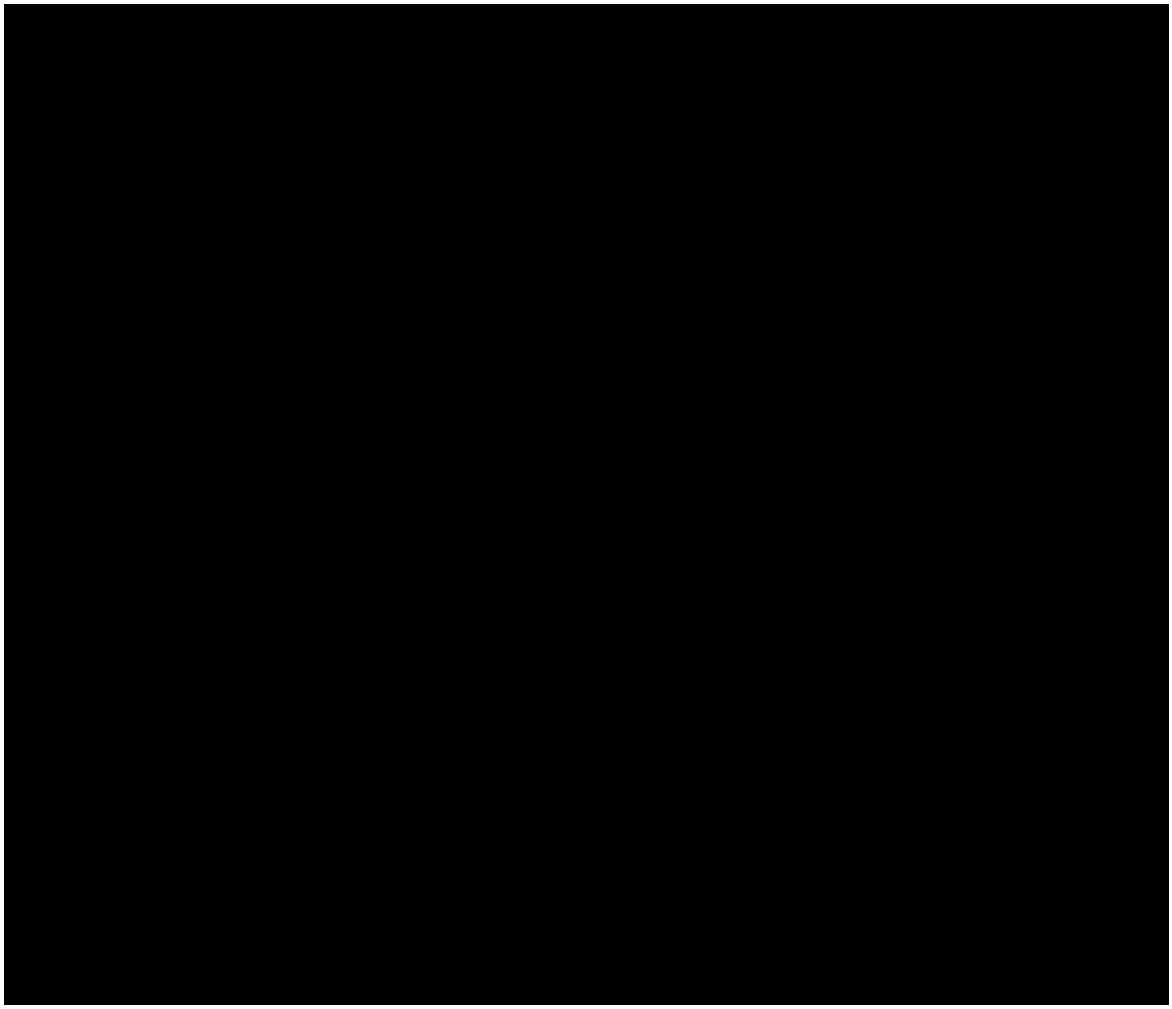
- ✓ SIMPLE: Java ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos. Por ej. no es necesario preocuparse de liberar memoria, tiene un garbage collector (reciclador de memoria dinámica) que se encarga de ello y como es un hilo de baja prioridad, cuando entra en acción, permite liberar bloques de memoria muy grandes, lo que reduce la fragmentación de la memoria.
- ✓ ORIENTADO A OBJETOS: La calificación de lenguaje orientado a objetos ha sido usada en muchos productos de computación, pero nunca tan bien aplicada como en Java, debido a que es realmente un lenguaje donde todo es un objeto. Java trabaja

con sus datos como objetos y con interfaces a esos objetos. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo.

- ✓ DISTRIBUIDO: Java se ha construido con extensas capacidades de interconexión TCP/IP. Existen librerías de rutinas para acceder e interactuar con protocolos como http y ftp. Esto permite a los programadores acceder a la información a través de la red con tanta facilidad como a los archivos locales.
- ✓ ROBUSTO: Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en Java ayuda a detectar errores, lo antes posible, en el ciclo de desarrollo. Java obliga a la declaración explícita de métodos, reduciendo así las posibilidades de error. Maneja la memoria para eliminar las preocupaciones por parte del programador de la liberación o corrupción de memoria.

  También implementa los arrays auténticos, en vez de listas enlazadas de punteros, con comprobación de límites, para evitar la posibilidad de sobreescribir o corromper memoria resultado de punteros que señalan a zonas equivocadas. Estas características reducen drásticamente el tiempo de desarrollo de aplicaciones en Java.
- ✓ ARQUITECTURA NEUTRAL: Para establecer Java como parte integral de la red, el compilador Java compila su código a un archivo objeto de formato independiente de la arquitectura de la máquina en que se ejecutará. Cualquier máquina que tenga el sistema de ejecución (run-time) puede ejecutar ese código objeto, sin importar en modo alguno la máquina en que ha sido generado. Actualmente existen sistemas run-time para Solaris 2.x, SunOs 4.1.x, Windows 95, Windows NT, Linux, Irix, Aix, Mac y Apple.

  Java es un lenguaje que puede ser ejecutado en múltiples plataformas. Es uno de los escasos lenguajes cuyos programas pueden ser transportados de sistema operativo, computadora o entorno, sin necesidad de cambiar el código. Esta característica ha sido la que alimentó su auge en Internet: una red que pueda ser accedida desde cualquier máquina, debe apelar a tecnologías multiplataformas. Java ha sido concebido, desde sus orígenes, como un lenguaje capaz de producir código totalmente transportable.



El código fuente Java se "compila" a un código de bytes de alto nivel independiente de la máquina. Este código (byte-codes) está diseñado para ejecutarse en una máquina hipotética que es implementada por un sistema run-time, que sí es dependiente de la máquina.

- ✓ SEGURO: En el lenguaje, características como los punteros o el casting implícito que hacen los compiladores de C y C++ se eliminan para prevenir el acceso ilegal a la memoria.

Java imposibilita, también, abrir un archivo de la máquina local (siempre que se realizan operaciones con archivos, éstas trabajan sobre el disco duro de la máquina de donde partió el applet).

Se puede decir que Java es un lenguaje seguro y que los applets están libres de virus.

Respecto a la seguridad del código fuente, no ya del lenguaje, JDK proporciona un desemsamblador de byte-code, que permite que cualquier programa pueda ser convertido a código fuente, lo que para el programador significa una vulnerabilidad total a su código. Utilizando javap no se obtiene el código fuente original, pero sí desmonta el programa mostrando el algoritmo que se utiliza, que es lo realmente interesante. La protección de los programadores ante esto es utilizar llamadas a programas nativos, externos (incluso en C o C++) de forma que no sea descompilable todo el código; aunque así se pierda portabilidad. Esta es otra de las cuestiones que Java tiene pendientes.

- ✓ PORTABLE: Más allá de la portabilidad básica por ser de arquitectura independiente, Java implementa otros estándares de portabilidad para facilitar el desarrollo. Los enteros son siempre enteros y además, enteros de 32 bits en complemento a 2. Además, Java construye sus interfaces de usuario a través de un sistema abstracto de ventanas de forma que las ventanas puedan ser implantadas en entornos Unix, Pc o Mac.
- ✓ INTERPRETADO: El intérprete Java (sistema run-time) puede ejecutar directamente el código objeto. Enlazar (linkear) un programa, normalmente, consume menos recursos que compilarlo. No obstante, el compilador actual del JDK es bastante lento. Por ahora, que todavía no hay compiladores específicos de Java para las diversas plataformas, Java es más lento que otros lenguajes de programación, como C++, ya que debe ser interpretado y no ejecutado como sucede en cualquier programa tradicional.
- ✓ MULTIHILO: Al ser multithread (multihilo), Java permite muchas actividades simultáneas en un programa. Los threads son básicamente pequeños procesos o piezas independientes de un gran proceso. Al estar los threads contruidos en el lenguaje, son más fáciles de usar y más robustos.

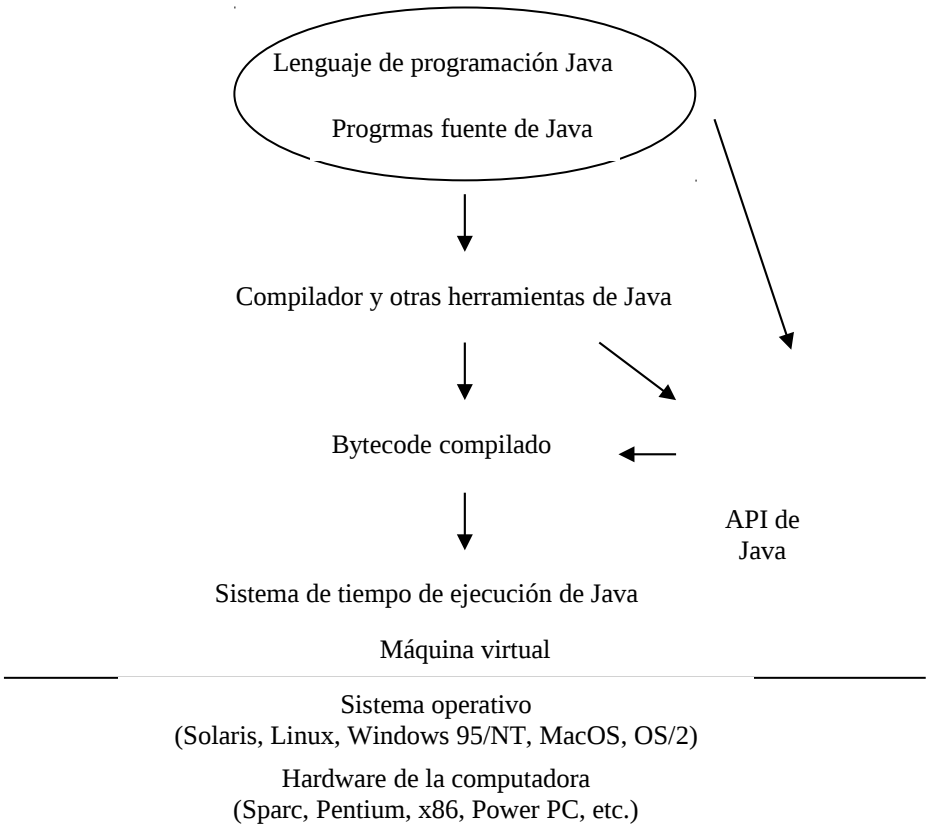
  El beneficio de ser multithread consiste en un mejor rendimiento interactivo y mejor comportamiento en tiempo real. Aunque el comportamiento en tiempo real está limitado a las capacidades del sistema operativo subyacente (Unix, Windows,

etc.), aún supera a los entornos de flujo único de programa (single-thread) tanto en facilidad de desarrollo como en rendimiento.

- ✓ DINAMICO: Java se beneficia todo lo posible de la tecnología orientada a objetos. Java no intenta conectar todos los módulos que comprenden una aplicación hasta el tiempo de ejecución. Las librerías nuevas o actualizadas no paralizarán las aplicaciones actuales (siempre que mantengan la API anterior).

Java también simplifica el uso de protocolos nuevos o actualizados. Si su sistema ejecuta una aplicación Java sobre la red y encuentra una pieza de la aplicación que no sabe manejar, Java es capaz de traer automáticamente cualquiera de esas piezas que el sistema necesita para funcionar.

6.3. La API de Java



El desarrollador de software escribe programas en el lenguaje Java que emplean paquetes de software predefinidos de la interfaz para programación de aplicaciones de Java (API). Luego compila sus programas mediante el compilador Java y el resultado de todo ello es lo que se denomina un bytecode compilado. El bytecode está en una forma que puede ser ejecutado en la máquina virtual, que es el núcleo del sistema de tiempo de ejecución Java. Esta máquina virtual funciona como un microprocesador implementado en software que funciona utilizando las capacidades que le prestan su sistema operativo y el hardware de su computadora. El sistema de tiempo de ejecución de Java se compone de la máquina virtual y del software adicional, como por ejemplo bibliotecas de enlace dinámico, necesarios para implementar la API de Java en su hardware y en su sistema operativo.

La API de Java contiene paquetes predefinidos de software dotados de numerosos enlaces independientes de la plataforma proyectados hacia las potencialidades nativas

de trabajo en ventanas y redes del sistema operativo. La API de Java proporciona una única API común para todos los sistemas operativos a los que es portado Java.

Las claves de la portabilidad de Java son su sistema de tiempo de ejecución y su API. El primero es sumamente compacto, pues deriva de anteriores esfuerzos de la empresa Sun para crear una plataforma de software para aparatos electrónicos de consumo. Esta plataforma no se diseñó a partir de un microprocesador ya existente, sino que arrancó de cero con el propósito de ser simple y eficaz. El hecho de no estar ligada a una determinada arquitectura de hardware le permitió desarrollar una arquitectura neutra. La naturaleza simple, eficaz, compacta y de arquitectura neutra del sistema de tiempo de ejecución es lo que le confiere su gran portabilidad y le hace alcanzar unas prestaciones notables.

Los potentes recursos para el trabajo en ventanas y redes incluidos en la API de Java facilitan a los programadores el desarrollo de un software que resulte, a la vez, atractivo e independiente de cualquier plataforma.

7.4. Por qué programar en Java ?

Java fue diseñado con múltiples niveles de seguridad incorporadas en el compilador, en el sistema de tiempo de ejecución y en los navegadores que admiten Java. Estas medidas hacen que Java sea de por sí mucho más seguro a la hora de crear cualquier tipo de software de aplicación fiable.

El soporte para trabajo en redes que incorpora Java, así como la capacidad de sus sistema de tiempo de ejecución para cargar dinámicamente bytecode Java por medio de la red, lo hacen especialmente adecuado para las aplicaciones distribuidas de conexión en red. Proporciona la capacidad de utilizar dinámicamente nuevos contenidos y software de gestión de protocolos.

La elección de Java resulta singularmente acertado cuando lo que importa por encima de todo es la fiabilidad del lenguaje, como es el caso en las aplicaciones de naturaleza crítica. Su carácter orientado al objeto, combinado con sus numerosas comprobaciones de la integridad del tiempo de compilación y de ejecución, eliminan de raíz muchos errores de programación difíciles de detectar. Por otra parte, el lenguaje Java prescinde por completo de muchas posibilidades de programación peligrosas, tales como los punteros modificables, la conversión de tipo no comprobada y la comprobación débil de

límites, que suelen permitirse en otros lenguajes de programación, como en C y C++, por ejemplo.

La API de Java proporciona pleno soporte a la programación multihilo. Los programas multihilo pueden ser desarrollados de una forma única y consistente, sin depender de las particularidades de la interfaz del sistema operativos principal.

Las clases y objetos Java admiten directamente los conceptos orientados a objetos de encapsulación, herencia, mensajes y métodos, y de ocultación de datos. Las interfaces Java admiten la herencia múltiple y el polimorfismo. El lenguaje Java retiene, en suma, todas las ventajas de programación oritentada a objetos.

La API de Java presta también un amplio soporte al desarrollo de la interfaz gráfica del usuario y al trabajo en ventanas, sin las complicaciones inherentes al mantenimiento de múltiples bibliotecas de clase de ventanas.

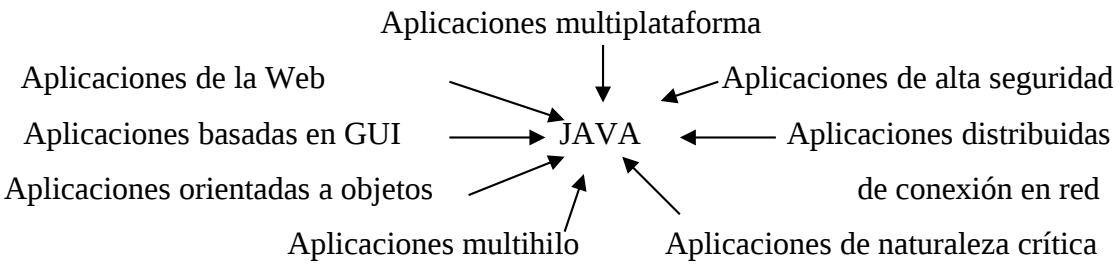
Los programas Java se dividen en dos categorías: applets y aplicaciones.

Las applets son programas Java que se almacenan en un servidor de Internet, se ejecutan desde un navegador y son descargados automáticamente como parte de una página Web, al igual que cualquier gráfico, cuando se activa ejecuta un programa.

Las applets deben ser lanzadas desde una página web.

Las aplicaciones son programas independientes, no requieren de un navegador para ejecutarse. Java puede ser usado para crear cualquier tipo de aplicaciones que se podría hacer con un lenguaje de programación convencional. Ej. el navegador HotJava es una aplicación Java.

El lenguaje cuenta, además, con varias herramientas de programación visuales creadas especialmente para Java.



7.5. Herramientas para el desarrollo de programas investigadas

7.5.1 JDK 1.1.3 y 1.1.6

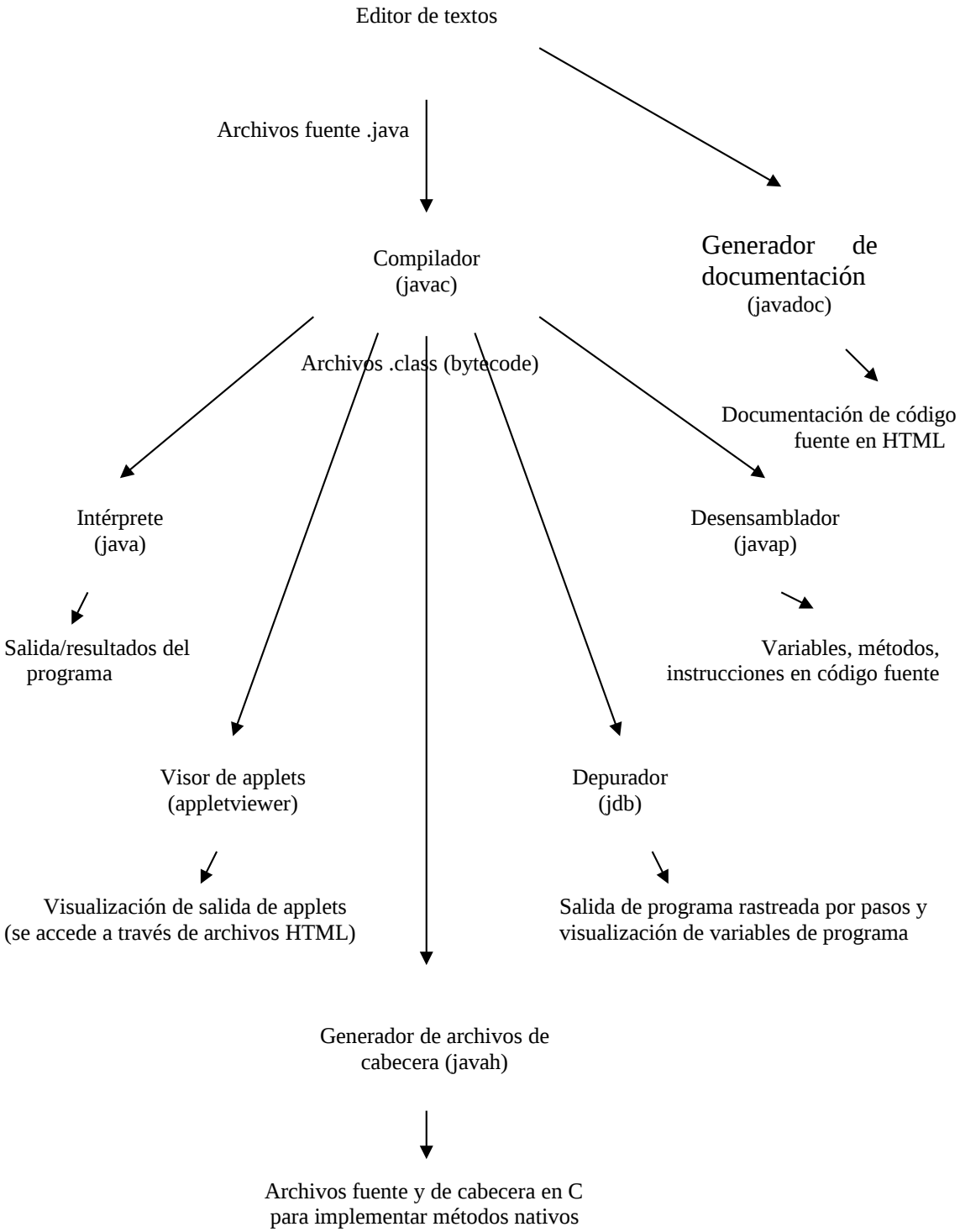
El JDK (Java Devolopment Kit) es un conjunto de herramienta provistas por Sun, para el desarrollo de Java. Se puede bajar libremente de http://java.sun.com/products/jdk/. Todo el JDK se ejecuta desde el DOS. Si bien las herramientas son gratuitas, son un tanto primitivas. No encontraremos ambientes de desarrollo gráfico, ni ventanas amigables. Simplemente, comandos a ejecutar desde el prompt del DOS

El JDK, es un conjunto completo de herramientas para desarrollar, probar, documentar y ejecutar programas java. Consta de los siguientes componentes:

- ✓ **javac.exe:** es el compilador del lenguaje Java. Convierte los archivos fuente (con extensión .java) en archivos que puedan ser ejecutados mediante el intérprete java. Estos archivos se denominan archivos bytecode y concluyen con la extensión .class. El compilador javac ha sido escrito en el mismo Java.
- ✓ **java.exe:** es el intérprete de Java, ejecuta bytecodes creados por javac, el compilador de Java. Verifica la integridad, el funcionamiento correcto y la seguridad de cada clase a medida que es cargada y ejecutada, e interacciona con el sistema operativo principal, el entorno de ventanas y los recursos de comunicación para conseguir el comportamiento deseado del programa.
- ✓ **appletviewer.exe:** es el visor de applets de Java. Muesta las applets contenidas en las páginas web, ubicadas en el sistema de archivos local o en ubicaciones de la web accesibles. Se utiliza también para probar las applets desarrolladas por uno mismo.
- ✓ **jdb.exe:** es un depurador para clases de Java. Tiene en común con el intérprete el hecho de ejecutar clases Java compiladas en archivos bytecode, pero ofrece, además, otras posibilidades especiales, como las de detener la ejecución de un programa en puntos de interrupción seleccionados y mostrar los valores de las variables de clase. Estas posibilidades resultan muy útiles para localizar eventuales errores de programación.
- ✓ **javadoc.exe:** es el generador de documentación automatizada. Se emplea para convertir porciones de archivos fuente Java en archivos HTML. Los archivos HTML generados por javadoc documentan clases, variables, métodos, interfaces y excepciones contenidas en archivos fuente Java a partir de comentarios especiales insertados en esos archivos.

- ✓ **javah.exe:** genera archivos de cabecera en C. Se utilizan para generar archivos de cabecera en lenguaje C y archivos fuente a partir de un archivo de bytecode Java. Los archivos generados por javah sirven para desarrollar métodos nativos, es decir, clases Java escritas en lenguajes distintos del Java.
- ✓ **javap.exe:** desensambla de clases. Toma los archivos bytecode y muestra las clases, los campos (variables) y los métodos compilados en los bytecodes. Identifica también las intrucciones en bytecode empleadas para implementar cada método. El desensamblador es una herramienta muy útil para recuperar el diseño del código fuente de aquellas clases compiladas Java para las que no se dispone de ningún código fuente.

La siguiente figura muestra como interactúan los programas del JDK



El entorno habitual pues, consiste en un compilador que convierta el código fuente Java a bytecode, el intérprete Java para ejecutar los programas y un navegador que pueda ejecutar applets. Estos son los componentes básicos para desarrollar algo en Java. No obstante se necesita un editor para escribir el código fuente.

7.5.2 MICROSOFT VISUAL J++ 1.1

Si bien el JDK de Sun tiene todo lo necesario para crear aplicaciones y applets Java, debemos admitir que es algo primitivo. No hay interfaces gráficas, ni entornos de desarrollo.

Microsoft cubrió estas falencias y produjo un entorno llamado Visual J++. Microsoft ha afirmado su compromiso con Java, produciendo, entre otras cosas, varios SDK (Software Development Kit) para Java, una máquina virtual Java que intenta integrar en el sistema operativo, y el conjunto de clases AFC (Application Foundation Classes), escritas totalmente en Java. Visual J++ 1.1 está basado en el JDK 1.0

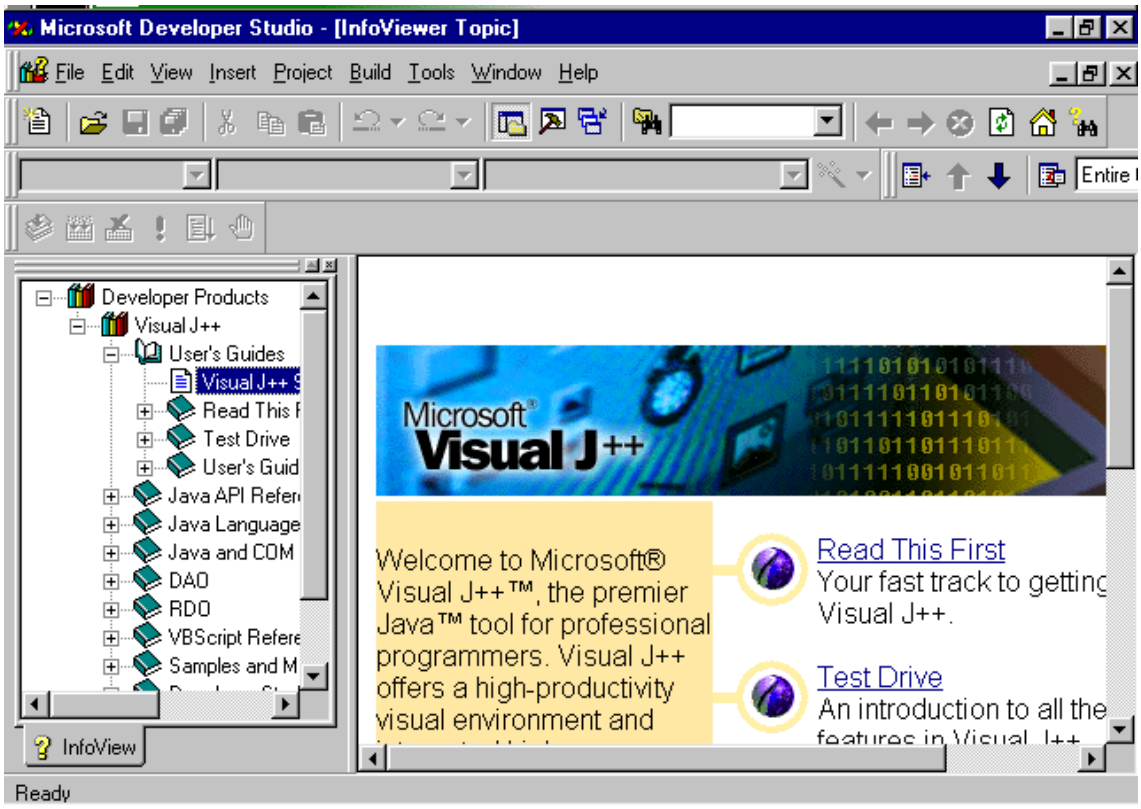
El producto Visual J++ se destaca por su entorno gráfico. Tal es el compromiso de Microsoft con Java, que incorporó el Visual J++ en su versión del Visual Studio 97, su entorno de desarrollo en múltiples lenguajes.

Para poder ejecutarse necesita como mínimo una máquina Pentium con Windows 95 o NT 4.0 o superior y un mímimo de 8 MB de memoria.

Al ejecutar el programa, podremos encontrarnos con el entorno de desarrollo adoptado por Micrososft para su Visual Studio 97, las herremientas de la línea (Visual C++, Visual J++, Visual Interdev) se instalan sobre esta interface, que comparten entre todos. Cada vez que se instala uno de esos programas, en realidad, se agrega a esta interface.

COMPONENTES

La siguiente pantalla es la principal del Developer Studio, la interface donde queda instalado el Visual J++.



El Devoloper Studio tiene varias áreas, de las que se destacan: un área a la izquierda, con distintos contenidos; cuando no hay ningún proyecto en desarrollo, sólo aparece un panel de información. En éste hay un árbol que sirve de índice para todos los temas.

Contiene una guía del usuario muy completa y una descripción de todos los paquetes que componen las clases predefinidas de java. Es una información abundante, con detalle de cada clase, interface, método o atributo.

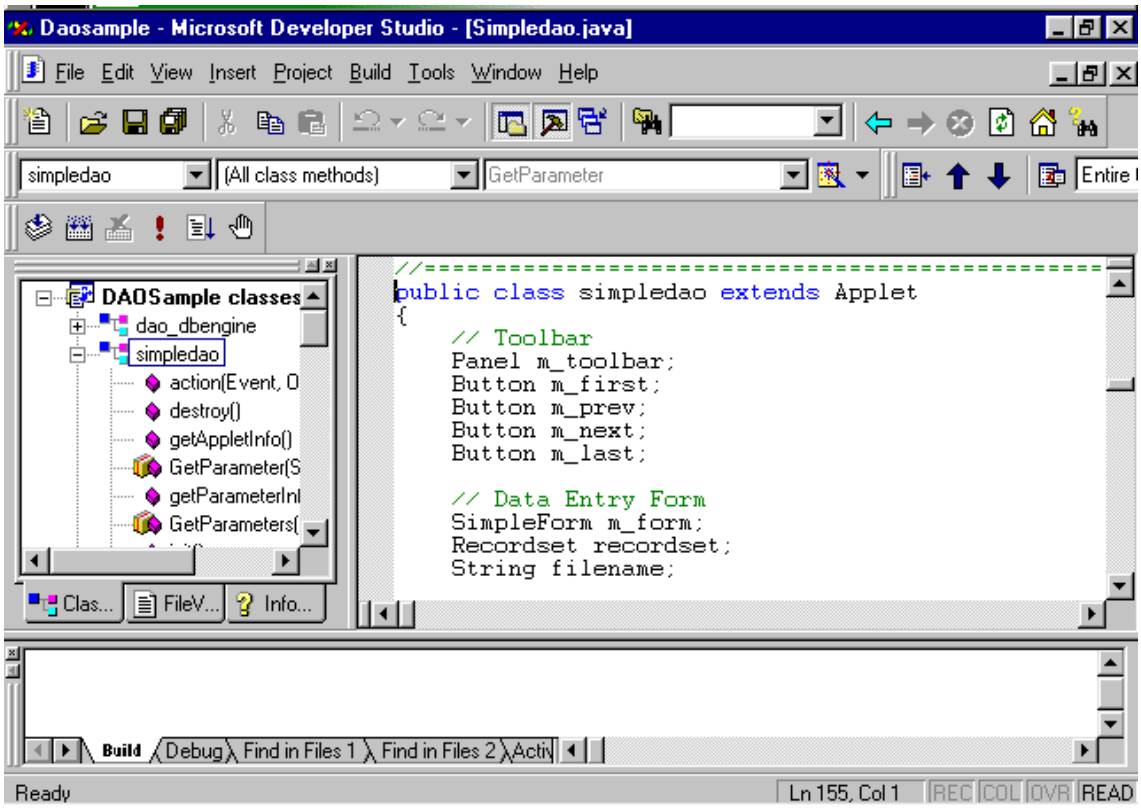
ESPACIOS DE TRABAJO Y PROYECTOS

Visual J++ maneja el concepto de espacio de trabajo (WorkSpace). Un espacio de trabajo es, en general, un proyecto. Este es un concepto de Visual J++ y de los modernos ambientes de trabajo.

Un proyecto es el conjunto de programas y recursos que arman el sistema en el que estamos trabajando. Un proyecto Java podrá componerse de los distintos archivos de programa, archivos de gráficos y de sonido, y páginas HTML para invocar a la applet que estamos armando.

Un espacio de trabajo puede estar compuesto de uno o más proyectos. Cada proyecto puede pertenecer a distintas categorías: uno podría ser un aplicación Java, y otro el desarrollo de un control ActiveX desde Visual C++.

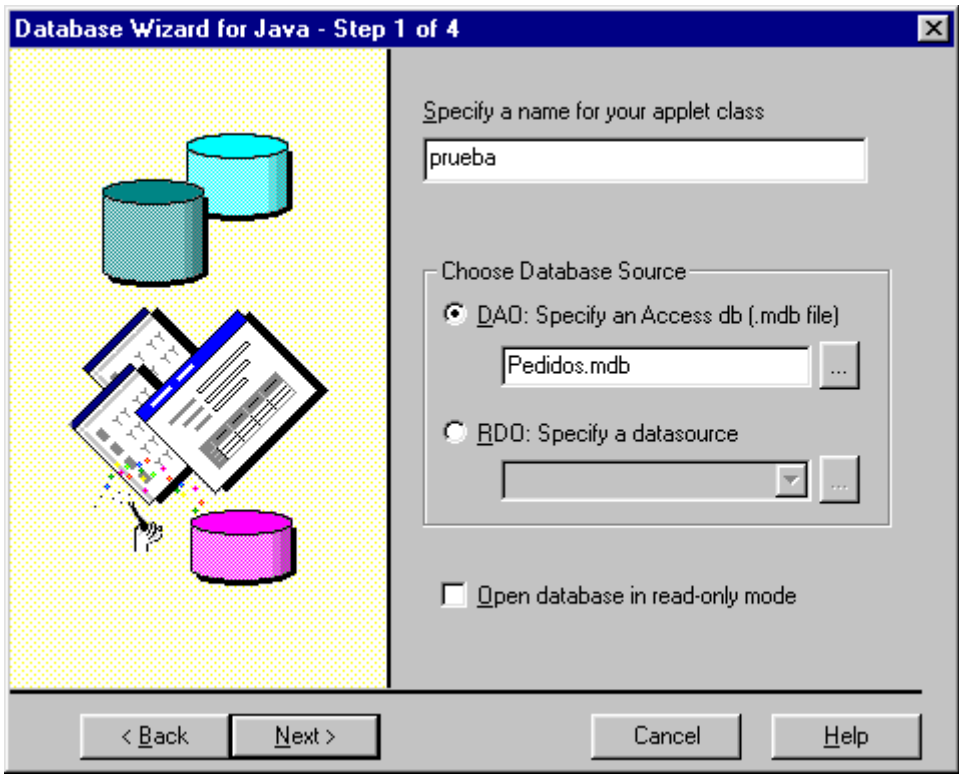
Una vez que ya estamos trabajando en un proyecto, podemos editar y crear sus componentes.



En el área izquierda podemos ver los distintos componentes del proyecto y en el área derecha el detalle de dichos componentes.

WIZARDS

Otra herramienta de gran utilidad del Visual J++ son los Wizards, estos nos permiten crear proyectos en base a unas preguntas que nos hace el asistente.



DEPURACION

Depurar un programa con el JDK puede ser realmente tedioso, difícil o imposible. En Visual J++ cada proyecto puede ejecutarse en modo normal o en modo de depuración. En este último caso se podrán colocar breakpoints, puntos de parada en el código, que nos permiten parar la ejecución y examinar los contenidos de las variables locales, los parámetros o los objetos que tengamos disponibles en ese momento. A partir de ahí, reanudamos la ejecución o la proseguimos “paso a paso”, es decir, instrucción por instrucción. Hasta tiene la posibilidad de ver el código de la máquina virtual Java, los propios bytecodes.

INCONVENIENTES

Por todo lo antes dicho, Visual J++ parece ser el entorno ideal de desarrollo pero presenta grandes inconvenientes al tratar de desarrollar una interfaz gráfica con botones gráficos o barra de herramientas.

7.5.3 BORLAND JBUILDER 2.0

La aplicación fue desarrollada en Java utilizando Borland JBuilder 2 Professional. No se consigue gratuitamente. Para el desarrollo se utilizó una versión de prueba.

Borland cubrió las falencias del JDK y Visual J++ con su producto JBuilder.

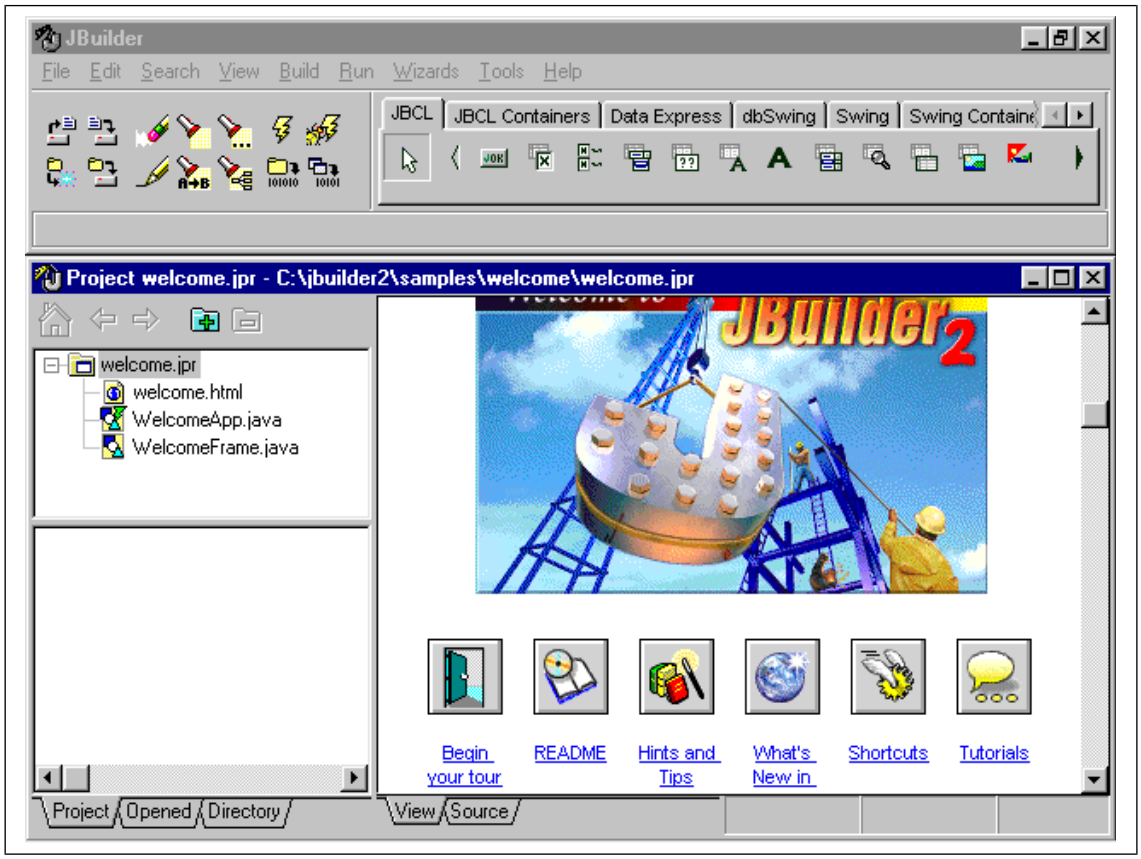
Para poder ejecutarse necesita como mínimo una máquina Pentium con Windows 95 o NT con 48 MB de memoria y monitor SVGA.

JBuilder 2 es un producto que contiene un conjunto de herramientas de desarrollo visuales para crear aplicaciones, applets y Java Beans. Comprende un ambiente de programación completo, compilador, bibliotecas de clases, depurador, editor, etc. Está basado en el JDK 1.1.6. Como una gran ventaja Borland permite cambiar la versión del JDK con la que se desea trabajar permitiendo seleccionar la versión JDK 1.1.x. o anteriores, pudiendo tener varias versiones disponibles a la vez. La gran desventaja es que no es compatible con JDK 1.2.

COMPONENTES

Una de las características más fuertes es su IDE (Integrated Development Environment) gráfico.

Al ingresar al JBuilder veremos la siguiente pantalla



Esta pantalla está compuesta por la ventana principal (en la parte superior) y la ventana llamada AppBrowser.

La ventana principal contiene una barra de menú, una barra de herramientas, la paleta de componentes y una barra de estado. La paleta de componentes muestra todos los componentes que se pueden utilizar para diseñar la interfaz del usuario.

La **paleta de componentes** permite construir fácilmente aplicaciones y applets Java, usando sólo herramientas gráficas. Todos los componentes son JavaBeans, es decir, componentes de software basados en Java reusables. JBuilder viene con más de 50 JabaBeans, entre ellos componentes visuales como botones y listas, como así también componentes designados para acceder y manipular información de bases de datos. JBuilder permite incluso agregar a la paleta JavaBeans ya sea de desarrollo propio o no.

Entre los componentes de la paleta cabe destacar los Swing. Estos han sido creados en conjunción de Sun con Netscape y proporcionan una serie de componentes muy bien descritos y especificados de forma que su presentación visual es independiente de la plataforma en que se ejecute el applet o la aplicación que utilice estas clases. Swing simplemente extiende el AWT añadiendo un conjunto de componentes, JComponents, y sus clases de soporte. Hay un conjunto de componentes de Swing que son análogos a los de AWT, y otros nuevos como los árboles. Swing no está incorporado en Visual J+ +.

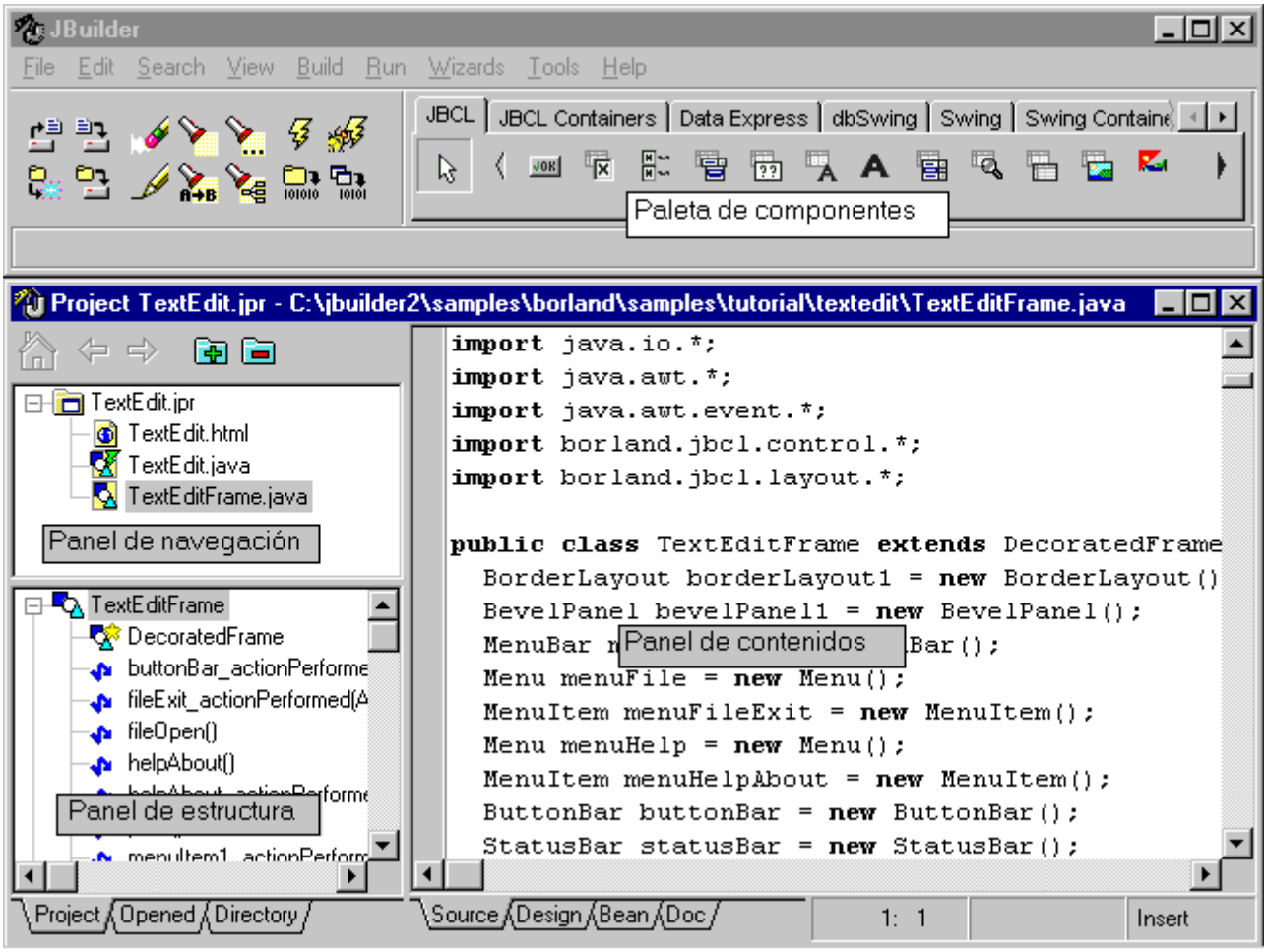
La ventana **AppBrowser** es usada para realizar todas las funciones de desarrollo habituales tales como explorar, editar, diseñar y depurar todo en una única ventana.

La ventana AppBrowser crea una interfaz para manejar todo el trabajo usando ventanas tabuladas. Permite dos formas para crear una pantalla, una es escribiendo a mano el código fuente y la otra más simple utilizando las herramientas visuales que provee, pudiendo cambiar entre una y otra para cualquier clase Java y hacer cambios en cualquiera de los dos ambientes. La sincronización del código de JBuilder asegura que cualquier cambio en un ambiente es instantáneamente reflejado en el otro. Esta es una característica que en otros programas como Visual Café no se reflejan.

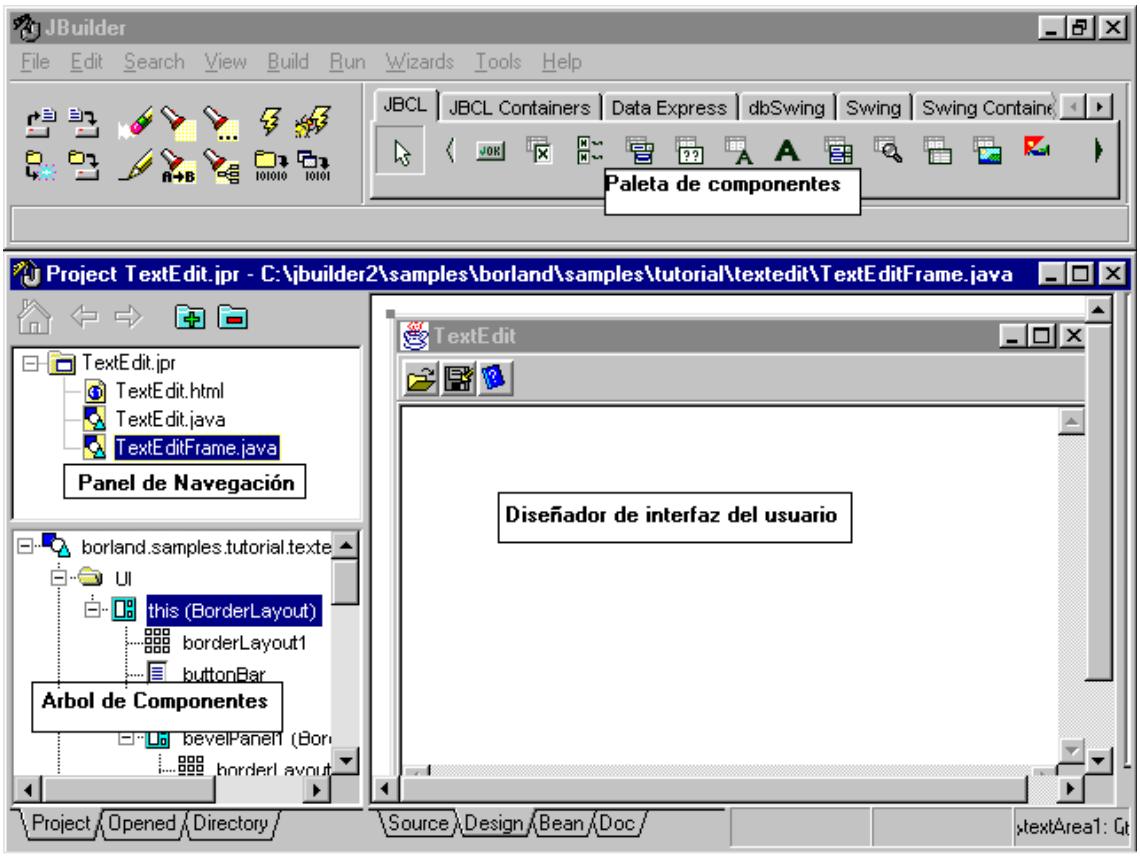
La ventana AppBrowser está compuesta por tres paneles:

- ✓ Panel de Navegación, se encuentra en la parte izquierda superior. Muestra la lista de los archivos con los que se está trabajando
- ✓ Panel de estructura en la parte izquierda inferior, muestra la información estructural del archivo seleccionado en el panel de navegación.
- ✓ Panel de contenidos en la parte derecha muestra el contenido detallado del archivo seleccionado en el panel de navegación.

La siguiente es la AppBrowser trabajando con el código fuente



La siguiente es la AppBrowser trabajando con la representación visual del código fuente



PROYECTOS

JBuilder maneja el concepto de proyecto. Un proyecto es el conjunto de programas y recursos que arman el sistema en el que estamos trabajando. Un proyecto Java podrá componerse de los distintos archivos de programa, archivos de gráficos y de sonido, y páginas HTML para invocar a la applet que estamos armando.

WIZARDS

Los wizards permiten ahorrar tiempo escribiendo el código por uno.

Provee dos tipos de wizards:

- Wizards de archivos: para crear nuevos archivos entre ellos JavaBean, proyectos, aplicaciones, applets, frames y diálogos.
- Wizards de utilidades entre ellos para sobreescribir métodos y para distribuir en archivos jar y zip.

DEPURACION

JBuilder permite depurar aplicaciones y applets en una forma muy sencilla. Un proyecto puede ejecutarse en modo normal o en modo de depuración. En modo depuración se pueden establecer breakpoints y ejecutar paso a paso cada instrucción. Se puede observar el valor de datos del programa, incluyendo clases, instancias y variables locales, parámetros de los métodos. Se pueden examinar estos valores y hasta incluso modificar. Modificar los valores de los datos del programa durante la depuración provee una forma de testear posibles errores durante la ejecución. Si se encuentra que una modificación repara el error, se puede salir de la depuración, reparar el código del programa y recompilarlo para repararlo en forma permanente.

INCONVENIENTES

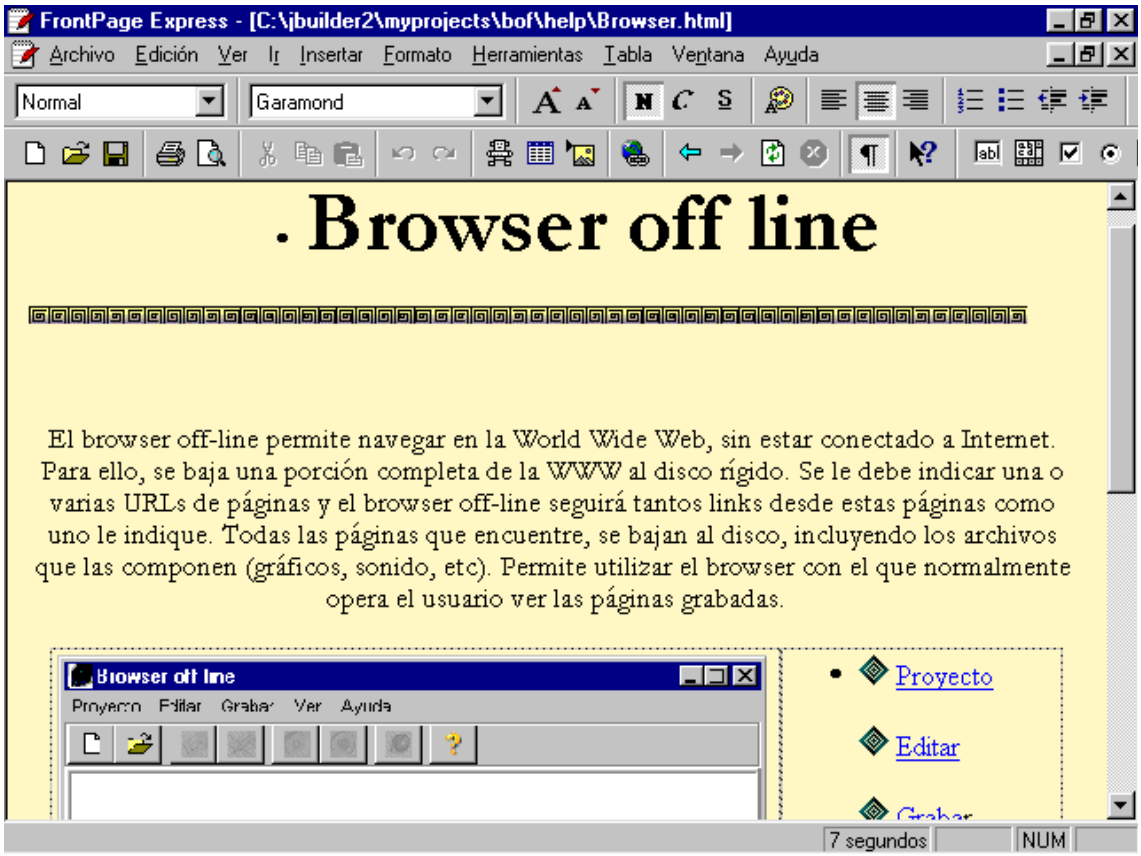
Existen métodos de Clases que no se comportan como deberían. Ej. el método setFilenameFilter (“*.java”) de la clase Filer debería filtrar los tipos de archivos a mostrar pero no lo hace.

No permite cambiar a la versión del JDK 1.2. Esta versión corrige varios errores que se presentan en las versiones anteriores.

7.5.4 HERRAMIENTAS ADICIONALES UTILIZADAS: FRONTPAGE EXPRESS

Para el desarrollo de la ayuda del sistema se crearon páginas html. Para ello se utilizó el FrontPage Express.

FrontPage Express es un editor de páginas Web que ofrece toda la eficacia de HTML (Lenguaje de marcado de hipertexto). Se puede usar FrontPage Express para crear y dar formato a páginas Web en HTML trabajando en una vista WYSIWYG ("lo que se ve es lo que se obtiene"), de modo que se podrá ver la apariencia real del formato y el diseño.



Las características de fácil uso de FrontPage Express permiten crear eficaces y sofisticadas páginas Web. Por ejemplo, se puede hacer lo siguiente:

- ✓ Aplicar etiquetas HTML eligiéndolas en la lista de la barra de herramientas, sin tener que escribirlas manualmente.
- ✓ Seleccionar texto o párrafos y aplicarles formato y alineación al hacer clic en un botón de la barra de herramientas.
- ✓ Insertar imágenes que se hayan creado o imágenes prediseñadas.
- ✓ Abrir directamente páginas Web existentes en Web o a partir de un archivo del equipo o de una red.
- ✓ Guardar el trabajo directamente en el Web (mediante el Asistente para la publicación en Web) o en un archivo.
- ✓ Agregar marquesinas (texto con desplazamiento).
- ✓ Agregar colores de fondo.
- ✓ Utilizar los componentes WebBot para agregar una amplia funcionalidad sin necesidad de realizar una programación compleja. Un WebBot es un "objeto dinámico" de una página Web que se ejecuta cuando el autor guarda la página en el

servidor de Web o, en algunos casos, cuando el usuario ve la página. FrontPage Express incluye WebBots que agregan a las páginas Web características de Marca de hora, Inclusión y Búsqueda.

- ✓ Utilizar controles ActiveX para ampliar la capacidad de las páginas Web.

FrontPage Express ofrece muchas de las características del Editor de Microsoft FrontPage en un paquete de menor tamaño. Es fácil cambiar a FrontPage sin necesidad de aprender un nuevo programa o método para crear páginas Web. FrontPage ofrece las características adicionales del Editor de FrontPage y el Explorador de FrontPage.

8. BREVE DESCRIPCION DE HTML

Básicamente, el programa de browsing off line baja al disco rígido una URL dada, incluyendo los archivos y links que la componen. Para ello, lo primero que hace es determinar el tipo de archivo de la URL, si es o no HTML. Si no es un archivo HTML copiará de a bytes todo el contenido. Si es una archivo HTML, debe realizar una análisis semántico de las líneas que componen el archivo, en busca de referencias a otros archivos. Por lo tanto, es necesario tener unos conocimientos básicos del HTML 4.0.

HTML es el lenguaje para publicar información en la World Wide Web. El HTML (Hypertext Markup Language) está formado por un conjunto de componentes que definen un documento. Es un lenguaje que define características físicas y estructurales de un documento (tipo de letra, ubicación de un párrafo o frase, el lugar que será activo para la conexión con otro documento, etc.). Esto lo logra por medio de órdenes o tareas que van colocadas al principio y al final del texto a marcar, dentro de los signos "mayor que" y "menor que".

Un componente de HTML puede incluir un nombre, algunos atributos y algún texto o hipertexto, y aparecerá en un documento de HTML como

<nombre_etiqueta> texto </ nombre_etiqueta >

< nombre_etiqueta nombre_atributo=argumento> texto </ nombre_etiqueta > o

< nombre_etiqueta >

Por ejemplo:

<title> Mi documento </title>

Un documento html está formado de un simple componente:

<html> ... </html>

que a su vez está compuesto de elementos cabecera y cuerpo

<head> . . . </head>

y

<body> . . . </body>

Entre los atributos del cuerpo se encuentra background=”URL”, que especifica un recurso imagen que se utilizará como fondo. (Este componente debe identificarse en el análisis de archivos referenciados por el documento HTML que realiza el browser off line)

ELEMENTOS DEL COMPONENTE HEAD

- ✓ <title>. . . </title>

  Especifica un título del documento. El título aparecerá en una barra de la ventana que identifica el contenido de la ventana.

- ✓ <base href = "URL">

  Especifica el nombre del archivo a partir del cual se buscarán las URLs relativas. Si no se especifica esta URL, las URLs relativas se resolverán a partir de la URL del documento actual. (Este componente debe identificarse en el análisis de archivos referenciados por el documento HTML que realiza el browser off line)

- ✓ <link rev="RELATIONSHIP" rel="RELATIONSHIP" href="URL">

  Permite definir relaciones entre el documento que contiene la etiqueta link y el documento especificado en “URL”. Los atributos rel y rev especifican relaciones entre el archivo HTML y la URL, hacia delante y atrás. (Este componente debe identificarse en el análisis de archivos referenciados por el documento HTML que realiza el browser off line)

ELEMENTOS DEL COMPONENTE BODY

- ✓ Elementos de texto

 ...

Identifica un párrafo que contiene texto que será formateado antes de mostrarse en pantalla.

<pre> ... </pre>

Identifica texto que ha sido preformateado por otro sistema y debe ser mostrado tal cual en pantalla. Puede incluir algunos elementos html dentro de él pero no todos funcionarán (Ej. link)

<xmp> ... </xmp>

Es similar a <pre> excepto en que los elementos html incluidos serán ignorados. Esta etiqueta es obsoleta, se utiliza <pre>.

<plaintext>

Es similar a <pre> excepto en que los elementos html incluidos serán ignorados y como no tiene etiqueta final, el resto del documento será tratado como un texto plano. Esta etiqueta es obsoleta, se utiliza <pre>.

✓ Links y Anclas

Un link es una conexión desde un recurso de la Web a otro. Un link tiene dos extremos, llamados anclas, y una dirección. El link empieza en el ancla origen y apunta al ancla destino, el cual puede ser cualquier recurso (imagen, video, sonido, programa, documento HTML, un elemento dentro del documento HTML, etc.)

 ...

Define una ubicación destino en el documento

 ...

Vincula a una ubicación en el mismo documento.

 ...

Vincula con otro documento o recurso de la Web. (Este componente debe identificarse en el análisis de archivos referenciados por el documento HTML que realiza el browser off line)

 ...

Vincula a un ancla en otro documento. (Este componente debe identificarse en el análisis de archivos referenciados por el documento HTML que realiza el browser off line)

Un ancla debe incluir un atributo name o href, y puede incluir ambos.

✓ Encabezados

<h1> ... </h1> Encabezado más importante

<h2> ... </h2>

<h3> ... </h3>

<h4> ... </h4>

<h5> ... </h5> Encabezado menos importante

ESTILOS LOGICO

Permite dar una característica a determinadas palabras del texto, es el estilo orientado al contenido del texto.

 . . .

Para dar énfasis a los caracteres que contiene. Generalmente se ve con letra itálica

<samp> . . . </samp>

Marca uno o varios caracteres como ejemplo.

<kbd> . . . </kbd>

Muestra una tecla del teclado

<var> . . . </var>

Define una variable

<dfn> . . . </dfn>

Muestra una definición

<cite> . . . </cite>

Es para pequeñas citas, como la cita de un libro por ejemplo. El texto que contiene es visto generalmente en itálica

ESTILO FISICO

Señala caracteres según un objetivo de visualización.

 . . .

Para que uno o varios caracteres sean vistos en negrita

 . . .

Para que uno o varios caracteres sean vistos en letra itálica

<u> . . . </u>

Para que uno o varios caracteres sean visualizados subrayados

GLOSARIO O LISTA DE DEFINICIONES

El glosario o lista de definiciones es un conjunto de ítems, cada uno de los cuales tiene un párrafo descriptivo propio.

La orden consta de tres partes: <DL> (siglas de Definition List) es la principal e incluye toda la lista; <DT> (siglas de Definition Term) que contiene el término a definir; y <DD> (siglas de Definition) que contiene la definición o explicación del término.

<dl>

<dt> Primer término a definir

<dd> Definición del primer término

<dt> Próximo término a definir

<dd> Definición del próximo término

...

</dl>

LISTA ORDENADA

La lista ordenada, <OL>, es una de las dos más usadas (junto a la lista no ordenada). Ordena los ítems según números o letras. Cada elemento va precedido de LI, que es el único elemento que puede contener una lista ordenada.

Las listas pueden ser anidadas, esto es, pueden acumularse.

 Primer ítem en la lista

 próximo ítem en la lista

...

LISTA NO ORDENADA

La lista no ordenada, <UL>, es, junto con la anterior, la lista más usada. Cada renglón es precedido por una LI, que es el único elemento que puede ir dentro de una UL. Organiza verticalmente los ítems, cada uno de los cuales se visualiza con un símbolo previo, como un asterisco o un círculo, etc.

 Primer ítem en la lista

 próximo ítem en la lista

...

MENU

<MENU> es un tipo de lista para un pequeño grupo de ítems con una descripción corta. Es como una lista no ordenada compacta. No es muy usada y es posible que desaparezca en el futuro.

Cada línea va con un elemento LI.

<menu>

 Primer ítem del menú

 próximo ítem del menú

...

</menu>

DIRECTORIO

El Directorio, <DIR>, define una lista de ítems cortos (no más de 20 caracteres). Cada ítem va precedido de LI, que es el único elemento que puede contener.

<dir>

 Primer ítem del directorio

 próximo ítem del directorio

...

</dir>

IMAGENES

Permite insertar una imagen en el documento

La forma básica del comando es <img src=”URL” alt=“Texto alternativo”>

Atributos:

src: es obligatorio. Señala el nombre del archivo de imagen a insertar. (Este componente debe identificarse en el análisis de archivos referenciados por el documento HTML que realiza el browser off line)

alt: opcional. Para aquellos usuarios que tengan un visor que no ve imágenes, señala un texto alternativo (lo más relacionado posible con la imagen) de hasta 1024 caracteres

align: ubica el texto que rodea a la imagen. Tiene los valores top, middle, bottom

ismap: si la etiqueta image está dentro de un ancla, la imagen se convertirá en una imagen clickeable

FORMULARIOS

Los formularios incorporan la posibilidad de ingresos de datos del usuario o lector, haciendo interactivo el documento.

Cuando el usuario activa el botón submit (enviar), los textos ingresados en los campos creados dentro del formulario son remitidos al servidor, para regresar a su vez una respuesta (agradecimientos, avisos de campos sin completar, etc.). El procesamiento de los datos en el servidor está a cargo de otros programas, fuera del lenguaje HTML.

Las siguientes etiquetas implementan la interfaz del formulario

<form> . . . </form>

Permite incluir uno o varios formularios dentro de cualquier sección del cuerpo del documento. Dentro de un formulario pueden incorporarse imágenes. No puede anidarse.

Atributos:

action=”URL”

Define la ubicación del programa que procesará el formulario

method=post/get

Indica el método para intercambio de datos entre el cliente y el programa que procesa el formulario: get (por defecto) o post

<input>

Define un campo de entrada para ingresos del usuario. Un campo es una caja vacía para incorporar texto. Solamente puede usarse dentro de FORM, y no hay límite para la cantidad de cajas (de INPUTs). Cada campo de entrada asigna un valor a una variable la cual está especificada con un nombre (name) y un tipo de datos específicos (type)

Atributos:

type=”tipo_variable”

Especifica el tipo de dato para la variable (text, password, radio, submit, reset, hidden, image)

name=”texto”

Texto es el nombre que identifica la variable de entrada

value=”texto”

Texto es el valor por defecto de la variable de entrada.

checked

Para el tipo "checkbox" o tipo="radio", si está presente este argumento, el campo de entrada está chequeado por defecto

size="ancho_de_muestra"

Ancho_de_muestra es un valor entero que representa el número de caracteres que se mostrarán para los tipos "text" o "password"

maxlength="longitud"

Longitud es el máximo de caracteres permitidos dentro de los tipos "text" o "password".

<select> . . . </select>

Define y muestra un conjunto de ítems de lista opcionales donde el usuario puede seleccionar uno o más ítems. Esos ítems son ingresados por OPTION.

Atributos:

name="texto"

Texto especifica el nombre-variable del elemento de SELECT.

size="longitud"

Longitud determina la cantidad de líneas que conforman el campo de opciones visualizables. Por defecto es 1: solamente se verá una línea mientras que las restantes estarán ocultas como en un menú. Si está presente el atributo MULTIPLE, determina cuántas líneas son vistas

multiple

Permite al usuario elegir más de un valor al mismo tiempo. Si este atributo no está presente, el usuario solamente podrá seleccionar un valor.

<option>

Define las diversas opciones a seleccionar dentro de un campo de un formulario. Esta tarea se ingresa dentro de SELECT, y sólo puede contener texto.

Si el atributo selected está presente entonces el valor de la opción es seleccionada por defecto.

```
<select multiple>
<option> Opción 1
<option selected> Opción 2
<option> Opción 3
</select>
```

<textarea> . . . </textarea>

Permite al usuario ingresar un texto en bloque dentro de un formulario. Es un campo más extenso que los INPUTs.

Atributos:

name=”texto”

Texto es el nombre que identifica la variable de entrada

rows=”cantidad_líneas” y cols=”cantidad_columnas”

Ambos atributos toman un valor entero que representa la cantidad de líneas y columnas del área de texto que se mostrará

9. ARQUITECTURA DEL SOFT

9.1 Clases

El sistema está compuesto por las siguientes clases

- ✓ BrowserOffLine: Es la clase principal que se encarga de poner en funcionamiento el sistema. Crea una instancia de la clase UIBrowser

- ✓ UIBrowser: Crea el frame principal que será la interfaz con la cual trabajará el usuario. Contiene los atributos y métodos generales del browser off line.

- ✓ Proyecto: Contiene los atributos y métodos para manejar un proyecto (conjunto de URLs a bajar con sus seteos correspondientes).

- ✓ BrowserControl: Esta clase permite mostrar una URL en el browser por defecto del sistema.

- ✓ CargarThread: Esta clase permite crear un nuevo hilo de ejecución que se utiliza para detectar que el usuario desea detener el hilo de grabación.

- ✓ TimerTest: Esta clase permite crear un timer que se utiliza para detectar el momento para empezar a grabar cuando el usuario ha configurado el sistema para que empiece a grabar a determinada hora.

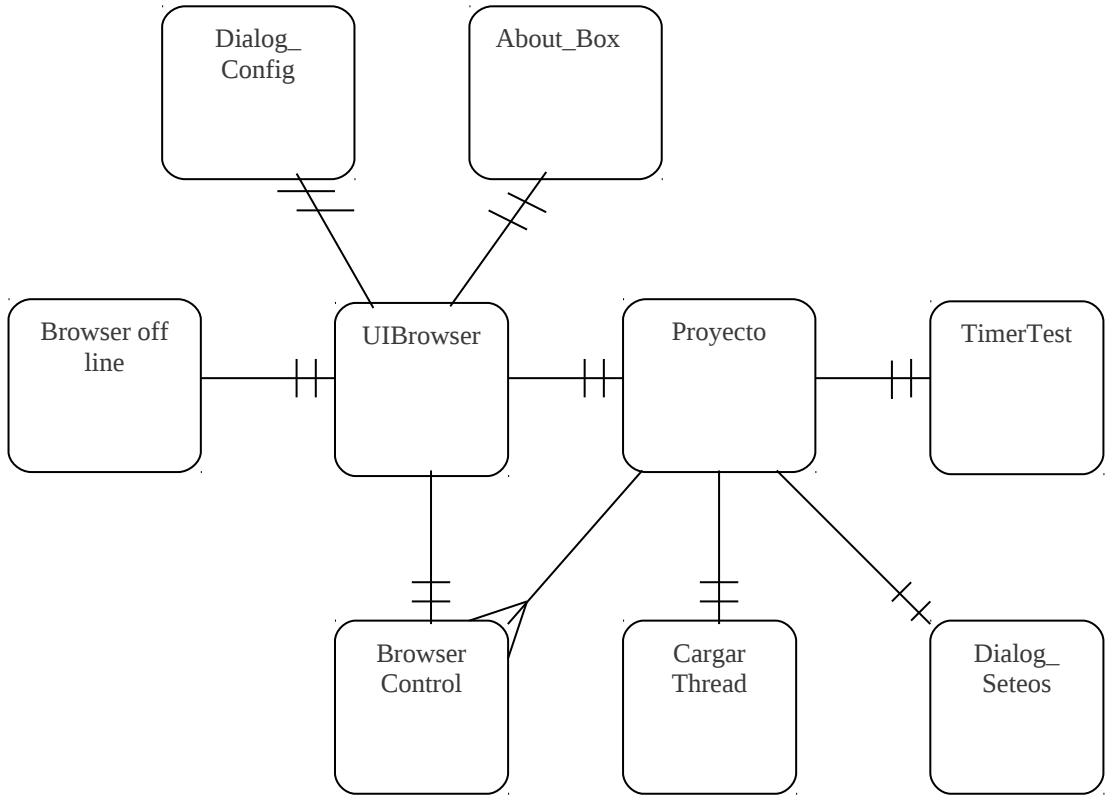
- ✓ Dialog_Config: Crea una ventana para realizar las configuraciones generales del browser off line. Contiene métodos para setear y obtener los seteos realizados por el usuario.

- ✓ Dialog_Seteos: Crea una ventana para realizar los seteos específicos de un proyecto. Contiene métodos para setear y obtener los seteos realizados por el usuario.

- ✓ About_Box: Crea una ventana con información acerca de la versión del browser off line

9.2 Conexiones entre los objetos

Los objetos no existen en una burbuja, razón por la cual se deben definir las relaciones para cada objeto del modelo. Una conexión de instancia es una notación de modelización que define una relación específica entre las instancias de un objeto.



Notación

Relación 1:1

Relación 1:muchos

9.3 Métodos y atributos de las clases

Se indican los principales atributos y métodos de las clases

Clase BrowserOffLine

Atributos -

Métodos BrowserOffLine

main

Clase UIBrowser

Atributos	**Métodos**
filer	jbInit
jtoolbar	cargarConfig
listControl	grabarConfig
loadThread	menuItemConfigurar_actionPerformed
proxy	bNew_actionPerformed
proyecto	menuItemNuevo_actionPerformed
puerto	bOpen_actionPerformed
timer	menuItemAbrir_actionPerformed
treeControl	bHelp_actionPerformed
treeRoot	menuItemIndice_actionPerformed
	bAdd_actionPerformed
	menuItemAgregar_actionPerformed
	bNoAdd_actionPerformed
	menuItemQuitar_actionPerformed
	menuItemSeteos_actionPerformed
	bPlay_actionPerformed
	menuItemEmpezar_actionPerformed
	bStop_actionPerformed
	menuItemDetener_actionPerformed
	bURL_actionPerformed
	menuItemURL_actionPerformed
	helpAbout_actionPerformed

Clase Proyecto

Atributos	**Métodos**
bajarNoMod	agregarURL
cantArchivos	bajarLinks
cantBytes	cargarBof
cantMaxArchivos	cargarURLs
cantMaxBytes	existeURL
ftp	fechaURL
hora	grabar
imágenes	grabarBof
maxArchivos	grabarURLs
maxBytes	quitarURL

mismoHost
minuto
niveles
URLs
fileNames
fileDates
dir
grabado
mostrarURL
nuevoProyecto
setear

Clase CargarThread

Atributos -

Métodos CargarThread
run

Clase TimerTest

Atributos hora
minuto

Métodos start
tick

Clase Dialog_Config

Atributos aceptar
cancelar
usarProxy
proxy
puerto

Métodos aceptar_actionPerformed
cancelar_ actionPerformed
Dialog_Config
jbInit
setear

Clase Dialog_Seteos

Atributos aceptar
bajarNoMod
cancelar

cantMaxArchivos
cantMaxBytes
ftp
hora
imagenes
maxArchivos
maxBytes
minuto
mismoHost
niveles
programar

Métodos aceptar_actiorPerformed
cancelar_actionPerformed
Dialog_Seteos
jbInit
setear

Clase About_Box

Atributos product
version
copyright

Métodos AboutBox
actionPerformed
jbInit

9.4 Diseño Detallado

9.4.1 Clase BrowserControl

La clase BrowserControl permite mostrar una URL en el browser por defecto del sistema. Permite controlar el Netscape o Internet Explorer desde Windows o Unix.
Desde una applet no hay inconveniente en visualizar una URL desde el navegador. El problema es visualizar una URL desde una aplicación, ya que no hay ningún método en todos los paquetes de Java que permita hacerlo.

Sin embargo utilizando el método exec() de la clase Runtime, es posible lanzar un proceso y hacer que ejecute un comando del Sistema Operativo; luego, es posible lanzar un navegador, lo cual reduce el problema a saber qué navegador se va a lanzar en cada una de las plataformas. La clase Runtime encapsula el proceso del intérprete Java que se ejecute. No se puede crear una instancia de Runtime; sin embargo, se puede obtener una referencia al objeto Runtime que se está ejecutando actualmente llamando al método Runtime.getRuntime().

El método exec() dispara un comando del sistema operativo subyacente. Devuelve un objeto Process, que se puede utilizar para controlar la interacción del programa Java con el nuevo proceso en ejecución. El problema a la hora de documentar exec() es que los programas que se ejecutan son muy dependientes del sistema. Se podría hacer exec ("netscape http://java.sun.com/") en Unix y exec(" rundll32 url.dll,FileProtocolHandler http://java.sun.com/") en Windows 95/NT para lanzar el navegador.

Lo primero que se hace previo a llamar el browser por defecto es determinar el sistema operativo. El método isWindowsPlatform de la clase BrowserControl determina si el sistema operativo es Windows utilizando el método de la clase System getProperty("os.name"). Según el sistema operativo que se obtenga ejecuta el comando para llamar al navegador.

En Unix, si el navegador es Netscape, el comando para abrirlo será distinto en función de que ya esté corriendo o no; en el primer caso el comando para arrancar el navegador es:

netscape -remote openURL(http://java.sun.com/)

y, en caso de que el navegador no esté en ejecución, el comando a utilizar para abrirlo es:

netscape http://java.sun.com/

Bajo entornos Windows, es más simple el comando que hay que utilizar, ya que no hay necesidad de saber si el navegador está lanzado o no, y además, siempre se invoca al navegador por defecto. Si este navegador por defecto es el Internet Explorer, la dirección URL se presentará sobre él. Para visualizar la página, el comando a usar es el siguiente:

rundll32 url.dll,FileProtocolHandler http://java.sun.com/

Básicamente, el comportamiento del método displayURL de la clase BrowserControl es el siguiente:

Si isWindowsPlatform()

  Runtime.getRuntime.exec(“rundll32 url.dll,FileProtocolHandler http://java.sun.com/”)

Sino

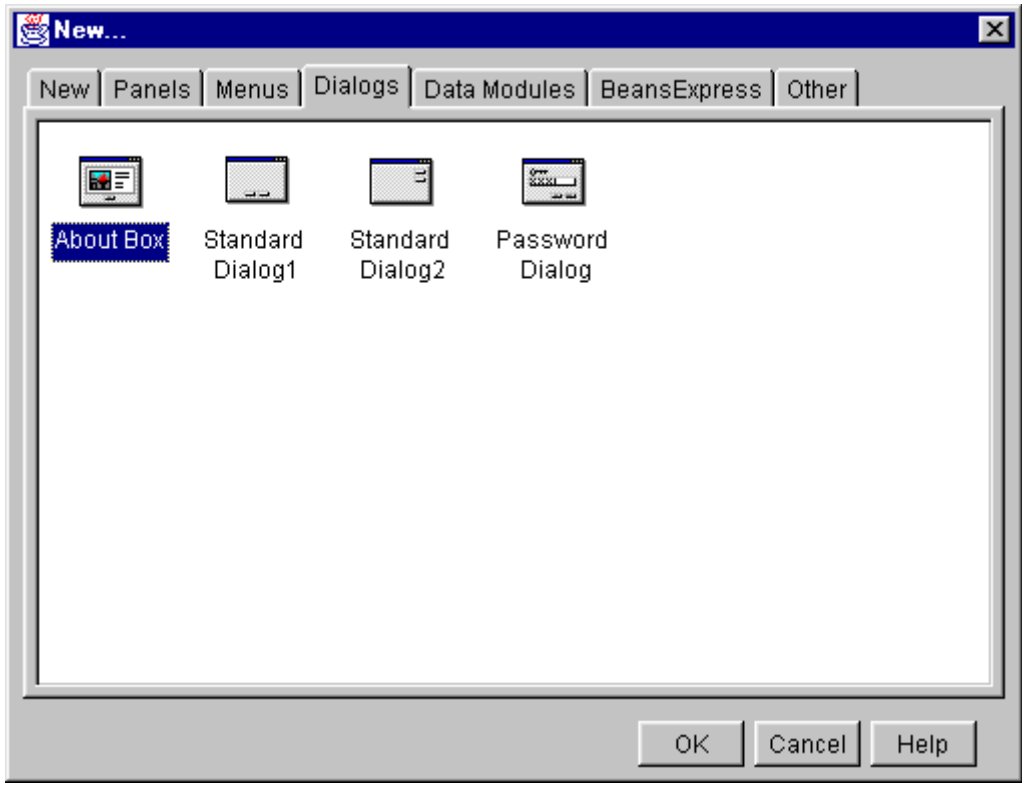
  Runtime.getRuntime.exec(“netscape http://java.sun.com/”)

FinSi

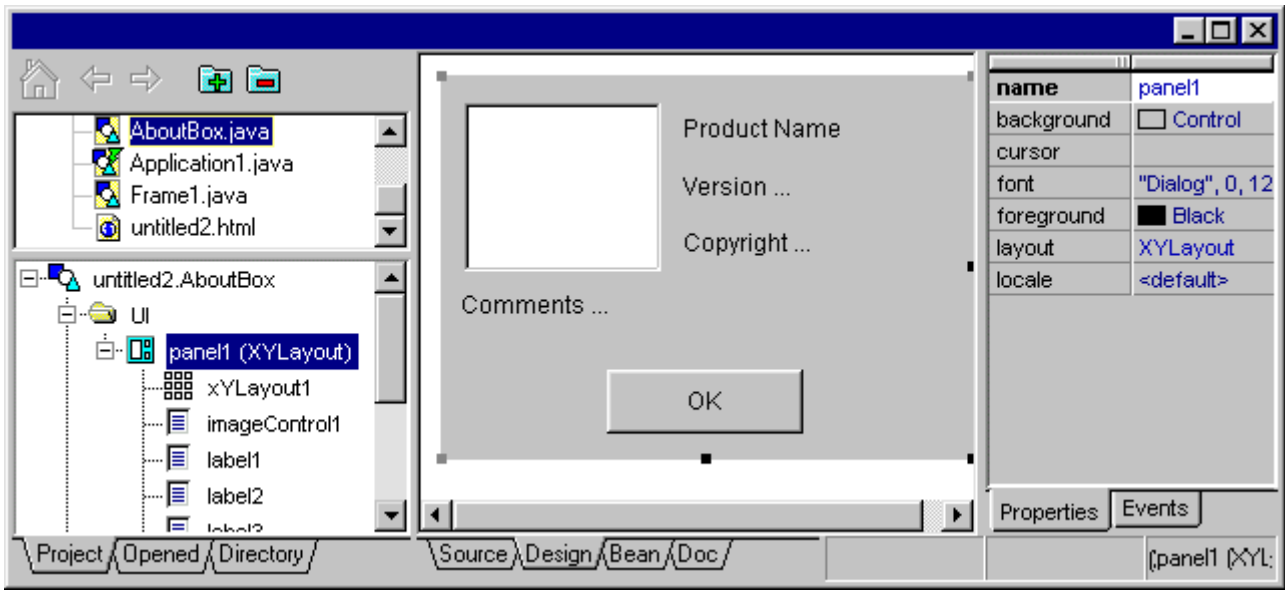
9.4.2 Clase AboutBox

Esta clase implementa la ventana con el mensaje de “Acerca de” el sistema. Para implementarla se utilizó el wizard del JBuilder para crear diálogos de este tipo.

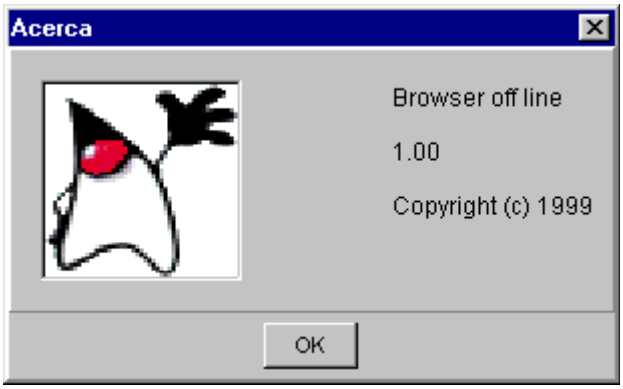
La siguiente es la pantalla para crear diálogos a través de wizards del JBuilder:



Automáticamente crea la clase. La siguiente es la ventana AppBrowser mostrando la representación visual del código fuente de la clase AboutBox:



Luego sólo hay que adecuar las etiquetas del nombre del sistema, la versión y la imagen que mostrará y obtenemos:



9.4.3 Clase TimerTest

Esta clase permite crear un timer que se utiliza para detectar el momento para empezar a grabar cuando el usuario ha configurado el sistema para que grabe automáticamente a determinada hora.

Para ello crea un objeto de la clase Timer. Esta clase provoca que ocurra una acción con una frecuencia dada. Cuando se crea un objeto Timer hay que indicarle el intervalo en milisegundos. Cada vez que pasa este tiempo, se llama automáticamente al método tick en el cual incorporamos la acción que queremos que realice. Esto se sigue ejecutando constantemente hasta que se ejecuta el método stop del timer. Se puede detener, continuar o resetear un timer en cualquier momento.

En la clase TimerTest creo un objeto Timer con una frecuencia de 30 segundos. En el método tick controlo si llegó la hora a la que se programó al sistema para que empiece a grabar automáticamente.

9.4.4 Clase CargarThread

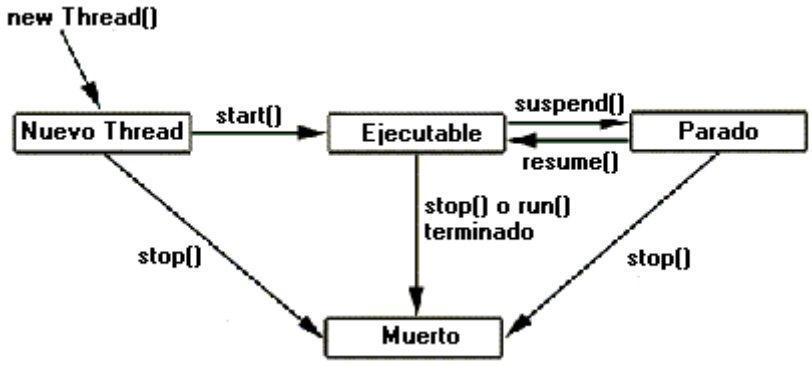
Esta clase permite crear un nuevo hilo de ejecución que se utiliza para detectar que el usuario desea detener el hilo de grabación. Esta clase está implementada como una subclase de Thread. Cada thread (hilo, flujo de control del programa) representa un proceso individual ejecutándose en un sistema. Típicamente, cada thread controla un único aspecto dentro de un programa. Todos los threads comparten los mismos recursos.

Un programa de flujo único o mono-hilvanado (single-thread) utiliza un único flujo de control (thread) para controlar su ejecución. Muchos programas no necesitan la potencia o utilidad de múltiples flujos de control.

La utilización de threads en Java, permite una enorme flexibilidad a los programadores a la hora de plantearse el desarrollo de aplicaciones. La simplicidad para crear, configurar, ejecutar y detener threads, permite que se puedan implementar muy poderosas y portables aplicaciones/applets que no se pueden con otros lenguajes. En un lenguaje orientado a Internet como es Java, esta herramienta es muy útil.

Una forma de conseguir threads en Java es extender la clase Thread. De esta manera se heredan los métodos y variables de la clase padre. El método run() es donde se realizará todo el trabajo de la clase. Por lo tanto, para implementar un thread habrá que sobrecargar el método Thread.run()

Durante el ciclo de vida de un thread, éste se puede encontrar en diferentes estados. La figura siguiente muestra estos estados y los métodos que provocan el paso de un estado a otro.



La ventaja de la utilización de múltiples threads en una aplicación, o aplicación multithreaded, es que pueden comunicarse entre sí. Se pueden diseñar threads para utilizar objetos comunes, que cada thread puede manipular independientemente de los otros threads. En este caso en particular, el objeto en común que manejan los threads es el proyecto.

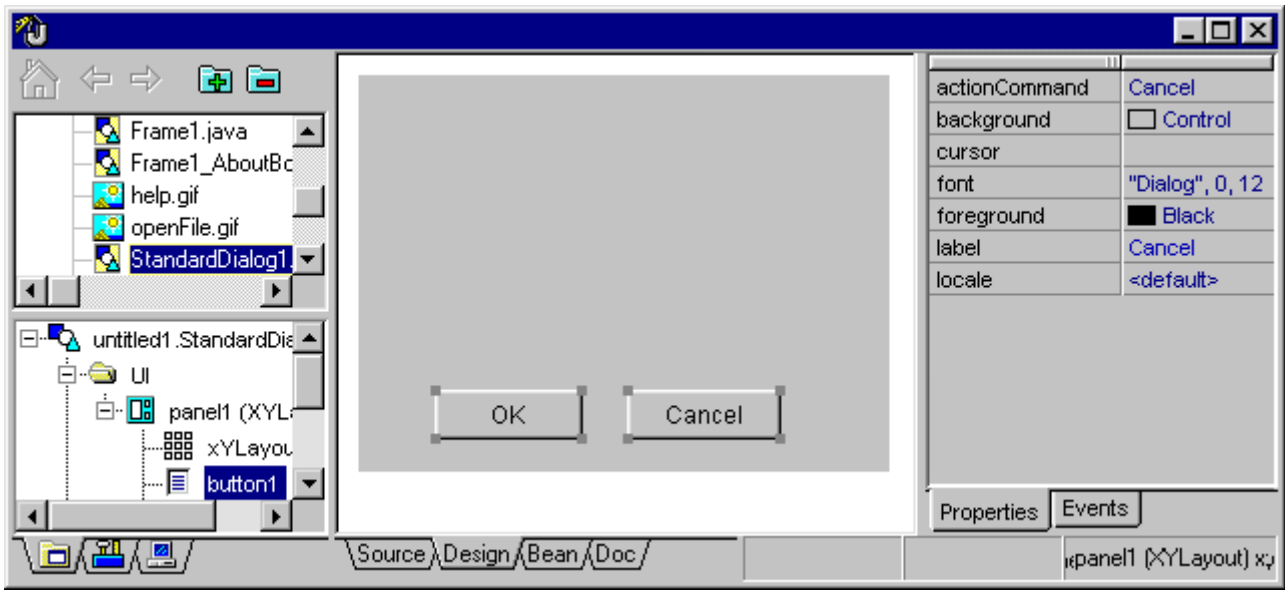
En la clase CargarThread, se sobrecarga el método run. Este lanza la grabación del proyecto. Como la grabación puede demandar varios segundos o minutos, hace falta que la grabación se realice en forma independiente para poder detectar en el hilo principal si el usuario desea detener la grabación. Cuando se quiere detener la grabación, se destruye al hilo que está grabando.

De esta forma, el sistema está grabando archivos en el disco que trae de la Web en un hilo de ejecución y soportando que el usuario realice una entrada por el teclado que requiera la detención de la grabación en otro hilo diferente.

9.4.5 Dialog_Seteos

Crea una ventana para realizar los seteos específicos de un proyecto. Contiene métodos para setear y obtener los seteos realizados por el usuario.

Para implementarla se utilizó el wizard del JBuilder para crear diálogos. Crea automáticamente un panel con dos botones OK y CANCEL.



Luego hay que agregar los campos de texto para que el usuario ingrese los seteos del proyecto. Para ello se utilizó la clase TextFieldControl de Borland. Una de las desventajas de las clases de Java que implementan campos de texto es que no proveen métodos para restringir la entrada del usuario a un determinado tipo de dato (entero, fecha, string, etc.). Para poder evitar que en los campos numéricos el usuario ingresara un valor incorrecto recurrí al manejo excepciones.

Una excepción es un evento que ocurre durante la ejecución de un programa y detiene el flujo normal de la secuencia de instrucciones de ese programa; en otras palabras, una excepción es una condición anormal que surge en una secuencia de código durante su ejecución.

Las excepciones en Java están destinadas, al igual que en el resto de los lenguajes que las soportan, para la detección y corrección de errores. Si hay un error, la aplicación no debería morirse. Se debería lanzar (throw) una excepción que a su vez se debería capturar (catch) y resolver la situación de error, o poder ser tratada finalmente (finally) por un gestor por defecto u omisión. Utilizadas en forma adecuada, las excepciones aumentan en gran medida la robustez de las aplicaciones.

La gestión de excepciones en Java proporciona un mecanismo excepcionalmente poderoso para controlar programas que tengan muchas características dinámicas durante su ejecución. Las excepciones son formas muy limpias de manejar errores y problemas inesperados en la lógica del programa, y no deberían considerarse como un mecanismo general de ramificaciones o un tipo de sentencias de salto.

Normalmente, un programa termina con un mensaje de error cuando se lanza una excepción. Sin embargo, Java tiene mecanismos para excepciones que permiten ver qué excepción se ha producido e intentar recuperarse de ella.

El primer paso en la construcción de un manejador de excepción es encerrar las declaraciones que tengan la posibilidad de generar un error de excepción durante la ejecución del programa en un bloque try

Una declaración try debe ser acompañada por al menos un bloque catch o un bloque finally (define un bloque de código que se quiere que sea ejecutado siempre, de acuerdo a si se capturó la excepción o no).

Se asocia el manejador con una declaración try poniendo uno o mas bloques catch directamente después que el bloque try :

```
try {
    // Declaraciones Java que podrían generar una excepción
} catch (AlgunObjectoThrow Nombrevariable) {
     // Declaraciones Java
} catch (AlgunObjectoThrow Nombrevariable) {
     // Declaraciones Java
}
```

El bloque try del código es donde se prevé que se genere una excepción. Es como si dijésemos "intentar estas sentencias y ver si se produce una excepción". El bloque try tiene que ir seguido, al menos, por una cláusula catch o una cláusula finally. Es el código que se ejecuta cuando se produce la excepción. El bloque catch es como si dijésemos "controlar cualquier excepción que coincida con mi argumento".

El tipo de argumento AlgunObjetoThrow declara el tipo de excepción que el manejador puede manejar y es el nombre por el cual el manejador puede referirse a la excepción que esta comprometida.

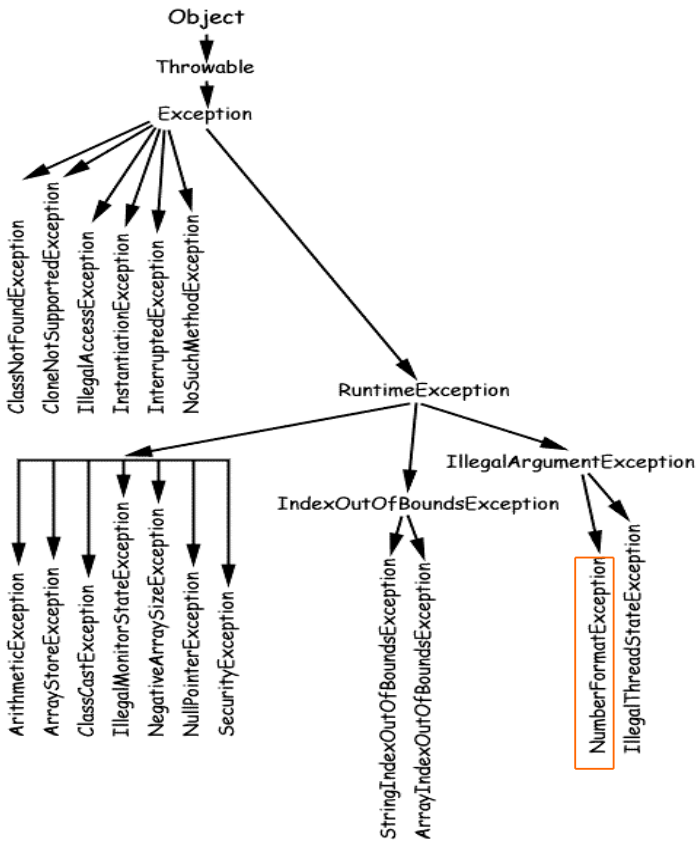
El bloque catch contiene una serie de declaraciones Java. Esas declaraciones son ejecutadas cuando el manejador de excepción es invocado.

Visto desde el lenguaje Java una excepción es cualquier objeto que es derivado directa o indirectamente de la clase Throwable (o cualquiera de sus subclases).

La clase Throwable tiene dos subclases: Error y Exception. Un Error indica que se ha producido un fallo no recuperable, del que no se puede recuperar la ejecución normal del programa, por lo tanto, en este caso no hay nada que hacer. Los errores, normalmente, hacen que el intérprete Java presente un mensaje en el dispositivo estándar de salida y concluya la ejecución del programa. El único caso en que esto no es así, es cuando se produce la muerte de un thread, en cuyo caso se genera el error ThreadDead, que lo que hace es concluir la ejecución de ese hilo, pero ni presenta mensajes en pantalla ni afecta a otros hilos que se estén ejecutando.

Una Exception indicará una condición anormal que puede ser subsanada para evitar la terminación de la ejecución del programa. Hay nueve subclases de la clase Exception ya predefinidas, y cada una de ellas, a su vez, tiene numerosas subclases.

Java proporciona muchas excepciones predefinidas. Las excepciones predefinidas y su jerarquía de clases es la que se muestra en la figura:



Para detectar si el valor ingresado por el usuario es numérico se utilizó la excepción NumberFormatException. Esta excepción es arrojada para indicar que la aplicación ha tratado de convertir un string a uno de los tipos numéricos, pero el string no tiene el formato apropiado. Cuando no es un número se mantiene el foco en el mismo campo hasta que ingrese un valor correcto.

El código utilizado para restringir la entrada del usuario a un valor numérico es el siguiente:

```
void text_cantMaxArchivos_focusLost(FocusEvent e) {
   double aux;
   try {
      aux=Double.valueOf(text_cantMaxArchivos.getText()).doubleValue();
   } catch (NumberFormatException nfe) {
   text_cantMaxArchivos.requestFocus();
   }
```

9.4.6 Clase Dialog_Config

Crea una ventana para realizar las configuraciones generales del browser off line. Contiene métodos para setear y obtener los seteos realizados por el usuario.

Para implementarla se utilizó el wizard del JBuilder para crear diálogos. La implementación es similar a la realizada en la clase Dialog_Seteos.

9.4.7 BrowserOffLine

Es la clase principal que se encarga de poner en funcionamiento el sistema. Crea una instancia de la clase UIBrowser.

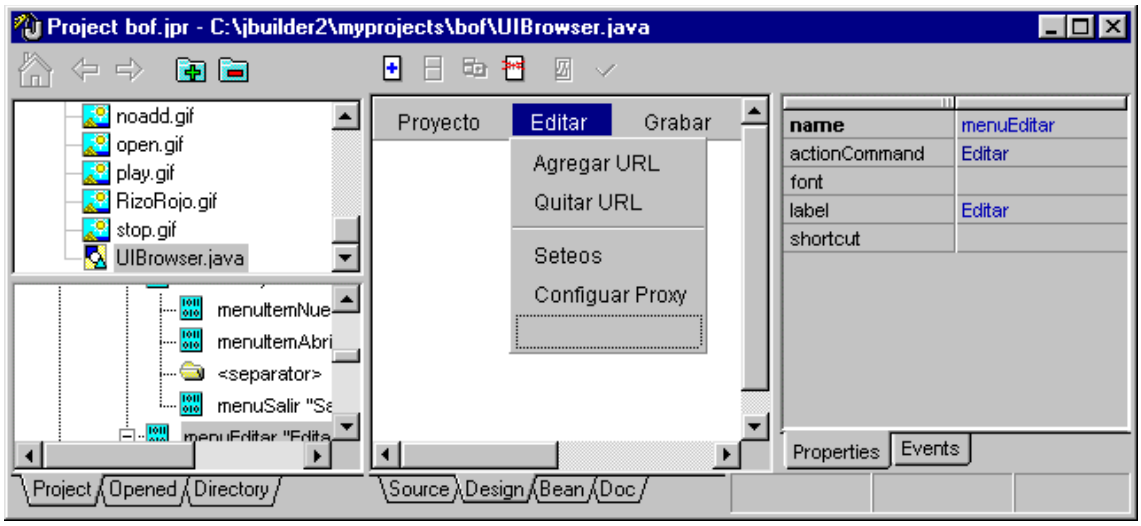
Para implementarla se utilizó el wizard del JBuilder para crear aplicaciones. Crea un proyecto JBuilder con dos archivos:

- ✓ Un archivo para la clase que arranca la aplicación que se llama BrowserOffLine.java. Esta clase contiene el método main() para la aplicación
- ✓ Un archivo para la clase de interface con el usuario llamada UIBrowser.java, que extiende a la clase DecoratedFrame. Esta clase es el principal contenedor.

9.4.8 Clase UIBrowser

Crea el frame principal que será la interfaz con la cual trabajará el usuario. Contiene los atributos y métodos generales del browser off line. El frame principal es una subclase de DecoratedFrame.

Para implementar el menú se utilizó el diseñador de menú de JBuilder. Este es una interface que permite diseñar visualmente el menú.



Detalle de los métodos:

⇒ jbInit: Configura el frame principal: ubicación, título de la ventana, etc. Crea una barra de menú con todas las opciones, una barra de herramientas, una barra de estado. Configura el frame principal. Ejecuta el método cargarConfig para inicializar las variables de instancia de la clase UIBrowser.

⇒ cargarConfig: inicializa las variables de instancia de la clase UIBrowser que se utilizan para las configuraciones generales del browser off line, a saber: si utiliza un proxy, el nombre del proxy y el puerto. Estos valores se encuentran almacenados en un archivo de texto llamado “config.txt” cuyo contenido es similar a:

```
usarProxy=1
proxy=proxy1
puerto=8080
```

Para acceder al archivo utilizamos la clase File. Esta se utiliza para acceder a objetos de archivo y de directorio. Emplea los convenios de nomenclatura del sistema operativo local. File incorpora constructores para crear archivos y directorios. Estos constructores admiten vías de acceso y nombres de archivo y directorio tanto absolutos como relativos. La entrada y salida de archivos se basa en el uso de flujos de datos llamados streams (los flujos también se utilizan para la entrada desde teclado, salida por pantalla, entrada y salida a buffers de memoria). Los flujos son secuencias de bytes que viajan desde un origen a un destino a través de una vía de comunicación. Cuando un programa escribe en un flujo, es el origen de éste. Cuando lee desde un flujo, es el destino de éste. La vía de comunicación depende del tipo de E/S que se lleve a cabo y puede consistir en transferencias de memoria a memoria, sistema de archivos, redes y otras formas de E/S.

El paquete java.io define las clases que componen la E/S. Ente las clases del paquete se encuentran los siguientes flujos: Reader, Writer, InputStream y OutputStream. A partir de estos flujos pueden establecerse subclases a fin de que proporcionen diversas funciones de E/S. La clase Reader es similar a la clase InputStream en el sentido de que es la raíz de una jerarquía de clases de entrada. Reader da soporte a la entrada de caracteres Unicode 16 bits, mientras que InputStream da soporte a la entrada de bytes de 8 bits. Las clases Writer y OutputStream son la analogía de salida de la clase Reader e InputStream respectivamente. Dan soporte a la salida de caracteres Unicode de 16 bits y 8 bits.

Para la lectura del archivo “config.txt” se utilizó la clase BufferedReader, subclase de Reader, que da soporte a la entrada de caracteres con búfer.

Además se utiliza para la identificación de las palabras del archivo “config.txt” la clase StreamTokenizer. Se utiliza para realizar un análisis sintáctico, convierte un flujo de caracteres de entrada en un flujo de tokens (símbolos) o unidades léxicas,

permitiendo leer de a un token por vez. StreamTokenizer permite reconocer identificadores, número, strings entre comillas y varios estilos de comentarios.

⇒ grabarConfig: graba en el archivo “config.txt” las variables de instancia de la clase UIBrowser. Para el manejo del archivo se utilizaron las clases File y BufferedWriter (da soporte a la salida de caracteres con búfer).

⇒ menuItemConfigurar_actionPerformed: este método crea una instancia de la clase Dialog_Config para que el usuario realice los seteos del proxy.

⇒ bNew_actionPerformed y menuItemNuevo_actionPerformed: crea un objeto Proyecto y ejecuta el método de proyecto nuevoProyecto, ya sea porque el usuario selecionó la opción del menú o hizo click en el botón de la barra de herramientas para crear un proyecto nuevo.

⇒ bOpen_actionPerformed y menuItemAbrir_actionPerformed: crea un objeto Proyecto y ejecuta el método de proyecto nuevoProyecto

⇒ bHelp_actionPerformed y menuItemIndice_actionPerformed: crea un objeto BrowserControl para mostrar la página de ayuda que se encuentra instalada en el disco local, usando el método displayURL.

⇒ bAdd_actionPerformed y menuItemAgregar_actionPerformed: llaman al método agregarURL del objeto Proyecto creado.

⇒ bNoAdd_actionPerformed y menuItemQuitar_actionPerformed: llaman al método quitarURL del objeto Proyecto creado.

⇒ menuItemSeteos_actionPerformed: llaman al método setear del objeto Proyecto creado.

⇒ bPlay_actionPerformed y menuItemEmpezar_actionPerformed: crea un objeto CargarThread.

⇒ bStop_actionPerformed y menuItemDetener_actionPerformed: se encarga de detener al thread que está grabando.

⇒ bURL_actionPerformed y menuItemURL_actionPerformed: llaman al método mostrarURL del objeto proyecto creado.

⇒ helpAbout_actionPerformed: crea un objeto de la clase AboutBox.

9.4.9 Clase Proyecto

Contiene los atributos y métodos para manejar un proyecto.

⇒ nuevoProyecto: este método se utiliza para crear un nuevo proyecto y abrir uno existente. Si existe un proyecto abierto, ejecuta los métodos grabarBof y grabarURLs.

Luego crea un objeto Filer para que el usuario ingrese el nombre del proyecto. El componente Filer muestra un cuadro de diálogo “Abrir” o “Guardar como” para abrir o guardar archivos. El cuadro de diálogo muestra la lista de directorios y archivos que el usuario puede usar para seleccionar un archivo a abrir o guardar.

Permite setear propiedades para el tipo de cuadro (Abrir o Guardar), el directorio y archivo a seleccionar. Otra propiedad muy útil que permite setear es “filenameFilter” (filtro de nombres de archivo), pero en la versión utilizada del JDK no realiza la función que debería: mostrar al usuario sólo determinados tipos de archivos. Esta es una falla del JDK.

Una vez que el usuario haya seleccionado el nombre del proyecto, si es nuevo, se creará un archivo con extensión “.bof”, un directorio con el mismo nombre del proyecto y dentro de este directorio un archivo llamado “urls.txt”. El primer archivo se utilizará para guardar la configuración propia del proyecto, el directorio es para almacenar todas las páginas grabadas y el archivo urls.txt para guardar información de las URLs bajadas entre ellos: URL, archivo local, fecha y hora de la última actualización de la URL.

Si el proyecto ya existe, ejecuta los métodos cargarBof y cargarURLs para cargar los seteos propios del proyecto y las identificaciones de las URLs que se usan en el proyecto.

⇒ grabarBof: graba en el archivo con extensión “.bof” los seteos particulares del proyecto. Para el manejo del archivo se utilizaron las clases File y BufferedWriter.

El contenido del archivo con extensión “.bof” es similar a:

```
niveles=2
imagenes=1
grabado=1
ftp=0
mismoHost=1
maxBytes=0
cantMaxBytes=0
maxArchivos=0
```

cantMaxArchivos=0
programar=1
hora=10
minuto=30
bajarNoMod=1

⇒ grabarURLs: graba en el archivo "urls.txt" información de las urls del proyecto. Para el manejo del archivo se utilizaron las clases File y BufferedWriter. El contenido de este archivo es similar al siguiente:

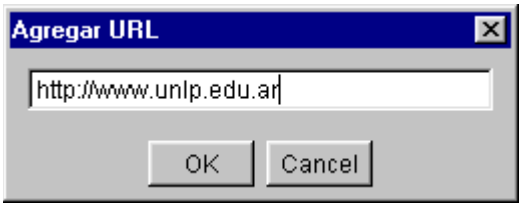
http://www.unlp.edu.ar/grado.htm grado.htm 09-sep-98 21:22:14
http://www.unlp.edu.ar/gerencia/

Estos datos representan la identificación de la URL y si se encuentran grabados, el nombre del archivo local, la fecha y hora del archivo en la WWW (estos últimos se utilizan para detectar si fue modificada la URL).

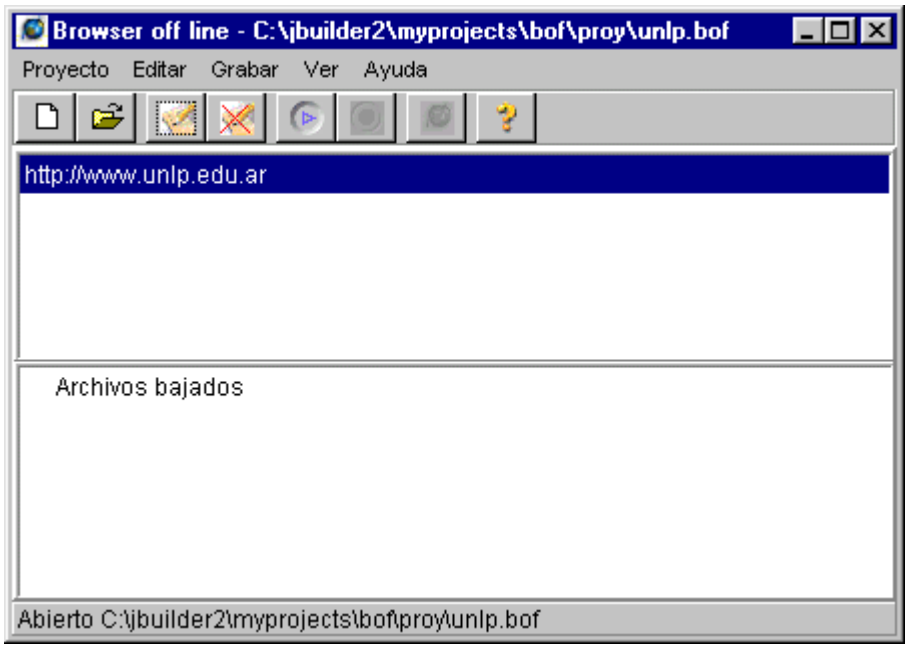
⇒ cargarBof: inicializa las variables de instancia de la clase Proyecto que se utilizan para las configuraciones particulares del proyecto abierto, a saber: cantidad de niveles, si se bajan imágenes, ftp, máxima cantidad de bytes, máxima cantidad de archivos a bajar, etc. Estas configuraciones se encuentran almacenadas en el archivo con extensión ".bof". Para implementar este método se utilizaron las clases: File, BufferedReader y StreamTokenizer.

⇒ cargarURLs: se encarga de guardar en un vector las urls con las que trabaja el proyecto. Estas urls se encuentran almacenadas en el archivo urls.txt. Para implementar este método se utilizaron las clases: File, BufferedReader y StreamTokenizer.

⇒ agregarURL: permite que el usuario ingrese las URLs a partir de las cuales quiere grabar. Para la interfaz se utilizó el componente StringInput. StringInput es un cuadro de diálogo que contiene una caja para la entrada de texto y botones OK y CANCEL.



Si el usuario confirma la entrada, esta URL se agrega al vector de URLs a bajar que se muestra en pantalla.



⇒ quitarURL: permite que el usuario elimine la URL seleccionada de la lista de URLs a bajar.

⇒ mostrarURL: crean un objeto BrowserControl para que muestre los archivos grabados con el browser por defecto del sistema.

⇒ setear: este método crea una instancia de la clase Dialog_Seteos para que el usuario realice los seteos particulares del proyecto.

⇒ grabar: realiza las siguientes funciones:

1. **Configuar el proxy:** Lo primero que hay que hacer antes de empezar a grabar, en el caso de que el usuario acceda a Internet a través de una red con un servidor proxy, es preparar el sistema para ello. Un proxy es un servicio que yace entre Internet y la red interna y maneja las conexiones entre los dos mundos. Los proxys ayudan a reducir las amenazas externas de hackers accediendo a la red interna mientras que permiten a los usuarios internos acceder a servicios de Internet. Mientras Java simplifica escribir aplicaciones clientes de Internet, estos clientes son inútiles si no pueden pasar a través del proxy. El secreto de combinar Java y proxys está en activar ciertas propiedades del sistema en el runtime de Java. Por lo tanto, antes de empezar a trabajar con el protocolo de Internet HTTP, hay que agregar las siguientes líneas a la aplicación:

```
System.getProperties().put( "proxySet", "true" );
System.getProperties().put( "proxyHost", "proxy" );
System.getProperties().put( "proxyPort", "80" );
```

La primera línea indica a Java que se usará un proxy para las conexiones, la segunda línea especifica la máquina en la que reside el proxy y la tercera línea indica el puerto en el que atiende el proxy.

Para acceder a URLs utilizando el protocolo FTP hay que agregar además las siguientes líneas:

```
System.getProperties().put( "ftpProxySet", "true" );
System.getProperties().put( "ftpProxyHost", "proxy" );
System.getProperties().put( "ftpProxyPort", "80" );
```

Afortunadamente es sencillo acceder a través de un proxy en Java, la desventaja es que no está documentado. Si creamos una aplicación con la siguiente línea:

```
System.getProperties().list(System.out);
```

Con esta línea podremos ver todas las propiedades del sistema, pero no se mostrarán las explicadas anteriormente: proxySet, proxyHost, proxyPort, ftpProxySet, ftpProxyHost y ftpProxyPort.

2. **Armar el vector completo de URLs:** para poder bajar la URL hace falta conocer la URL completa incluyendo el nombre del archivo. Para ello se prueba si existe la URL con el método existeURL y en el caso de que sea un directorio se prueba tratar de acceder al archivo index.html, index.htm y default.asp. También se detecta si está repetida la URL, eliminándola del vector.
3. **Detectar URLs modificadas:** para el caso que el usuario haya seteado que sólo se bajen las URLs que hayan sido modificadas y en el caso de que ya se hayan grabado previamente las URLs, se accede a través del método fechaURL a la fecha de última modificación de la URL para detectar si coincide con la fecha de la grabada previamente.
4. **Grabar URLs:** una vez preparado el vector con todas las URLs a grabar, se utiliza el método bajarLinks tantas veces como niveles a bajar haya seteado el usuario. Al método bajarLinks se le pasa como parámetro la lista de URLs a bajar.

⇒ bajarLinks:

Para describir este método, es necesario conocer algunos conceptos sobre URLs y cómo trabaja Java con ellas.

URL es la abreviatura de Uniform Resource Locator y es la referencia, dirección, a un recurso de Internet. Las URLs son la puerta de entrada a Internet y la Wordl Wide Web.

Una URL, o dirección, es en realidad un puntero a un determinado recurso de un determinado sitio de Internet. Se utilizan diversos tipos de URLs junto con protocolos de aplicación diferentes; los más habituales son el Hypertext Transfer Protocol (HTTP) y el File Transfer Protocol (FTP). Las URLs correspondientes a estos tipos de protocolo se emplean principalmente para identificar la ubicación de archivos, como páginas Web, soportar imágenes, archivos multimedia, archivos de texto y programas descargables. Las URLs de HTTP también hacen referencia a programas ejecutables, como secuencias de instrucciones CGI, son programas escritos normalmente en un lenguaje de instrucciones, que reciben una entrada y generan una salida de acuerdo con la especificación common gateway interface (CGI).

Al especificar una URL, se está indicando:

- ✓ El protocolo utilizado para acceder al servidor (http, por ejemplo)
- ✓ El nombre del servidor
- ✓ El puerto de conexión (opcional)
- ✓ El camino, y
- ✓ El nombre de un archivo determinado en el servidor (opcional)
- ✓ Un punto de referencia dentro del archivo (opcional)

A veces el nombre del archivo se puede omitir, ya que si se accede a través de un navegador, éste incorporará automáticamente el nombre de archivo index.html cuando no se indique ninguno, e intentará descargar ese archivo.

Además de indicar el archivo o página a la que se desea acceder, también es posible indicar una referencia, o ancla, que se haya establecido dentro de esa página.

Los programas Java que interactúan con Internet pueden usar URLs para buscar los recursos en Internet a los que se quiere acceder. El paquete java.net contiene la clase URL. Proporciona un conjunto de constructores que permiten construir fácilmente objetos URL y un conjunto de métodos de acceso que posibilitan realizar operaciones complejas de escritura y lectura utilizando URLs.

La clase URL brinda cuatro constructores URL que permiten crear objetos URL mediante diversos parámetros como el tipo de protocolo, el nombre del host, el puerto y la ruta de acceso. Estos parámetros pueden especificarse aparte o en forma de texto como parte de una cadena URL. La clase URL trata la ruta de acceso de un

archivo y su nombre como una entidad individual para facilitar el trabajo con componentes URL.

Se puede, construir una URL utilizando su dirección absoluta o una dirección relativa a otra URL. Una URL absoluta contiene toda la información necesaria para alcanzar al recurso en cuestión. La forma más simple de crear un objeto URL es crearlo a partir del String que representa la URL completa, absoluta:

URL unlp = new URL(“http://www.unlp.edu.ar/grado.htm”);

Una dirección relativa es una ruta de acceso/nombre de archivo o desplazamiento de archivo que se especifica relacionada con una URL absoluta.)

Las URLs relativas tienen sólo la información suficiente para alcanzar al recurso relativo en el contexto de otra URL. Las URLs relativas se suelen usar dentro de archivos HTML. Por ejemplo, en la página http://www.unlp.edu.ar/index.html hay links a otras páginas como grado.htm. Dentro de la página hay links a otras páginas que están en la misma máquina y el mismo directorio que la página principal. La página index.html tiene especificaciones como la siguiente:

<a href="grado.htm">

Por ejemplo, la URL absoluta http://www.unlp.edu.ar/index.html puede combinarse con la URL relativa grado.htm para obtener la URL http://www.unlp.edu.ar/grado.htm. Para crear esta URL en Java sería:

URL unlp = new URL(unlp_base, “grado.htm”);

El primer argumento es un objeto URL que especifica la base para la nueva URL, y el segundo argumento es un String que especifica el resto del nombre del recurso relativo a esta base.

Los métodos URL proporcionan un conjunto completo de capacidades de procesamiento de URL. Los métodos getProtocol(), getHost(), getPort(), getFile() y getRef() permiten determinar los componentes individuales de la dirección de una URL: protocolo, nombre del host, número del puerto, nombre del archivo y referencia. No todas las URL contienen estos componentes. La clase URL provee estos métodos porque las URLs HTTP contienen estos componentes y son los componentes URLs más comúnmente usados.

Una vez creada la URL, se puede llamar al método openStream() para obtener un stream del cual se puede leer el contenido de la URL. El método openStream()

devuelve un objeto java.io.inputStream(), por lo cual se puede leer de la URL usando los métodos normales de InputStream.

El método openConnection crea un objeto de la clase URLConnection para el objeto Web al que señala la URL, para iniciar comunicaciones con la URL. La clase URLConnection contiene varios métodos que permiten comunicarse con la URL a través de la red. Muchos de sus métodos son útiles sólo cuando se trabaja con URLs HTTP.

La clase URLConnection es abstracta, por lo que no se puede instanciar directamente, pero sí se puede extender. Una forma habitual de conseguir un objeto de tipo URLConnection es invocar un método sobre un objeto URL (ej. openConnection) que devuelva un objeto de la clase URLConnection

Tiene variables que contienen información muy útil sobre la conexión que se haya establecido y métodos que se pueden utilizar para examinar y manipular el objeto que se crea con la instanciación de la clase, entre ellos getLastModified() (para obtener la última fecha en que se ha modificado el archivo) y getContentType() (para obtener el tipo de contenido del archivo: texto/html, image/gif, image/jpeg).

El método bajarLinks recibe como parámetro un vector con las identificaciones de las URLs a grabar. Para cada una de ellas realiza las siguientes acciones:

1. Crea un objeto URL con la identificación y un objeto URLConnection
2. Obtiene el tipo de contenido del archivo de la URL (texto/html, image/gif, image/jpeg) para determinar si corresponde bajar según los seteos que haya realizado el usuario
3. Crea un archivo en el disco rígido para guardar la información de la URL
4. Si el tipo de archivo de la URL no es HTML

   Utilizando la clase inputStream lee de a bytes la URL y la guarda en el archivo creado en el disco rígido.

   Sino (es un html)

   Utilizando la clase inputStream lee de líneas la URL, realiza un análisis de cada línea en busca de links y referencias a otros archivos, modificándola para que el link o la referencia sea a un archivo local.

   Para cada línea:

   - ✓ Busca las palabras claves (href, src, background) incluidas en elementos html que puedan contener referencias o links. Las palabras

claves deben tener a continuación el signo "=" y la identificación de una URL.

- ✓ Chequea si la URL encontrada está precedida de la palabra "base". En este caso se trata de la URL que se utilizará para construir las URLs relativas
- ✓ Crea un objeto URL a partir de la identificación de la URL y la URL base y verifica su existencia usando el método existeURL
- ✓ Si existe la URL
  - obtiene el protocolo, el servidor, el tipo de archivo para controlar si debe grabarse en base a los seteos que haya realizado el usuario
  - agrega la URL al vector de URLs a bajar
  - genera un nombre para el archivo local en el que se guardará la URL
  - reemplaza en la línea del archivo HTML a la URL por el nombre del archivo local

⇒ existeURL: devuelve verdadero si la URL existe, falso caso contrario. Para ello se utiliza la clase HttpURLConnection. Esta clase extiende a URLConnection y tiene soporte de características HTTP específicas. El método getRespondeCode() permite obtener el código de status. Si la página existe devuelve HTTP_OK.

⇒ fechaURL: Devuelve la fecha de última modificación de la URL. Para ello se crea un objeto URLConnection y se utiliza el método getLastModified().

10. Ejemplo de utilización

10.1 Instalación del Browser Off Line

Para ejecutar el browser off line se requiere instalar la máquina virtual Java (JRE1.1.6) y la aplicación Java. Los pasos a seguir son:

1. Ejecutar el archivo jre1_1_6-007-win-i.exe ubicado en la carpeta JRE1.1.6 del CD
2. Ejecutar el archivo install.exe ubicado en la carpeta BrowserOffLine_instalador. Cuando el instalador solicite la selección de la máquina virtual Java a utilizar para la aplicación, seleccionar la instalada en el punto anterior (JRE 1.1.6).

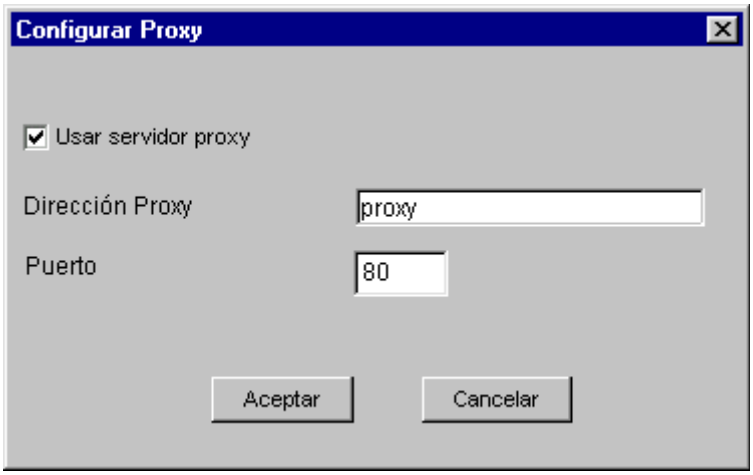
Se puede acceder al browser de las siguientes formas:

- ✓ Hacer click en el menú inicio, apuntar a la carpeta programas, luego apuntar a la carpeta Browser Off Line y hacer click en Browser off line
- ✓ O desde el explorador hacer click en el archivo BrowserOffLine.exe de la carpeta en la que se haya instalado.

10.2 Ejemplo

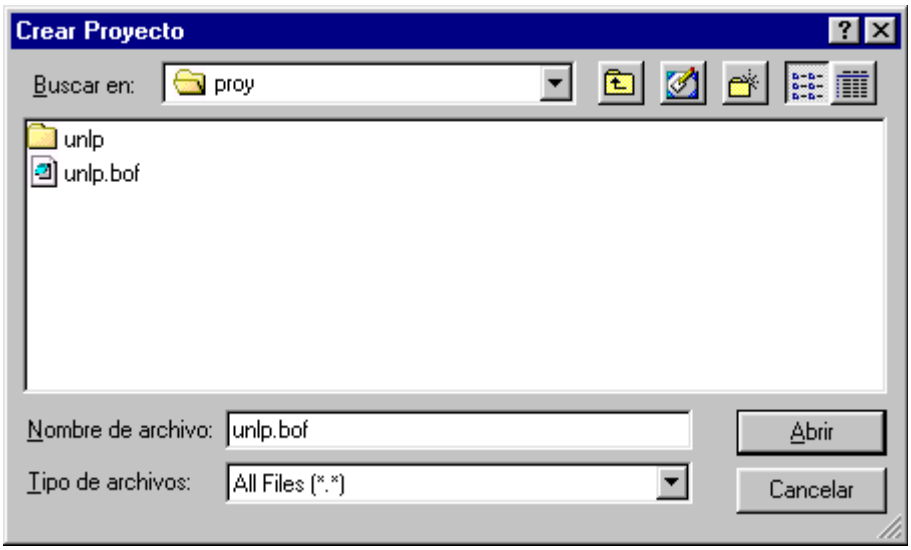
Para grabar un sitio hay que seguir los siguientes pasos:

- ✓ Configurar el proxy en el caso que se esté accediendo a Internet a través de una red con un servidor proxy. Para ello se debe seleccionar la opción Configurar proxy del menú Editar. Seleccionar "Usar servidor proxy" e ingresar la dirección y el puerto del mismo.



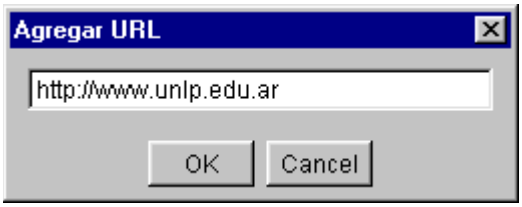
- ✓ Crear un proyecto nuevo. Para ello debe se debe seleccionar la opción nuevo del menú proyecto o hacer click en el botón de la barra de herramientas  .

  Se presenta el cuadro de diálogo para seleccionar el directorio donde se creará el nuevo proyecto e ingresar el nombre de éste.

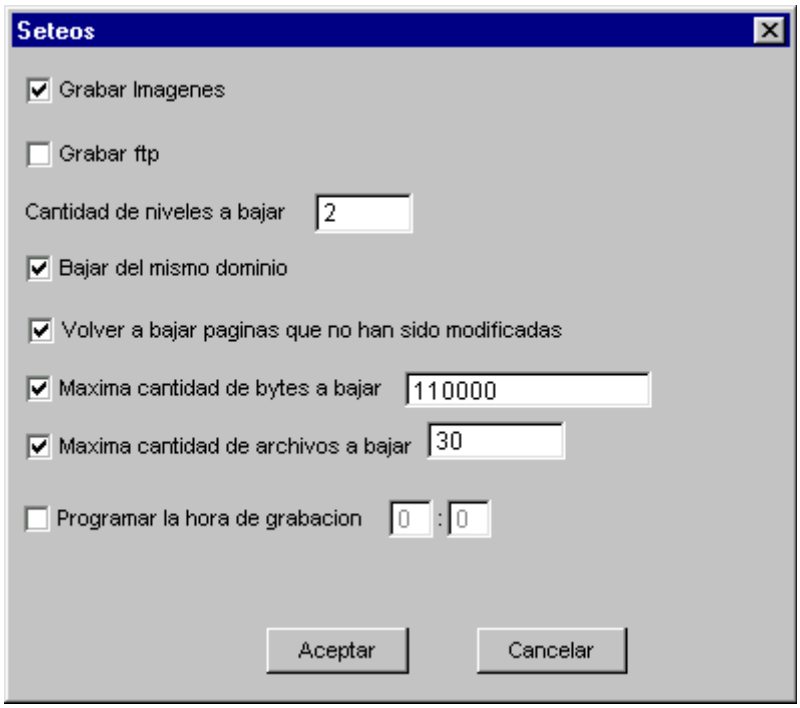


Una vez ingresado estos datos hacer click en “Abrir” para confirmar o en “Cancelar” si no se desea crearlo.

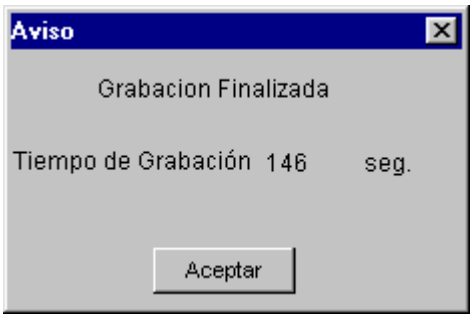
- ✓ Agregar las URLs que se desean grabar. Para ello hacer click en la opción “Agregar URL” del menú Editar o en el botón de la barra de herramientas  .



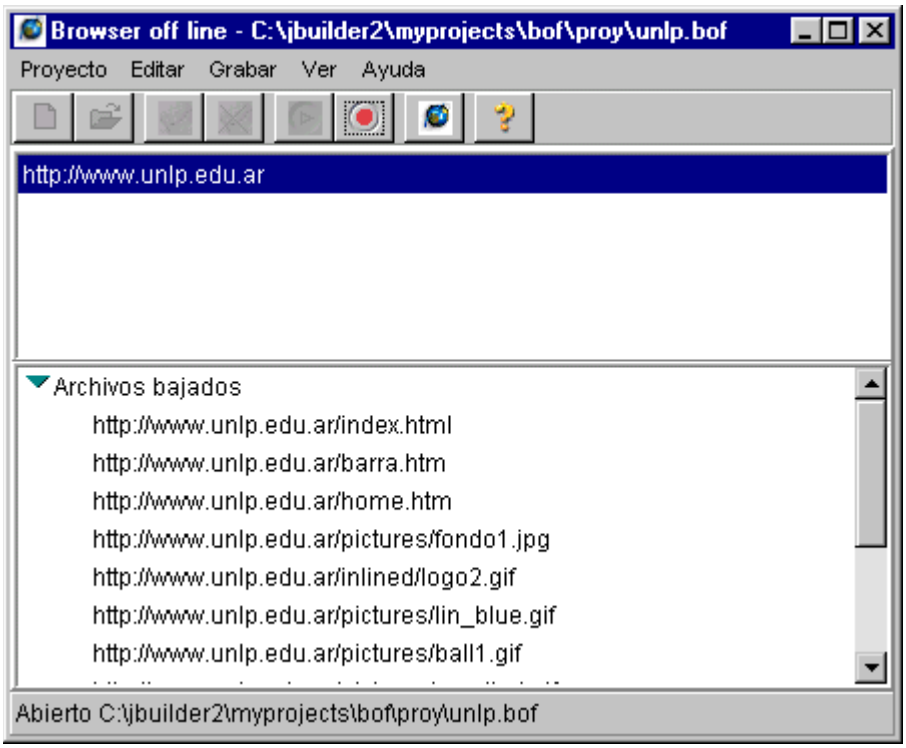
- ✓ Editar los seteos propios de proyecto. Para ello hacer click en la opción “Seteos” del menú Editar.



- ✓ Empezar a grabar. Para ello hacer click en la opción “Empezar” del menú Grabar o en el botón de la barra de herramientas  .
  Cuando finaliza la grabación aparece la siguiente ventana mostrando el tiempo de grabación



En el panel inferior de la pantalla principal se verán todos los archivos grabados



- ✓ Navegar off line. Para ello hacer click en la opción “URL” del menú Ver o en el botón de la barra de herramientas  .

11. Posibles Mejoras

- ✓ Incorporar wizard y tutorial
- ✓ Posibilidad de tener varios proyectos abiertos

12. Dificultades

- ❑ Métodos no documentados. Ej. Para configurar el proxy
- ❑ Métodos de Clases que no se comportan como deberían. Ej
  - ⇒ método setFilenameFilter (“*.java”) de la clase Filer
  - ⇒ Clase Calendar, al usar getMinutes() y getHours() al compilar informa que son obsoletos; al usar los métodos sugeridos devuelven siempre el mismo minuto y hora.
- ❑ Trabajar con un trial que tiene vencimiento 90 días. Al reinstalar porque se venció, detectó que se reinstalaba imposibilitando su uso, por lo cual tuve que formatear el disco rígido.
- ❑ El planteo del proyecto lo hice con Valeria Octtinger, que abandonó la carrera poco tiempo después, por lo cual la investigación del lenguaje y el desarrollo del sistema tuve que realizarlo sola.

13. Bibliografía

13.1. JAVA

- ✓ Java – La programación del futuro. Angel Lopez. MP Ediciones.
- ✓ Teach Yourself JAVA in 21 Days. Laura Lemay y Charles Perkins. Sams Net.
- ✓ JAVA Guía de desarrollo. Jamie Jaworski. Prentice Hall
- ✓ JAVA 1.2 Al descubierto. Jamie Jaworski. Prentice Hall
- ✓ The source for Java Technolohy. http://java.sun.com/
- ✓ JBuilder Home Page. http://www.borland.com/jbuilder/
- ✓ Tutorial de Java. http://members.es.tripod.de/froufe/
- ✓ Zerog. InstallAnywhere. http://www.zerog.com

13.2. HTML

- ✓ Dave Raggett's Introduction to HTML. http://www.w3.org/MarkUp/Guide/
- ✓ HTML 4.0 Specification. http://www.w3.org/TR/REC-html40/
- ✓ Tutorial de HTML en español. http://dns.uncor.edu/info/tutorial.htm

13.3. BROWSERS OFF LINE

- ✓ Grab-A-Site. http://www.bluesquirrel.com/products/grabasite/grabasite.html
- ✓ WebWhacker. http://www.bluesquirrel.com/products/whacker/whacker.html
- ✓ Anawave WebSnake The World's Most Powerful Off-Line Browser. http://www.e-mailsoftware.com/web.snake.htm
- ✓ Tympani - Information Management & Offline Web Search Agents. http://www.tympani.com
- ✓ Go-Get-It - http://softwareforfree.com/ggi/gogetit.html
- ✓ NCSA. A beginner' s guide to HTML - http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimerAll.html