

**UNIVERSIDAD CATOLICA DE COLOMBIA
FACULTAD DE INGENIERIA DE SISTEMAS**

CURSO: JAVA BASICO
PROFESOR: EMERSON CASTAÑEDA SANABRIA

TEMA: Applets

OBJETIVOS:

- Definir que es un applet.
- Comprender el funcionamiento y el ciclo de vida de los Applets.
- Proporcionar los elementos necesarios sobre HTML para la puesta en marcha de los Applets.
- Crear un applet y ejecutarlo.

CONTENIDO:

1. Definición
2. Ciclo de Vida de un Applet
3. Elementos de HTML
4. Creación de un Applet básico

DESARROLLO:

1. Definición

La definición más extendida de **applet**, indica que un applet es *"una pequeña aplicación accesible en un servidor Internet, que se transporta por la red, se instala automáticamente y se ejecuta localmente como parte de un documento web"*. Claro que así la definición establece el entorno (Internet, Web, etc.). En realidad, un applet es una aplicación normalmente corta basada en un formato gráfico sin representación independiente: es decir, se trata de un elemento a embeber en otras aplicaciones; es un componente en su sentido estricto.

De esta manera se puede afirmar que un applet es una mínima aplicación Java diseñada para ejecutarse en un navegador Web. Por tanto, no necesita preocuparse por un método *main()* ni en dónde se realizan las llamadas. El applet asume que el código se está ejecutando dentro de un navegador.

El appletviewer es una herramienta del jdk, que se asemeja al mínimo navegador. Espera como argumento el nombre del archivo *html* que debe cargar, no se le puede pasar directamente un programa Java. Este archivo html debe contener una marca que especifica el código que cargará el appletviewer:

```
<HTML>
<APPLET CODE="HolaMundo.class" WIDTH="300" HEIGHT="100">
</APPLET>
</HTML>
```

El appletviewer crea un espacio de navegación, incluyendo un área gráfica, donde se ejecutará el applet, entonces llamará a la clase applet apropiada. En el ejemplo anterior, el appletviewer cargará una clase de nombre HolaMundo y le permitirá trabajar en su espacio gráfico.

2. Ciclo de Vida de un Applet

Cuando un applet se carga, comienza su ciclo de vida, que pasa por cada una de las siguientes fases:

- Se crea una instancia de la clase que controla el applet.
- El applet se inicializa.
- El applet comienza a ejecutarse.
- El applet empieza a recibir llamadas. Primero recibe una llamada *init* (inicializar), seguida de un mensaje *start* (empezar) y *paint* (pintar). Estas llamadas pueden ser recibidas asíncronamente.

init()

El método *init()* se llama cada vez que el appletviewer o el navegador carga por primera vez la clase. Si el applet llamado no lo sobrecarga, *init()* no hace nada. Fundamentalmente en este método se debe fijar el tamaño del applet, aunque en el caso de Netscape el tamaño que vale es el que se indique en la línea del archivo html que cargue el applet. También se deben realizar en este método las cargas de imágenes y sonidos necesarios para la ejecución del applet. Y, por supuesto, la asignación de valores a las variables globales a la clase que se utilicen. En el caso de los applet, este método únicamente es llamado por el sistema al cargar el applet.

start()

start() es la llamada para arrancar el applet cada vez que es visitado. La clase **Applet** no hace nada en este método. Las clases derivadas deben sobrecargarlo para comenzar la animación, el sonido, etc. Esta función es llamada automáticamente cada vez que la zona de visualización en que está ubicado el applet se expone a la visión, a fin de optimizar en uso de los recursos del sistema y no ejecutar algo que no puede ser apreciado (aunque el programador puede variar este comportamiento y hacer que un applet siga activo aun cuando esté fuera del área de visión). Esto es, imaginemos que cargamos un applet en un navegador minimizado; el sistema llamará al método *init()*, pero no a *start()*, que sí será llamado cuando restauremos el navegador a un tamaño que permita ver el applet. Naturalmente, *start()* se puede ejecutar varias veces: la primera tras *init()* y las siguientes (porque *init()* se ejecuta solamente una vez) tras haber aplicado el método *stop()*.

stop()

stop() es la llamada para detener la ejecución del applet. Se llama cuando el applet desaparece de la pantalla. La clase **Applet** tampoco hace nada en este método, que debería ser sobrecargado por las clases derivadas para detener la animación, el sonido, etc. Esta función es llamada cuando el navegador no incluye en su campo de visión al applet; por ejemplo, cuando abandona la página en que está insertado, de forma que el programador puede paralizar los threads que no resulten necesarios respecto de un applet no visible, y luego recuperar su actividad mediante el método *start()*.

destroy()

El método *destroy()* se llama cuando ya no se va a utilizar más el applet, cuando se necesita que sean liberados todos los recursos dispuestos por el applet, por ejemplo, cuando se cierra el navegador. La clase **Applet** no hace nada en este método. Las clases derivadas deberían sobrecargarlo para hacer una limpieza final. Los applet multithread deberían utilizar *destroy()* para detener los threads que quedasen activos.

El appletviewer también contiene la clase **Component** (componente), que usa dos métodos para ayudar al applet a escribir en el espacio gráfico que el appletviewer le proporciona para su ejecución.

paint(Graphics g)

Es la función llamada cada vez que el área de dibujo del applet necesita ser refrescada. La clase **Applet** simplemente dibuja un rectángulo gris en el área, es la clase derivada, obviamente, la que debería sobrecargar este método para representar algo inteligente en la pantalla. Cada vez que la zona del applet es cubierta por otra ventana, se desplaza el applet fuera de la visión o el applet cambia de posición debido a un redimensionamiento del navegador, el sistema llama automáticamente a este método, pasando como argumento un objeto de tipo Graphics que delimita la zona a ser pintada; en realidad se pasa una referencia al contexto gráfico en uso, y que representa la ventana del applet en la página web.

update(Graphics g)

Esta es la función que realmente se llama cuando se necesita una actualización de la pantalla. La clase **Applet** simplemente limpia el área y llama al método *paint()*. Esta funcionalidad es suficiente para la mayoría de los casos; aunque, de cualquier forma, las clases derivadas pueden sustituir esta funcionalidad para sus propósitos especiales. Es decir, en las situaciones detalladas anteriormente que dañan la zona de exposición del applet, el sistema llama al método *paint()*, pero en realidad la llamada se realiza al método *update()*, cuyo comportamiento establecido en la clase **Component** es llamar al método *paint()*, tras haber rellenado la zona del applet con su color de fondo por defecto. Pudiera parecer así que se trata de un método de efecto neutro, pero si la función *paint()* cambiara el color del fondo, podríamos percibir un *flick* de cambio de colores nada agradable. Por tanto, habrá que cuidarse por lo común, de eliminar este efecto de limpieza primero, sobrecargando el método *update()*, para que llame únicamente a *paint()*. Otra solución sería insertar el código de pintado en una sobrecarga del método *update()* y escribir un método *paint()* que sólo llame a *update()*. La última solución pasaría por usar el mismo método *setBackground(Color)*, en el método *init()* para así evitar el efecto visual sin tener que sobrecargar el método *update()*. Estas son las mismas razones que aconsejan usar el método *resize()* inserto en *init()*, para evitar el mismo desagradable efecto.

repaint

Llamando a este método se podrá forzar la actualización de un applet, la llamada a *update()*. Pero hay que tener cierto cuidado, porque AWT posee cierta inteligencia (combinación casi siempre nefasta), de forma que si se llama a *update()* mediante *repaint()* con una frecuencia muy corta, AWT ignorará las llamadas a *update()* que estime oportuno, pues considera a esta función como un bien escaso.

3. Elementos de HTML

Definición:

HTML es una implementación del standard SGML (Standard Generalized Markup Language), estándar internacional para la definición de texto electrónico independiente de dispositivos, sistemas y aplicaciones. Un Metalenguaje para definir lenguajes de diseño descriptivos; proporciona un medio de codificar documentos hipertexto cuyo destino sea el intercambio directo entre sistemas o aplicaciones.

Características:

- Permite crear lenguajes de codificación descriptivos.
- Define una estructura de documentos jerárquica, con elementos y componentes interconectados.
- Proporciona una especificación formal completa del documento.
- No tiene un conjunto implícito de convenciones de señalización. Soporta, por tanto, un conjunto flexible de juegos de etiquetas.
- Los documentos generados por él son legibles.

Sintaxis general:

- Son válidos todos los caracteres incluidos en ISO 8859-1
- El formato es libre. El formato introducido en el archivo fuente (saltos de línea, líneas en blanco, etc.) es irrelevante para el formato final del documento.
- Caracteres de significado especial:
 - < Marca el comienzo de una etiqueta.
 - > Marca el final de una etiqueta.

Estructura de un documento HTML

Un documento HTML consta de las siguientes partes:

1. Identificación SGML
2. Una etiqueta <HTML>
3. Cabecera (iniciada por la etiqueta <HEAD> y cerrada por </HEAD>)
4. Cuerpo del documento (iniciada por la etiqueta BODY y cerrada por </BODY>)
5. Una etiqueta de fin de documento </HTML>

```
<!DOCTYPE PUBLIC HTML "-//IETF/DDT HTML 2.0/EN">
<HTML>
  <HEAD>
    ...
  </HEAD>
  <BODY>
    ...
  </BODY>
</HTML>
```

Identificación SGML:

Permite identificar la DTD adecuada para procesar el documento. No es obligatorio.

Cabecera:

Es un conjunto sin orden con información acerca del documento. Se identifica con la etiqueta <HEAD> y finaliza por tanto con </HEAD>

En su ámbito se pueden emplear diferentes elementos referenciados por sus etiquetas, los más relevantes son:

<TITLE> [cadena de caracteres] </TITLE> que da el título al documento, en la mayoría de los navegadores se visualiza en la barra de título

<BASE HREF="Url"> que indica la localización de los archivos, gráficos, sonidos, etc. a los que se hace referencia en nuestra página web. Si no se incluye esta directiva, el navegador entiende que los elementos se encuentran en el mismo lugar que nuestra página.

Cuerpo del documento:

Es el contenedor de la información propia del documento.

Se identifica con la etiqueta **<BODY [BACKGROUND="url de imagen"] [BGCOLOR="color"]>** y finaliza por tanto con la etiqueta **</BODY>**.

BACKGROUND: Url de la imagen de fondo de la pagina

BGCOLOR: Color de fondo de la pagina en RGB

En su ámbito se pueden emplear los elementos referenciados por las siguientes etiquetas:

- Cabeceras
- Elementos con estructura de bloque
- Elementos generales
- Etiquetas de resaltado y control de fuentes texto
- Listas
- Resaltados
- Formularios

Elementos del cuerpo del documento:

Cabeceras:

Existen 6 niveles de cabeceras:

Cabecera de nivel 1: **<H1>Texto de la cabecera</H1>**

Cabecera de nivel 2: **<H2>Texto de la cabecera</H2>**

Cabecera de nivel 3: **<H3>Texto de la cabecera</H3>**

Cabecera de nivel 4: **<H4>Texto de la cabecera</H4>**

Cabecera de nivel 5: **<H5>Texto de la cabecera</H5>**

Cabecera de nivel 6: **<H6>Texto de la cabecera</H6>**

El formato en que se visualizan las cabeceras depende de su nivel, variando: Tamaño de la letra, Tipo de resaltado, y el numero de líneas a saltar antes y después del texto.

Bloques de texto:

Definen la estructura de un bloque: **<P>** párrafo. Conjunto de texto que empieza y acaba con un salto de línea, la etiqueta de finalización no es obligatoria. Admite el parámetro **ALIGN=(left|center|right)** Para definir la alineación del texto dentro del bloque

<PRE> Texto con formato previo. Conjunto de texto que se muestra como se introdujo en el formato original.

<ADDRESS>Dirección. Información sobre el autor del documento, dirección, etc.

<BLOCKQUOTE>Anotación. Sirve para escribir una cita, el texto se presenta indentado y en un formato distinto al del párrafo normal.

<DIV> Permite agrupar varios bloques de texto en uno solo, heredando todos ellos la alineación especificada mediante el parámetro **ALIGN=(left|center|right)**

Elementos generales:

**
** Rotura de línea. Fuerza que se parta una línea de texto independientemente del formato en que se este trabajando.

<HR WIDTH= ALIGN= SIZE= > Línea horizontal. Dibuja una línea horizontal.

WIDTH= Tamaño en puntos o porcentaje

ALIGN= Alineación en la pagina: LEFT, RIGTH

SIZE= Grosor de la línea.

<CENTER> Centrar un texto. Centra en texto en la pagina

Etiquetas de resaltado de textos:

Se utilizan para enfatizar o resaltar una zona del texto.

La Etiqueta Applet de HTML

Dado que los applets están mayormente destinados a ejecutarse en navegadores Web, había que preparar el lenguaje HTML para soportar Java, o mejor, los applets. El esquema de etiquetas de HTML, y la evolución del estándar marcado por Netscape hicieron fácil la adición de una nueva marca que permitiera, una vez añadido el correspondiente código gestor en los navegadores, la ejecución de programas Java en ellos.

La sintaxis de las etiquetas <APPLET> y <PARAM> es la que se muestra a continuación y que iremos explicando en párrafos posteriores:

```
<APPLET CODE= WIDTH= HEIGHT= [CODEBASE=] [ALT=] [NAME=] [ALIGN=] [VSPACE=]
[HSPACE=]>
    <PARAM NAME= VALUE= >
</APPLET>
```

Atributos obligatorios:

CODE : Nombre de la clase principal

WIDTH : Anchura inicial

HEIGHT : Altura inicial

Atributos opcionales:

CODEBASE : URL base del applet

ALT : Texto alternativo

NAME : Nombre de la instancia

ALIGN : Justificación del applet

VSPACE : Espaciado vertical

HSPACE : Espaciado horizontal

Los Applets se incluyen en las páginas web a través de la etiqueta <APPLET>.

Atributos de los Applets

Los atributos que acompañan a la etiqueta <APPLET>, algunos son obligatorios y otros son opcionales. Todos los atributos, siguiendo la sintaxis de html, se especifican de la forma: *atributo=valor*.

Los **atributos obligatorios** son:

CODE Indica el archivo de clase ejecutable, que tiene la extensión *.class*. No se permite un URL absoluto, como ya se ha dicho, aunque sí puede ser relativo al atributo opcional **CODEBASE**.

WIDTH Indica la anchura inicial que el navegador debe reservar para el applet en pixels.

HEIGHT Indica la altura inicial en pixels. Un applet que disponga de una geometría fija no se verá redimensionado por estos atributos. Por ello, si los atributos definen una zona menor que la que el applet utiliza, únicamente se verá parte del mismo, como si se visualiza a través de una ventana, eso sí, sin ningún tipo de desplazamiento.

Los **atributos opcionales** que pueden acompañar a la marca APPLET comprenden los que se indican a continuación:

CODEBASE Se emplea para utilizar el URL base del applet. En caso de no especificarse, se utilizará el mismo que tiene el documento.

ALT Como ya se ha dicho, funciona exactamente igual que el ALT de la marca , es decir, muestra un texto alternativo, en este caso al applet, en navegadores en modo texto o que entiendan la etiqueta APPLET pero no implementen una máquina virtual Java.

NAME Otorga un nombre simbólico a esta instancia del applet en la página que puede ser empleado por otros applets de la misma página para localizarlo. Así, un applet puede ser cargado varias veces en la misma página tomando un nombre simbólico distinto en cada momento.

ALIGN Se emplea para alinear el applet permitiendo al texto fluir a su alrededor. Puede tomar los siguientes valores: LEFT, RIGHT, TOP, TEXTTOP, MIDDLE, ABSMIDDLE, BASELINE, BOTTOM y ABSBOTTOM.

VSPACE Indica el espaciado vertical entre el applet y el texto, en píxeles. Sólo funciona cuando se ha indicado ALIGN = LEFT o RIGHT.

HSPACE Funciona igual que el anterior pero indicando espaciado horizontal, en píxeles. Sólo funciona cuando se ha indicado ALIGN = LEFT o RIGHT.

Es probable encontrar en algunas distribuciones otras etiquetas para la inclusión de Applets, como <APP>. Esto se debe a que estamos ante la tercera revisión de la extensión de HTML para la incrustación de Applets y ha sido adoptada como la definitiva. Por ello, cualquier otro medio corresponde a implementaciones obsoletas que han quedado descartadas.

4. Creación de un Applet básico

Vamos a comenzar la creación del código fuente de un applet que satisfaga nuestras necesidades. Recordamos que Java utiliza la extensión .java para designar los archivos fuente.

HolaMundo

A continuación está el código fuente del applet HolaMundo, que es la versión applet de la mínima aplicación Java que antes habíamos escrito. Guardar este código en un archivo fuente Java como HolaMundo.java.

```
//
// Applet HolaMundo de ejemplo
//
import java.awt.Graphics;
import java.applet.Applet;

public class HolaMundo extends Applet {
    public void paint( Graphics g ) {
        g.drawString( "Hola Mundo!",25,25 );
    }
}
```

Componentes básicos de un Applet

```
/*
Sección de importaciones
*/

public class NombreDelNuevoApplet extends Applet {
    /*
    Aquí se declaran las variables de estado (public y private) o atributos
    */

    /*
    Los métodos para la interacción con los objetos se
    declaran y definen aquí
    */
    public void MetodoUno( parámetros ) {
        /*
        Aquí viene para cada método, el código Java que
        desempeña la tarea.
        Qué código se use depende del applet
        */
    }
}
```

Para HolaMundo, se importan las dos clases que necesita. No hay variables de estado, y sólo se tiene que definir un método para que el applet tenga el comportamiento esperado.

Clases incluidas

El comando `import` carga otras clases dentro del código fuente. El importar una clase desde un paquete de Java hace que esa clase importada esté disponible para todo el código incluido en el archivo fuente Java que la importa. Por ejemplo, en el applet `HolaMundo` se importa la clase `java.awt.Graphics`, y así se pueden llamar a los métodos de esta clase desde cualquier método del programa que se encuentre en el archivo `HolaMundo.java`. Esta clase define una área gráfica y métodos para poder dibujar dentro de ella. La función `paint()` declara a `g` como un objeto de tipo `Graphics`; luego, `paint()` usa el método `drawString()` de la clase **Graphics** para generar su salida.

La clase Applet

Se puede crear una nueva clase, en este caso **HolaMundo**, extendiendo la clase básica de Java: **Applet**. De esta forma, se hereda todo lo necesario para crear un applet. Modificando determinados métodos del applet, podemos lograr que lleve a cabo las funciones que deseamos.

```
import java.applet.Applet;
. . .
public class HolaMundo extends Applet {
```

Métodos de Applet

La parte del applet a modificar es el método `paint()`. En la clase **Applet**, se llama al método `paint()` cada vez que el método arranca o necesita ser refrescado, pero no hace nada. En nuestro caso, lo que hacemos es:

```
public void paint( Graphics g ) {
    g.drawString( "Hola Mundo!",25,25 );
}
```


De acuerdo a las normas de sobrecarga, se ejecutará este último *paint()* y no el *paint()* vacío de la clase **Applet**. Luego, aquí se ejecuta el método *drawString()*, que le dice al applet cómo debe aparecer un texto en el área de dibujo. Otros métodos básicos para dibujar son:

```
drawLine( int x1,int y1,int x2,int y2 )
drawRect( int x,int y,int ancho,int alto )
drawOval( int x,int y,int ancho,int alto )
```

Tanto para *drawRect()* como para *drawOval()*, las coordenadas (x,y) son la esquina superior izquierda del rectángulo (para *drawOval*, el óvalo es encajado en el rectángulo que lo circunscribe).

Paso de parametros a Applets

El espacio que queda entre las marcas de apertura y cierre de la definición de un applet, se utiliza para el paso de parámetros al applet. Para ello se utiliza la marca PARAM en la página HTML para indicar los parámetros y el método *getParameter()* de la clase *java.applet.Applet* para leerlos en el código interno del applet. La construcción puede repetirse cuantas veces se quiera, una tras otra. Los atributos que acompañan a la marca PARAM son los siguientes:

NAME Nombre del parámetro que se desea pasar al applet.

VALUE Valor que se desea transmitir en el parámetro que se ha indicado antes.

Texto HTML Texto HTML que será interpretado por los navegadores que no entienden la marca APPLET en sustitución del applet mismo.

Para mostrar esta posibilidad se hace una modificación sobre el applet básico HolaMundo para que pueda saludar a cualquiera. Lo que se hará será pasarle al applet el nombre de la persona a quien se quiere saludar. Se genera el código para ello y se guarda en el archivo HolaTal.java

```
import java.awt.Graphics;
import java.applet.Applet;

public class HolaTal extends Applet {
    String nombre;

    public void init() {
        nombre = getParameter( "Nombre" );
    }

    public void paint( Graphics g ) {
        g.drawString( "Hola "+nombre+"!",25,25 );
    }
}
```

Si se compila el ejemplo se obtendrá el archivo HolaTal.class que se incluye en la página web. Hay que generar el archivo HolaTal.html, en el que se coloca el applet, y que debería tener el siguiente contenido:

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<APPLET CODE="HolaTal.class" WIDTH="300" HEIGHT="100">
<PARAM NAME="Nombre" VALUE="UCC">
</APPLET>
</BODY>
</HTML>
```

Por supuesto, que se puede sustituir el valor del parametro. Este cambio no afectará al código Java, no será necesario recompilarlo.

Los parámetros no se limitan a uno solo. Se puede pasar al applet cualquier número de parámetros y siempre hay que indicar un *nombre* y un *valor* para cada uno de ellos.

El método *getParameter()* es fácil de entender. El único argumento que necesita es el *nombre* del parámetro cuyo valor queremos recuperar. Todos los parámetros se pasan como Strings, en caso de necesitar pasarle al applet un valor entero, se ha de pasar como String, recuperarlo como tal y luego convertirlo al tipo que deseemos. Tanto el argumento de NAME como el de VALUE deben ir colocados entre dobles comillas (") ya que son String.

El hecho de que las marcas <APPLET> y <PARAM> sean ignoradas por los navegadores que no entienden Java, es inteligentemente aprovechado a la hora de definir un contenido alternativo a ser mostrado en este último caso. Así la etiqueta es doble:

```
<APPLET atributos>  
parámetros  
contenido alternativo  
</APPLET>
```

El archivo para mostrar el applet de ejemplo se modifica para que pueda ser visualizado en cualquier navegador y en unos casos presente la información alternativa y en otros, ejecute nuestro applet:

```
<HTML>  
<HEAD>  
</HEAD>  
<BODY>  
    <APPLET CODE="HolaTal.class" WIDTH="300" HEIGHT="100">  
        <PARAM NAME="Nombre" VALUE="UCC">  
            No se vera hasta conseguir un navegador <|>Java Compatible</|>  
    </APPLET>  
</BODY>  
</HTML>
```