



Java Avanzado

Java Applets

Copyright

- Copyright (c) 2004
José M. Ordax
- Este documento puede ser distribuido solo bajo los términos y condiciones de la Licencia de Documentación de javaHispano v1.0 o posterior.
- La última versión se encuentra en
<http://www.javahispano.org/licencias/>

¿Qué es un Applet Java?

- Es un tipo de aplicación Java que se inserta dentro de páginas HTML. Cuando estas se descargan, los Applets se ejecutan en el navegador.
- Los Applets Java se cargan de la siguiente forma:
 - Se escribe una URL en el navegador.
 - El navegador carga la página HTML.
 - El navegador carga las clases del Applet Java.
 - Se ejecuta el Applet Java.

Restricciones de seguridad

- Los Applets Java, a diferencia de las aplicaciones Java no se ejecutan siempre con el conocimiento del usuario. Podemos estar navegando y cargar un Applet Java sin saberlo.
- Por ello, sufren una serie de restricciones de seguridad no existentes en las aplicaciones Java convencionales.
- Los navegadores implementan un entorno de ejecución seguro habitualmente conocido con el nombre de SandBox.
- Existen técnicas de firma digital para poder evitar estas restricciones.

Restricciones de seguridad



Algunas restricciones son:



Realizar llamadas a programas externos.



Realizar operaciones de entrada/salida.



Realizar llamadas a métodos nativos (JNI).



Abrir conexiones con servidores remotos distintos al origen del propio Applet Java.



De esta manera, un Applet Java jamás podrá dañar o robar información del sistema.

Las clases Applet y JApplet



La clase Applet es un contenedor visual del estilo de los Frame y Panel o los JFrame y JPanel.



Existen dos implementaciones: AWT y Swing



AWT:



Se encuentra en el paquete `java.applet.*`



Hereda de `java.awt.Panel`



Se le aplican `LayoutManagers` y se le añaden componentes como ya hiciéramos con los contenedores estándar AWT. Ocurre lo mismo con la gestión de eventos.

Las clases Applet y JApplet



Swing:



Se encuentra en el paquete javax.swing.*



Hereda de java.applet.Applet



Se le aplican LayoutManagers y se le añaden componentes como ya hicieramos con los contenedores estándar Swing. Ocurre lo mismo con la gestión de eventos.

Ciclo de vida



A diferencia de las aplicaciones Java convencionales, los Applets Java no tienen un método: *public static void main(String[] args)* para ejecutarse.



Por el contrario tienen estos cinco métodos:



`public void init();`



`public void start();`



`public void stop();`



`public void destroy();`



`public void paint(Graphics g);`

Ciclo de vida



`public void init();`



Es el primer método que se ejecuta una vez que se ha llamado al constructor del applet.



Solo se ejecuta una vez en la vida del applet.



Se suele usar para la inicialización del applet.



`public void start();`



Se ejecuta una vez haya terminado el método `init()`.



A diferencia del método `init()`, `start()` se ejecuta cada vez que haya que arrancar el applet (reload, maximizar,...).



Se suele usar para arrancar procesos (ej: Threads).

Ciclo de vida



`public void stop();`



Se ejecuta cada vez que haya que parar el applet (reload, minimizar, cargar otra página,.....).



Se suele usar para parar procesos (ej: Threads).



`public void destroy();`



Se ejecuta al cerrar el Navegador o cuando este decide eliminarle (ej: purgado de caché).



Solo se ejecuta una vez en la vida del applet.



Se suele usar para liberar cualquier recurso utilizado.

Ciclo de vida



`public void paint(Graphics g);`



Recibe una instancia de la clase `java.awt.Graphics`



Esta instancia se puede utilizar para escribir o pintar en el applet.



Se ejecuta cada vez que se necesita refrescar el Applet (ej: reload, maximizar,...).

Ejemplo del Ciclo de vida



Carga del Applet:



Se ejecuta el constructor por defecto (sin parámetros) del applet.



Se ejecuta el método `init()`.



Se ejecuta el método `start()`.



Salida y vuelta a entrar en la página HTML:



Se ejecuta el método `stop()` al salir.



Se ejecuta el método `start()` al volver.



Cierre del navegador:



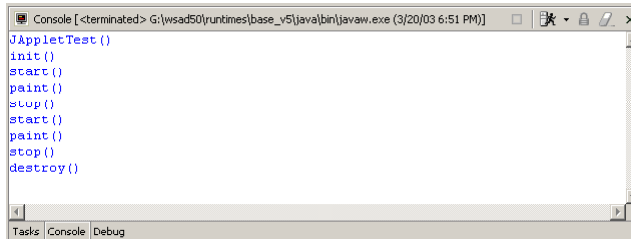
Se ejecuta el método `stop()`.



Se ejecuta el método `destroy()`.

Ejemplo del Ciclo de Vida

```
import java.awt.Graphics;  
import javax.swing.JApplet;  
  
public class JAppletTest extends JApplet  
{  
    public JAppletTest() { System.out.println("JAppletTest()"); }  
    public void init() { System.out.println("init()"); }  
    public void start() { System.out.println("start()"); }  
    public void paint(Graphics g) { System.out.println("paint()"); }  
    public void stop() { System.out.println("stop()"); }  
    public void destroy() { System.out.println("destroy()"); }  
}
```



HTML

- HTML – HyperText Markup Language
- Es un lenguaje simple utilizado para crear documentos Web de extensión *.html o *.htm
- Se limita a describir la estructura y el contenido de un documento.
- Actualmente se encuentra en su versión 4.01
- Es un lenguaje basado en etiquetas. Cada etiqueta se escribe entre <>.
- Las etiquetas pueden tener atributos.

Estructura básica

- Las etiquetas suelen ir formando bloques: `<>` y `</>`.
- Esqueleto básico:

```
<HTML>
<HEAD>
  <TITLE>Mi primer HTML</TITLE>
</HEAD>
<BODY>

</BODY>
</HTML>
```

La etiqueta <applet>

- Sintaxis:

```
<applet [name=?] [codebase=?] code=? [archive=?] width=? height=?
      [align=?] [vspace=?] [hspace=?] [alt=?]>
  [<param name=? value=?>
  .....
</applet>
```

- Descripción:

- name: especifica un nombre al applet.
- codebase: especifica la URL por defecto del applet en caso de que sea distinta a la del HTML.
- code: especifica el nombre de la clase principal del applet, incluyendo el paquete y la extensión (org.javahispano.MiApplet.class)

La etiqueta <applet>



Descripción (cont.):



archive: especifica los archivos JAR o ZIP que contengan clases necesarias y que deban ser precargados por el navegador.



width: especifica el ancho del applet en pixels.



height: especifica el alto del applet en pixels.



align: especifica el alineamiento del applet respecto del texto. Algunos valores válidos son: left, right, top, middle....



vspace: especifica el espacio a dejar por encima y por debajo del applet en pixels.



hspace: especifica el espacio a dejar por derecha e izquierda del applet en pixels.

La etiqueta <applet>



Descripción (cont.):



alt: especifica el texto a mostrar en caso de que el navegador no sea capaz de mostrar el applet.



<param>: es la manera de pasar parámetros al applet desde el exterior. Esta etiqueta tiene dos atributos: *name* para el nombre del parámetro y *value* para su valor.



Los applets cuentan con dos métodos para acceder a los parámetros (no usar en el constructor):



```
public String getParameter(String name);
```



```
public String[][] getParameterInfo();
```

Ejemplo





```
<HTML>
<HEAD>
  <TITLE>Mi primer HTML</TITLE>
</HEAD>
<BODY>

  <APPLET name="Test1" codebase="." code="org.javahispano.MiApplet.class"
          archive="miApplet.jar" width="200" height="300" align="middle"
          vspace="5" hspace="5" alt="No tiene soporte de Applets Java">
    <PARAM name="param1" value="value1">
    <PARAM name="parma2" value="value2">
  </APPLET>

  <APPLET code="org.javahispano.OtroApplet.class" width="200" height="300">
  </APPLET>

</BODY>
</HTML>
```

Probando los Applets

-  El SDK (o JDK) contiene una herramienta para probar los Applets Java sin necesidad de un navegador.
-  Esta herramienta se llama appletviewer.exe
-  Carga un fichero HTML como argumento:
appletviewer.exe HelloWorld.html
-  Ese fichero HTML requiere como mínimo:

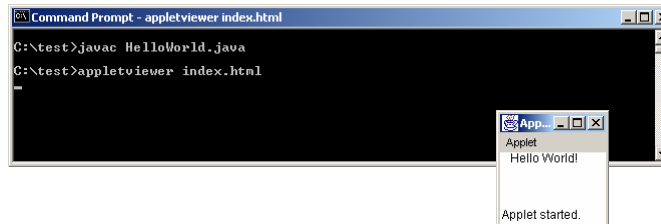
```
<HTML>
  <APPLET code="HelloWorld.class" width="200" height="300">
  </APPLET>
</HTML>
```

Ejemplo

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

```
public class HelloWorld extends Applet  
{  
    public void paint(Graphics g)  
    {  
        g.drawString("Hello World!", 10, 10);  
    }  
}
```

```
<HTML>  
<APPLET code="HelloWorld.class"  
        width="100" height="50">  
</APPLET>  
</HTML>
```



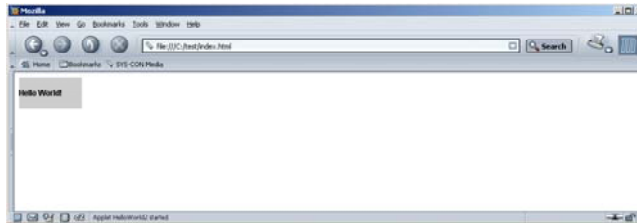
Probando los Applets

- No obstante podemos probar directamente en un navegador abriendo en local la página HTML:
file:///c:/test/index.html
- O desplegando la página HTML y el Applet Java a un Servidor Web:
http://www.javahispano.org/test/index.html

Ejemplo

```
import javax.swing.JApplet;  
import javax.swing.JLabel;  
  
public class HelloWorld2 extends JApplet  
{  
    public void init()  
    {  
        JLabel l = new JLabel("Hello World!");  
        this.getContentPane().add(l);  
    }  
}
```

<HTML>
<APPLET code="HelloWorld2.class" width="100" height="50">
</APPLET>
</HTML>



Otras posibilidades

 public AudioClip getAudioClip(URL url, String clip):

```
import java.applet.Applet;  
import java.applet.AudioClip;  
  
public class AudioClipTest extends Applet  
{  
    AudioClip clip = null;  
  
    public void init()  
    {  
        clip = this.getAudioClip(this.getDocumentBase(), "audio/test.au");  
    }  
  
    public void start() { clip.play(); }  
  
    public void stop() { clip.stop(); }  
}
```

Otras posibilidades






 `public Image getImage(URL url, String img):`

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

public class ImageTest extends Applet
{
    Image img = null;

    public void init()
    {
        img = this.getImage(this.getDocumentBase(),"images/test.gif");
    }
    public void paint(Graphics g)
    {
        g.drawImage(img, 10,20, this);
    }
}
```

El fichero JAR

-  Es un formato de fichero.
-  Empaqueta comprimidos en un único fichero clases Java y otro tipo de recursos (imágenes, sonidos,...).
-  El SDK (o JDK) contiene una herramienta para crear, listar o expandir archivos JAR.
-  Sintaxis:
 `jar.exe {ctxu} [vfm0M] [fichero-jar] [fichero-manifest] [-C dir] ficheros....`

El fichero JAR

```
Command Prompt
Usage: jar <ctxu>[vfm0M] [jar-file] [manifest-file] [-C dir] files ...
Options:
  -c create new archive
  -t list table of contents for archive
  -x extract named (or all) files from archive
  -u update existing archive
  -v generate verbose output on standard output
  -f specify archive file name
  -m include manifest information from specified manifest file
  -0 store only; use no ZIP compression
  -M do not create a manifest file for the entries
  -i generate index information for the specified jar files
  -C change to the specified directory and include the following file
If any file is a directory then it is processed recursively.
The manifest file name and the archive file name needs to be specified
in the same order the 'm' and 'f' flags are specified.

Example 1: to archive two class files into an archive called classes.jar:
  jar cvf classes.jar Foo.class Bar.class
Example 2: use an existing manifest file 'mymanifest' and archive all the
  files in the foo/ directory into 'classes.jar':
  jar cvfm classes.jar mymanifest -C foo/ .

C:\>
```

Bibliografía



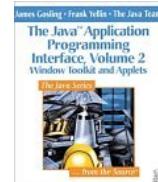
HTML & XHTML (5th edition).

Chuck Musciano y Bill Kennedy.
O'Reilly.



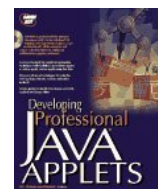
Window Toolkit and Applets, Volume 2

James Gosling y Frank Yellin.
Addison-Wesley.



Developing professional Java Applets

K.C.Hopson, Stephen E. Ingram y Patrick Chan.
Sams Publishing.



The Java tutorial (on-line)

<http://java.sun.com/docs/books/tutorial/applet/index.html>