

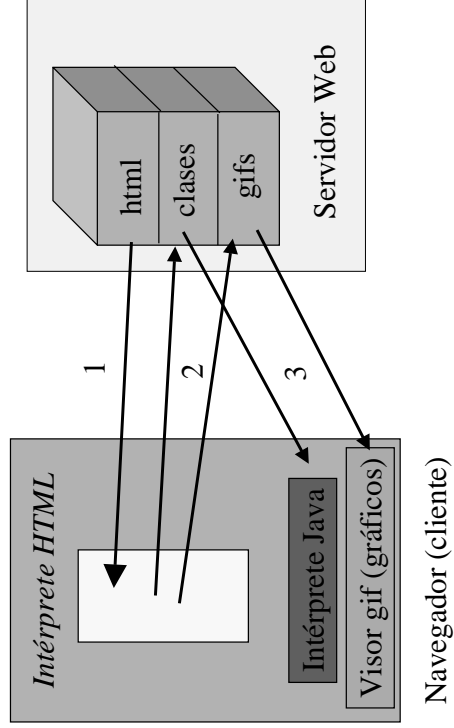


# Applets

**Baltasar Fernández Manjón**

Dpto. de Sistemas Informáticos y Programación,  
Universidad Complutense de Madrid  
Avda. Complutense s/n, 28040, Madrid, Spain.

# Ejecución de un applet

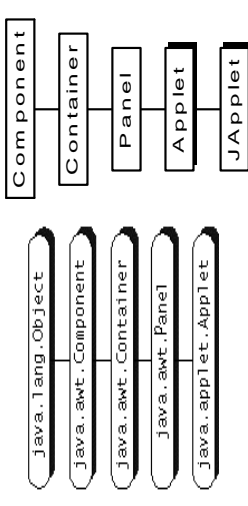


# Applet

- Un applet es:
  - una “pequeña” aplicación
  - accesible en un servidor Internet
  - que se transporta por la red
  - se instala automáticamente
  - se ejecuta *in situ* (en el cliente) como parte de un documento web
- Los applets se ejecutan dentro de otro programa, por tanto no son aplicaciones independientes
  - Navegador (e.g. Explorer, Navigator)
  - AppletViewer

# La clase `java.applet.Applet`

- Proporciona el framework para la ejecución del applet
  - Define métodos a los que llama el sistema cuando se producen determinados eventos en el ciclo de vida del applet
    - Sobreescribiendo estos métodos se obtiene el comportamiento deseado del applet
    - Proporciona el interfaz estándar de comunicación entre el applet y el entorno de ejecución
- Hay dos tipos principales de métodos
  - Dibujo - presentación en pantalla de imágenes o componentes de la interfaz
    - Gestión de eventos
      - Ciclo de vida
        - Inicio, parada, ...
      - Interacción del usuario
        - Raton, teclado



## Creación de un applet

- Por lo tanto, para hacer un applet Java hay que definir:
- Una clase Java (nuestro applet) que herede de Applet
  - En esta clase se redefinen uno o varios métodos para especificar qué hace el applet

```
public class HolaMundoApplet extends Applet {  
    .....  
}
```

- Una página HTML desde la que se llame al applet
  - De la misma manera que una página HTML puede incluir otros objetos (gráficos, etc.), también applets, con la etiqueta <APPLET>

```
<HTML>  
<APPLET CODE= "HolaMundo.class" WIDTH=200 HEIGHT= 100>  
</APPLET>  
</HTML>
```

Baltasar Fernández Manjón

5

## Ejemplo de applet

Hola.html

```
<HTML>  
<HEAD>  
<TITLE> Hola Mundo</TITLE>  
</HEAD>  
<BODY>  
Saluda amigablemente:  
<APPLET  
    ARCHIVE="AppletClasses.jar"  
    CODE="HolaMundoApplet.class"  
    WIDTH=150 HEIGHT=125>  
</APPLET>  
</BODY>  
</HTML>
```

HolaMundoApplet.java

```
import java.applet.Applet;  
import java.awt.*;  
  
public class HolaMundoApplet  
    extends Applet {  
    public void init() {  
        resize(150,25);  
    }  
    public void paint(Graphics g){  
        g.drawString("Hola Mundo", 50,25);  
    }  
}
```

Baltasar Fernández Manjón

6

## Métodos del ciclo de vida de un applet

- En general un applet pasa por varias fases en las que se invocan automáticamente distintos métodos
  - Se crea e inicializa
    - init()
  - Puede comenzar a ejecutarse
    - start()
  - Puede suspenderse su ejecución
    - stop()
  - Puede detenerse completamente su ejecución (destruirse)
    - destroy()
- Estos métodos definen lo que se hace cuando el applet cambia su estado de ejecución o existencia
- En general, no se define constructor para los applets

Baltasar Fernández Manjón

7

## Método init()

- *public void init()*
- Se llama una sola vez, al crearse el applet:
- Es el método que se ejecuta cuando se carga el applet por primera vez en el navegador y es donde se deben realizar todas las tareas de inicialización
  - o al recargarse la página
- Sirve para:
  - Normalmente es donde se construye la interfaz gráfica de usuario
  - En general siempre se reescribe este método con el código que en una aplicación normal se incluiría en el constructor
    - Por ejemplo, el tamaño del applet
      - aunque tiene prioridad el especificado en el html, y Netscape ignora la operación `resize(anchura, altura)`*
  - Inicializar variables globales al applet
- Examinar los parámetros del applet
- Cargar imágenes, sonidos, etc., de la red en memoria
- Por defecto no hace nada

Baltasar Fernández Manjón

8

## Método *start()*

---

- *public void start()*
- Es el método que se ejecuta después del método `init()` y cada vez que el usuario vuelve a la página web que contiene la applet.
  - Sirve para indicar a la applet que debe iniciar su ejecución normal. Debe reescribirse si hay alguna operación que se desea que se ejecute cada vez que se visite la página web que la contiene
  - se ejecuta siempre que la applet pasa a estar visible de modo que puede ejecutarse muchas veces en la vida de una applet.
- Se puede:
  - crear hilos de ejecución paralela (threads)
  - comenzar a reproducir música o imágenes
- Por defecto no hace nada

## Método *stop()*

---

- *public void stop()*
- Es el método que se ejecuta cada vez que el usuario abandona la página web que contiene la applet.
  - Permite detener o suspender temporalmente operaciones costosas cuando la applet no es visible.
  - Estas operaciones se deberían reanudar en el método `start()` para que la applet funcione correctamente.
  - Este método también se ejecuta antes de destruir la applet.
- Se puede
  - parar los hilos de ejecución paralela
  - parar la animación y la reproducción de sonidos
- Por defecto no hace nada

## Método *destroy()*

---

- *public void destroy()*
- Es el método que se ejecuta cuando se va a descargar o destruir completamente la applet.
- En este método se deberían liberar los recursos del sistema que pudieran estar asignados a la applet
  - destruir cualquier hilo de ejecución (thread) activo
  - cerrar conexiones a red
- Por defecto no hace nada

## Creación de una applet: Los pasos habituales

---

1. La clase principal de la applet debe heredar de la clase `Applet` (o `JApplet` para applets Swing) que le proporciona la comunicación con el entorno y la funcionalidad básica de ejecución.
2. Se debe definir el método `init()` para inicializar todos los elementos de la applet. Aquí se construye la interfaz gráfica de usuario que en las aplicaciones gráficas normales se incluía en el constructor.
3. Se pueden definir los métodos `start()`, `stop()` y `destroy()` para obtener el comportamiento deseado.
4. Se debe crear con HTML una página web que contenga a la applet.
  - La applet se ejecuta cuando se carga y visualiza en un navegador la página web que la contiene.
  - El navegador de Internet debe ser compatible con Java para que no se produzcan errores.
  - También se puede ejecutar directamente utilizando el visualizador de applets `AppletViewer` incluido en el JDK de Sun.

## Ejemplo de ciclo de vida

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

public class Simple extends Applet {
    StringBuffer buffer;

    public void init() {
        buffer = new StringBuffer();
        addItem("initializing...");
    }

    public void start() {
        addItem("starting...");
    }

    public void stop() {
        addItem("stopping...");
    }
}
```

```
public void destroy() {
    addItem("preparing for unloading...");
}

void addItem(String newWord) {
    System.out.println(newWord);
    buffer.append(newWord);
    repaint();
}

public void paint(Graphics g) {
    //Dibuja un rectángulo alrededor del applet
    g.drawRect(0, 0, size().width - 1,
        size().height - 1);
    //Dibuja el texto actual dentro del rectángulo
    g.drawString(buffer.toString(), 5, 15);
}
}
```

Baltasar Fernández Manjón

13

Baltasar Fernández Manjón

14

## Presentación o dibujo de los applets

- La presentación del applet en la página del navegador se realiza en un hilo de ejecución (thread) diferente denominado AWT thread
- Implica tres métodos
  - `paint(Graphics g)`
  - `update()`
  - `repaint()`

## Métodos de dibujo: *paint(Graphics g)*

- *public void paint(Graphics g)*
  - Método de mostrado básico que dibuja la presentación del applet en la página del navegador
  - El área gráfica `g` representa la ventana del applet en la página web (*paint()* es un método de la clase *Component*)
- Se llama para refrescar el área de dibujo del applet
  - después de la inicialización, si la ventana se ha escondido y luego vuelve al frente, o si se mueve la ventana del navegador
- Indica qué hace el applet
  - Dibuja en el área gráfica `g`
    - `g.drawString("Hola Mundo", 25, 25);`
- Por defecto no hace nada

Baltasar Fernández Manjón

15

Baltasar Fernández Manjón

16

## Métodos de dibujo: *update(Graphics g)*

- *public void update(Graphics g)*
  - Se llama para actualizar la pantalla
  - Prepara para pintar, y llama a *paint(g)*
    - Por defecto, limpia el rectángulo del *Component* y llama a *paint(g)*
      - `g.clear();`
      - `paint(g);`
- Se puede modificar cuando interesa mejorar la eficiencia
  - reducir el efecto de parpadeo producido al limpiar y repintar la pantalla

Baltasar Fernández Manjón

15

Baltasar Fernández Manjón

16

## Metodos de dibujo: *repaint()*

- *public void repaint()*
- Se llama
  - Cuando hace falta refrescar el contenido gráfico del applet porque se ha realizado algún cambio
- Provoca la ejecución de *update()*
- En general no se reescribe

## Programación de un applet mínimo

- Para que un applet haga algo debe reescribir por lo menos uno de los métodos que hereda de la clase Applet
- Es decir tiene que reescribir por lo menos uno de los siguientes métodos
  - *init()*
  - *start()*
  - *paint()*
- Un applet no necesita el método *main()*
- Si se incluye se ignora en la ejecución

## HTML y la etiqueta <APPLET>

```
<APPLET
[CODEBASE = URLdelCodigo]
[ARCHIVE = archivoDeClasesYrecursos]
CODE = FicheroClasePrincipalDelApplet
[ALT = textoAlternativoSiNoSePuedeEjecutarElApplet]
[NAME = nombreDelApplet]
WIDTH = pixelsAnchuraApplet
HEIGHT = pixelsAlturaApplet
[ALIGN = alineamientoDelApplet]
[SPACE = pixelsEncimaYDebajoDelApplet]
[HSRSPACE = pixelsAcadalaadoDelApplet]
>
[< PARAM NAME = nombreParametro1 VALUE = valorParametro1 >]
[< PARAM NAME = nombreParametro2 VALUE = valorParametro2 >]
...
[codigoHTMLalternativoAlApplet]
</APPLET>
```

## Uso de parámetros del HTML

- *getParameter(String)*
  - permite obtener desde el applet el valor del parámetro especificado en el fichero HTML a partir del nombre del parámetro (*NAME=String*)
  - devuelve el *valor* que se indica en *VALUE="valor"*
- **Atributos opcionales**
  - *CODEBASE* = "URL base del Applet"
    - Se usa cuando el código del applet no está en el mismo directorio que el html
  - *NAME* = "Nombre del ejemplar"
    - Se puede hacer *start()* de un applet guardado en forma serializada
    - Puede servir para comunicar applets de una misma página
  - *ALIGN* = LEFT | RIGTH | TOP | TEXTTOP | MIDDLE | BOTTOM
  - *ARCHIVE* = "archivo.jar"
  - *ARCHIVE=""*fichero1, fichero2"

## Ejemplo de uso de parámetros HTML

```
import java.awt.*;
import java.applet.Applet;
public class HolaColega extends java.applet.Applet {
    String colega;
    public void init() {
        colega = getParameter("nombre");
        if ( colega == null ) colega = "colega anónimo";
    }
    public void paint(Graphics g) {
        g.drawString("Hola " + colega, 10, 0);
    }
}
```

```
<APPLET CODE= "HolaColega.class" HEIGHT=100 WIDTH=200 >
<PARAM NAME= "nombre" value= "Juan" >
</APPLET>
```

Baltasar Fernández Manjón

21

Baltasar Fernández Manjón

22

## Otros métodos de los applets

- **getCodeBase()**
  - obtiene el URL base del applet
- **getDocumentBase()**
  - obtiene el documento URL en el que está embebido el applet
- **getAudioClip(URL)**
  - obtiene un clip audio que se puede reproducir con `play()`
- **getImage(URL)**
  - obtiene una imagen que se puede visualizar
- **play(URL)**
  - reproduce un clip audio de la red

Baltasar Fernández Manjón

21

Baltasar Fernández Manjón

22

## Otros métodos de los applets

- **getAppletInfo()**
  - devuelve una cadena con información sobre el autor, la versión y el copyright del applet. Hay que implementarlo.
    - Es útil por ejemplo para explicar lo que hace el applet o poner el nombre de su autor
- **getParameterInfo()**
  - devuelve un array de cadenas que describen los parámetros que entiende el applet. Hay que implementarlo.
- **showStatus(String)**
  - Muestra un mensaje de estado en el área reservada para ello en la ventana del navegador (por ejemplo, en Netscape, abajo a la izquierda).

Baltasar Fernández Manjón

23

Baltasar Fernández Manjón

24

## Métodos de gestión de eventos de usuario

- La gestión de eventos de usuario se realiza como en cualquier interfaz gráfica
  - La clase `java.applet.Applet` es una especialización de `java.awt.Panel`
  - Por tanto se dispone de todos los métodos de `Component`, `Container` y `Panel`
- Métodos para utilizar componentes GUI en applets
  - `add()` - Añade el componente especificado
    - `add(new Button("Boton"));`
  - `remove()` - Elimina el componente especificado
  - `setLayout()` - Establece el gestor de disposición (por defecto es `FlowLayout`)

Baltasar Fernández Manjón

23

Baltasar Fernández Manjón

24

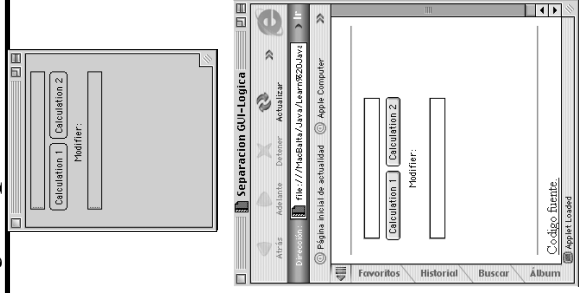
## Limitaciones de seguridad de las applets

- En general a las applets no se les permite hacer nada que pueda dañar el equipo cliente en el que se ejecutan
- Las applets cargadas de la red normalmente no pueden
  - Cargar bibliotecas o definir métodos nativos
  - Leer o escribir archivos locales del cliente
  - Realizar conexiones con otros servidores
  - Ejecutar programas en el cliente
  - Obtener determinadas propiedades del sistema
  - Además las ventanas emergentes de un applet tienen un aspecto diferente al de las aplicaciones
- Las applets locales y las applets con firma (seguras) no tienen tantas limitaciones de seguridad

## Permisos de las applets

Operation	Remote	Local	Appletviewer	Appletviewer
read local file	No	No	Yes	Yes
write local file	No	No	Yes	Yes
obtain file information	No	No	Yes	Yes
delete file	No	No	No	Yes
run local executable	No	No	Yes	Yes
load Java library	No	Yes	Yes	Yes
call exit	No	No	Yes	Yes
connect to other host	No	Yes	Yes	Yes

## Ejemplo



```

//: Separacion.java
// Separación entre la interfaz y la lógica de la aplicación
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

//lógica de la aplicación
class BusinessLogic {
    private int modifier;
    BusinessLogic(int mod) {
        modifier = mod;
    }
    public void setModifier(int mod) {
        modifier = mod;
    }

    public int getModifier() {
        return modifier;
    }

    // Algunas operaciones
    public int calculation1(int arg) {
        return arg * modifier;
    }
    public int calculation2(int arg) {
        return arg + modifier;
    }
}
    
```

```

// interfaz de la aplicación
public class Separation extends Applet {
    TextField
        t = new TextField(20),
        mod = new TextField(20);

    BusinessLogic bl = new BusinessLogic(2);
    Button
        calc1 = new Button("Calculación 1"),
        calc2 = new Button("Calculación 2");

    public void init() {
        add(t);
        calc1.addActionListener(new Calc1LO());
        calc2.addActionListener(new Calc2LO());
        add(calc1); add(calc2);
        mod.addActionListener(new ModLO());
        add(new Label("Modificar:"));
        add(mod);
    }

    static int getValue(TextField tf) {
        try {
            return Integer.parseInt(tf.getText());
        } catch (NumberFormatException e) {
            return 0;
        }
    }
}
    
```

```

class Calc1L implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        t.setText(Integer.toString(bl.calculation1(getValue(t))));}
}

class Calc2L implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        t.setText(Integer.toString(bl.calculation2(getValue(t))));}
}

class ModL implements TextListener {
    public void textValueChanged(TextEvent e) {
        bl.setModfier(getValue(mod)); }
}

// permite el doble comportamiento como aplicacion y como applet
// si se ejecuta como applet este código se ignora
public static void main(String[] args) {
    Separation applet = new Separation();
    JFrame aFrame = new JFrame(" Aplicacion Separation");
    aFrame.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
            System.exit(0);});
    aFrame.add(applet, BorderLayout.CENTER);
    aFrame.setSize(200,200);
    applet.init();
    applet.start();
    aFrame.setVisible(true); }
}

```

Baltasar Fernandez Vazquez

29

## Applet Swing

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;

public class CicloVidaApplet extends JApplet
{
    ArrayList lista;
    JButton boton;
    JPanel panel;
    public void init() {
        lista = new ArrayList();
        lista.add("Inicialización en init()");
        // creacion del interfaz de usuario
        Container contenido = getContentPane();
        contenido.add(new JLabel("Mi primera applet"),
            BorderLayout.NORTH);
        panel = new JPanel();
        contenido.add(panel, BorderLayout.CENTER);
        boton = new JButton("Boton");
        boton.addActionListener(new OyenteBoton());
        contenido.add(boton, BorderLayout.SOUTH);
    }
}

```

Baltasar

30

```

public void start() {
    lista.add("se ejecuta start()");
}

public void stop() {
    lista.add("se ejecuta stop()");
}

public void destroy() {
    lista.add("se ejecuta destroy()");
}

class MiPanel extends JPanel {
    public void paintComponent(Graphics g) {
        super.paintComponent(g); //pintado del fondo
        for (int i=10, cont=0; cont < lista.size(); cont++, i=i+15)
            g.drawString(lista.get(cont).toString(), 15,i);
    }
} // MiPanel

class OyenteBoton implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog(null,
            "Ha pulsado el botón", // mensaje
            "Mensaje informativo", // titulo
            JOptionPane.INFORMATION_MESSAGE); // icono
    }
} // OyenteBoton
} // CicloVidaApplet

```

Baltasar

31

