

## APPLICAZIONI & APPLET

- Java è un ottimo linguaggio per costruire applicazioni
  - anche non per Internet
  - anche non grafiche
- ma si è diffuso storicamente, e trae forza, dal concetto di *applet* come piccola (?) applicazione da eseguirsi dentro un browser Internet
  - grafica portatile ed eseguibile ovunque
  - modello di sicurezza "sandbox"

## APPLET

- Una *applet* ("applicazioncina") è una applicazione *non autonoma*, ma pensata per *far parte di una pagina Internet*
- Porta dinamicità alle pagine "statiche"
- Viene eseguita dal browser, che quindi deve incorporare un interprete Java
- Attenzione alla versione!
  - Molti browser supportano tuttora solo Java 1.1
  - Per essere flessibili, il JDK installa il "Java Plug in", che consente al browser di usare un interprete Java esterno (più aggiornato)

## APPLET

In quanto applicazione *non autonoma*, un'applet:

- non deve creare un frame principale, perché usa la finestra del browser che la ospita
- non ha un main, perché la sua vita è dipendente dalla pagina in cui è visualizzata
- è organizzata intorno a 4 metodi standard:
  - `init()`, che gioca il ruolo del costruttore
  - `start()` e `stop()`, chiamati dal browser ogni volta che occorre avviare /fermare l'applet
  - `destroy()`, invocato quando il browser viene chiuso.

## APPLET

In quanto applicazione *non autonoma*, un'applet:

- non deve creare un frame principale, perché usa la finestra del browser che la ospita
- non ha un main, perché la sua vita è dipendente dalla pagina in cui è visualizzata
- è organizzata intorno a 4 metodi standard:
  - `init()`, che gioca il ruolo del costruttore
  - `start()` e `stop()`, chiamati dal browser ogni volta che occorre avviare /fermare l'applet
  - `destroy()`, invocato quando il browser viene chiuso.

## APPLET

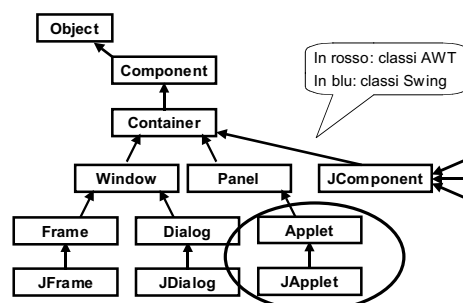
Esistono due versioni di Applet:

- la classe `Applet` dell'AWT standard (da Java 1.0 in poi)
- la classe `JApplet` di Swing (da Java 1.1.6 in poi)

Attenzione:

- se si usano componenti Swing, occorre necessariamente usare `JApplet`
- una `Applet` con componenti Swing non viene disegnata correttamente

## SWING: GERARCHIA DI CLASSI



## COSTRUIRE UN'APPLET

Per costruire un'Applet (con Swing), occorre:

- creare una propria classe che estenda JApplet
- implementare i metodi opportuni (tipicamente, almeno `init()`)
- compilare l'applet nel modo standard
- preparare una pagina HTML che la carichi
- eseguire l'applet tramite l'*AppletViewer*
  - oppure -
- aprire tale pagina con un browser

## ESEMPIO 1

```
import java.applet.*;
import java.awt.*;
import javax.swing.*;

public class Applet1 extends JApplet {
    Font f = new Font("Times", Font.BOLD, 36);
    public void paint(Graphics g) {
        g.setFont(f);
        g.setColor(Color.red);
        g.drawString("Ciao mondo!", 100, 50);
    }
}
```

Non implementa nessuno dei metodi `init()`, `start()`, `stop()`, ecc.

## ESEMPIO 1 - LA PAGINA HTML

```
<HTML><HEAD>
<TITLE> Applet Hello World </TITLE>
</HEAD> <BODY>
```

Questo e' cio' che produce la mia applet in un rettangolo 500 x 100 (la scritta e' alle coordinate 500,100 e si riferisce all'angolo inferiore sinistro della stringa): <P>

```
<APPLET CODE="Applet1.class"
WIDTH=500 HEIGHT=100 >
</APPLET>
</BODY>
</HTML>
```

Classe dell'applet

Dimensioni finestra (frame) dell'applet

## ESEMPIO 1 - TEST

**Compilazione:**

C:\prova> javac Applet1.java

**Esecuzione via appletviewer:**

C:\prova> appletviewer Applet1.html



## ESEMPIO 1 - TEST

**Alternativa: esecuzione via browser**

- **Problema:**  
*molto browser non supportano Swing*
- **Soluzione:** usare il Java Plug-in

Ma scrivere direttamente la pagina HTML che lo chiama è *molto complicato*  
→ si sfrutta il Convertitore HTML

Noi useremo semplicemente l'appletviewer.

## ESEMPIO 2

**Un'applet con tre pulsanti per cambiare il colore di sfondo**



- Il metodo `init()` fa le veci del costruttore: crea i componenti, installa i listener, etc.
- Non occorrono in questo caso gli altri metodi

## ESEMPIO 2

```
public class Applet2 extends JApplet {
    JButton redButton, blueButton, greenButton;
    JTextField messaggio;
    public void init() {
        Container c = getContentPane();
        c.setBackground(SystemColor.window);
        c.setLayout(new FlowLayout());
        redButton = new JButton("Rosso");
        blueButton = new JButton("Azzurro");
        greenButton = new JButton("Verde");
        messaggio = new JTextField(26);
        messaggio.setText("Premere un pulsante");
        messaggio.setEditable(false);
        ...
    }
}
```

## ESEMPIO 2

I componenti vanno aggiunti **non alla JApplet**, ma al **Container** che la contiene

```
...
c.add(messaggio); c.add(redButton);
c.add(blueButton); c.add(greenButton);
redButton.addActionListener(new
    Applet2Listener(this, Color.red, messaggio));
blueButton.addActionListener(new
    Applet2Listener(this, Color.cyan, messaggio));
greenButton.addActionListener(new
    Applet2Listener(this, Color.green, messaggio));
}
```

(segue la classe Applet2Listener)

## ESEMPIO 2

Il gestore degli eventi:

```
class Applet2Listener implements ActionListener {
    JApplet app; Color colore; JTextField txt;
    Applet2Listener(JApplet a, Color c, JTextField t){
        app = a; colore = c; txt = t;
    }
    public void actionPerformed(ActionEvent e){
        app.getContentPane().setBackground(colore);
        txt.setText("Premuto il pulsante " +
            e.getActionCommand());
        app.repaint();
    }
}
```

## ESEMPIO 2

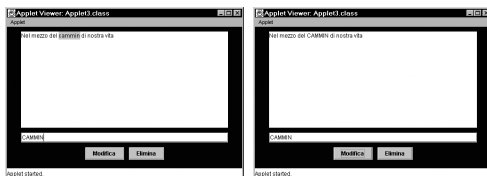
La pagina HTML:

```
<HTML><HEAD>
<TITLE> Applet 2 </TITLE>
</HEAD> <BODY>
<APPLET CODE="Applet2.class"
    WIDTH=300 HEIGHT=100 >
</APPLET>
</BODY>
</HTML>
```



## ESEMPIO 3

Un'applet che gestisce un'area di testo:



I pulsanti MODIFICA ed ELIMINA sostituiscono il testo selezionato con quello scritto nel campo di testo sotto.

## ESEMPIO 3

```
public class Applet3 extends JApplet {
    JButton deleteButton, editButton;
    JTextArea ta; JTextField tf;
    int pos, endPos;
    public void init() {
        Container c = getContentPane();
        c.setBackground(Color.blue);
        c.setLayout(new
            FlowLayout(FlowLayout.CENTER, 10, 10));
        deleteButton = new JButton("Elimina");
        editButton = new JButton("Modifica");
        ta = new JTextArea("", 12, 40);
        tf = new JTextField("", 40);
        ...
    }
}
```

Ricorda: si opera sul **Container** che contiene la **JApplet**, non direttamente su essa!

### ESEMPIO 3

```
...
c.add(ta); c.add(tf);
c.add(editButton); c.add(deleteButton);
Applet3Listener editor = new
    Applet3Listener(ta,tf);
editButton.addActionListener(editor);
deleteButton.addActionListener(editor);
}
```

(segue la classe *Applet2Listener*)

### ESEMPIO 3

```
class Applet3Listener implements ActionListener {
    JTextArea area; JTextField riga;
    Applet3Listener(JTextArea ta, JTextField tf){
        area = ta; riga = tf;
    }
    public void actionPerformed(ActionEvent e){
        String azione = e.getActionCommand();
        String testoDaInserire;
        if (azione.equals("Elimina")) testoDaInserire = "";
        else testoDaInserire = riga.getText();
        int start = area.getSelectionStart();
        int end = area.getSelectionEnd();
        area.replaceRange(testoDaInserire,start,end);
        area.select(start,start);
    }
}
```

### ESEMPIO 3

La pagina HTML:

```
<HTML><HEAD>
<TITLE> Applet 2 </TITLE>
</HEAD> <BODY>
<APPLET CODE="Applet3.class"
    WIDTH=500 HEIGHT=300 >
</APPLET>
</BODY>
</HTML>
```



### APPLET E PARAMETRI

- Il file HTML può specificare parametri da passare all'applet, nella forma:

```
<APPLET CODE="Applet3.class"
    WIDTH=500 HEIGHT=300>
<PARAM NAME="ascissa" VALUE="123">
<PARAM NAME="ordinata" VALUE="67">
...
</APPLET>
```

- L'applet può recuperarli con il metodo `getParameter (nomeparametro)`, che restituisce una `String`

### ESEMPIO 4

L'applet dell'esempio 1 modificata:

```
import java.applet.*;
import java.awt.*;
import javax.swing.*;
public class Applet4 extends JApplet {
    Font f = new Font("Times", Font.BOLD, 36);
    public void paint(Graphics g) {
        g.setFont(f);
        g.setColor(Color.red);
        g.drawString(getParameter("Frase"), 100, 50);
    }
}
```

### ESEMPIO 4

La pagina HTML dell'esempio 1 modificata:

```
<HTML><HEAD>
<TITLE> Applet Parametrica </TITLE>
</HEAD> <BODY>
<APPLET CODE="Applet4.class"
    WIDTH=500 HEIGHT=100 >
    <PARAM NAME="Frase" VALUE="Oh, che bello!">
</APPLET>
</BODY>
</HTML>
```

Nome del parametro

Valore del parametro

## ESEMPIO 4

Risultato:



## APPLET: I METODI STANDARD

Un'applet è organizzata intorno a 4 metodi:

- `init()`, che viene chiamato dal browser quando lancia l'applet per la prima volta
  - fa le veci di un costruttore, legge i parametri, etc
- `start()`, che viene chiamato dal browser ogni volta che l'applet deve essere riavviata (perché torna visibile nella pagina)
  - tipicamente riavvia un'animazione o un thread
  - non occorre implementarlo se non ci sono animazioni o thread da riattivare

(segue)

## APPLET: I METODI STANDARD

(segue)

- `stop()`, che viene chiamato dal browser ogni volta che l'applet deve essere fermata (perché esce dall'area visibile della pagina)
  - tipicamente ferma un'animazione o un thread
  - non occorre implementarlo se non ci sono animazioni o thread da fermare
- `destroy()`, che viene chiamato dal browser quando il browser stesso si chiude
  - utile in casi particolari, per liberare i contesti grafici (di norma non occorre implementarlo)

## DA APPLICAZIONE A APPLET

Come convertire un'applicazione in un'applet?

- eliminare il main che crea il frame: non serve più, il frame è quello del browser
- sostituire `JFrame` con `JApplet` e assicurarsi che la classe sia **pubblica**, altrimenti l'applet non potrà essere caricata
- eliminare la chiamata a `setSize()`: ora le dimensioni del frame sono decise dalla pagina HTML tramite `HEIGHT` e `WIDTH`

(segue)

## DA APPLICAZIONE A APPLET

(segue)

- eliminare la chiamata a `addWindowListener()`: un'applet non può essere chiusa, termina quando l'utente esce dal browser
- eliminare la chiamata a `setTitle()`: un'applet non ha titolo, è la pagina HTML che lo definisce
- sostituire il costruttore col metodo `init()`: in realtà, un'applet può avere un costruttore, ma solo `init()` può recuperare i parametri tramite `getParameter()`.

## APPLET "DOUBLE FACE"

Un'applet può essere costruita in modo da poter funzionare anche come applicazione: basta aggiungere un main che svolga le funzioni normalmente svolte dal browser:

- creare il frame, dimensionarlo con `setSize()` e fissare il titolo con `setTitle()`
- impostare un `WindowListener` per gestire la chiusura della finestra (frame)

(segue)

## APPLET "DOUBLE FACE"

(segue)

- invocare il metodo `init()`
- avviare l'applet chiamando `start()`

**Nota:**

- non occorre chiamare il metodo `stop()`, perché un'applicazione termina quando il suo frame viene chiuso.

## ESEMPIO 5

Si vuole rendere "double face" l'Esempio 2.



Occorre creare un main che:

- crei un oggetto `Applet5`
- crei un `JFrame`, lo dimensioni, recuperi il `Container`, e gli aggiunga l'applet
- avvii l'applet con `init()`, e mostri il frame.

## ESEMPIO 5

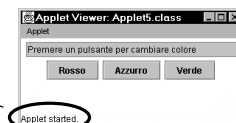
```
public class Applet5 extends JApplet {
    ...
    public static void main(String args[]) {
        Applet5 applet = new Applet5();
        JFrame f = new JFrame(
            applet.getClass().getName() );
        f.setSize(new Dimension(300,100));
        f.addWindowListener( new Terminator() );
        Container c = f.getContentPane();
        c.add(applet);
        applet.init(); // non c'è start();
        f.show();
    }
}
```

## ESEMPIO 5

Come applet:

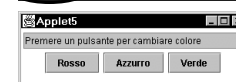
`appletviewer Applet5.html`

La finestra dell'applet  
si riconosce



Come applicazione:

`java Applet5`



## APPLET "DOUBLE FACE"

### UN APPROCCIO ALTERNATIVO

- Non toccare l'applet già fatta
- Ma sfruttare l'ereditarietà per definire una nuova classe che
  - erediti dall'applet che interessa
  - contenga il main opportuno

**Vantaggi:**

- è molto semplice
- non tocca neanche un file dell'applet originale
- si può usare sempre, per qualunque applet

## APPLET "DOUBLE FACE"

```
public class Application2 extends Applet2 {
    public static void main(String args[]) {
        Application2 applet = new Application2();
        JFrame f = new JFrame(
            applet.getClass().getName() );
        f.setSize(new Dimension(300,100));
        f.addWindowListener( new Terminator() );
        Container c = f.getContentPane();
        c.add(applet);
        applet.init(); // non c'è start();
        f.show();
    }
    // qui c'è anche la classe Terminator
}
```

## APPLET e SICUREZZA

Un'applet non può fare tutto quello che fa una applicazione.

- Poiché può essere scaricata dalla rete, sarebbe troppo pericoloso permettere a un'applet di fare qualunque cosa.
- Un'applet è costretta a rispettare un ben preciso modello di sicurezza ("sandbox")
  - è eseguita in una "scatola" da cui non può uscire
  - non può contaminare (o spiare) i dati del computer dell'utente

## APPLET e SICUREZZA

Un'applet di norma non può:

- accedere al file system locale (neppure per leggere un file)
- eseguire un altro programma
- ottenere informazioni sull'utente
- connettersi via rete a un computer *diverso da quello da cui è stata scaricata*
- caricare la libreria Java, chiamare `System.exit()`

Questi vincoli non si applicano all'appletviewer

## APPLET e SICUREZZA

Un'applet, inoltre:

- può aprire un'altra finestra, ma in essa compare automaticamente un avviso ("Warning: Applet window")

Problema:

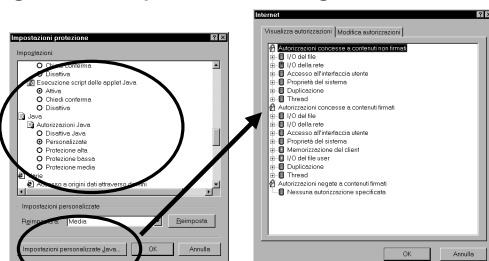
- in molte situazioni, questi vincoli sono *troppo rigidi*
- rischierebbero di rendere *impossibile* la costruzioni di applet *utili*.

## APPLET FIRMATE

- Attraverso tecnologie di cifratura, un'applet può essere firmata, ossia a essa può essere allegato un certificato che ne garantisce l'origine.
- Alle applet firmate, cui si attribuisce maggiore fiducia, *l'utente può consentire di svolgere alcune o tutte le operazioni sottoposte a vincolo*.

## APPLET FIRMATE

Ogni browser può essere configurato:



## POLITICHE DI SICUREZZA

- A partire da Java 2, l'utente può decidere *caso per caso* quali politiche di sicurezza applicare, con una *granularità molto fine*
- Esiste il concetto di *policy file*, che elenca le politiche locali
  - si può stabilire che una certa applet, proveniente da un ben preciso sito, ha diritti particolari
- Tale file può essere fornito da chi sviluppa l'applet, o modificato dall'utente con lo strumento *PolicyTool*.