Lezione 16 - Applet

Un'applet è un particolare tipo di programma Java il cui output è pensato per essere incluso in pagine Web. Quando un utente visualizza una pagina Web che contiene un'applet, l'applet viene eseguita **localmente** (cioè sulla macchina dell'utente) e **non** **remotamente** (cioè sul server come le servlet). L'applet, come tutte le applicazioni Java, gira su una Virtual Machine su un processo separato da quello del browser.

Dato che il codice dell'applet viene fornito dal server ma gira sulla macchina del client l'applet stessa è soggetta a delle limitazioni per renderne sicura l'esecuzione stessa.
Le principali limitazioni imposte alle applet sono:

- Non si può leggere dal disco locale;
- Non si può scrivere nel disco locale;
- Non si possono aprire connessioni con server diversi dal server da cui proviene l'applet;
- Non si possono invocare programmi locali;
- Non si possono scoprire informazioni privare riguardo all'utente.

Per ovviare a queste limitazioni è possibile firmare le applet. In questo caso è possibile associare un file policy file che descrive tutte le operazioni che possono essere compiute dall'applet (Es. leggere un certo file sul disco locale).

HTML

Il browser dev'essere istruito su come visualizzare l'applet stessa. Questo lo otteniamo attraverso il tag HTML "APPLET" che associa il file .class alla pagina Web. L'attributo CODE indica con un URL relativo il file .class da caricare. I parametri WIDTH e HEIGHT indicano rispettivamente la larghezza e l'altezza dello spazio da riservare all'applet nella pagina Web.
Ecco un esempio di codice HTML per la visualizzazione di un'applet:

```
<html>
<head>
  <title>AppletReload</title>
</head>
<body>
<applet code="ReloadApplet.class" width="120" height="60">
<b>Attenzione: Devi abilitare le Applet!</b></applet>
</body>
</html>
```

java 2 (jdk 1.2)

Il tag Applet è usabile solo se la nostra applet utilizza la versione del linguaggio 1.
In particolare nella prima versione non sono disponibili:

1. Swing (a meno di non scaricare le classi che la contengono);
2. Java 2D;
3. collezioni (List, Map etc.);
4. ottimizzazioni per l'esecuzione più veloce.

Per ottenere tutti i vantaggi di Java 2 dobbiamo operare in maniera diversa e usare i seguenti tag:

- Internet Explorer: OBJECT;

- Netscaep: EMBED.

In particolare dovremmo procedere nel seguente modo:

```
<OBJECT classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
WIDTH = "600" HEIGHT = "300" NAME = "ReloadApplet"
codebase="http://java.sun.com/products/plugin/1.1.1/jinstall-111-
win32.cab#Version=1,1,1,0">
<PARAM NAME = CODE VALUE = "ReloadApplet.class" >
<PARAM NAME = NAME VALUE = "ReloadApplet" >

<PARAM NAME="type" VALUE="application/x-java-applet;version=1.1">
<COMMENT>
<EMBED type="application/x-java-applet;version=1.1" java_CODE =
"ReloadApplet.class" NAME = "ReloadApplet" WIDTH = "600" HEIGHT =
"300"   pluginspage="http://java.sun.com/products/plugin/1.1.1/plugin-
install.html">
<NOEMBED></COMMENT>

</NOEMBED></EMBED>
</OBJECT>
```

java 2 e JSP

La gestione dei tag per le applet viene fatto in automatico dal motore delle servlet nel seguente modo:

```
<jsp:plugin type="applet"
            code="ReloadApplet.class"
            width="475" height="350">
</jsp:plugin>
```

che produce in automatico i tag HTML necessari.

Java

```
import java.applet.Applet;
import java.awt.Graphics;
public class Ciao extends Applet {
   public void paint(Graphics g) {
       g.drawString("Ciao", 50, 25);
       }
}
```

La classe Applet è sottoclasse di Panel, nella libreria Awt. Questo implica in particolare che:
Applet ha come layout manager di default il Flow Layout Manager, che dispone gli elementi grafici da sinistra a destra, con allineamento centrale
Applet eredita le variabili ed i metodi delle sue superclassi Component, Container e Panel.
Con un applet è possibile visualizzare all'interno della finestra della pagina Html tutto quello che è possibile fare con una normale applicazione Java senza le limitazione dell'html (vedi sotto esempio dell'applet con testo scorrevole).

ciclo vita di una Applet

Anche l'applet analogamente alla servlet ha un ciclo di vita è una serie di metodi che vengono invocati automaticamente dalla Virtual Machine dove gira l'applet. I metodi invocati sono:

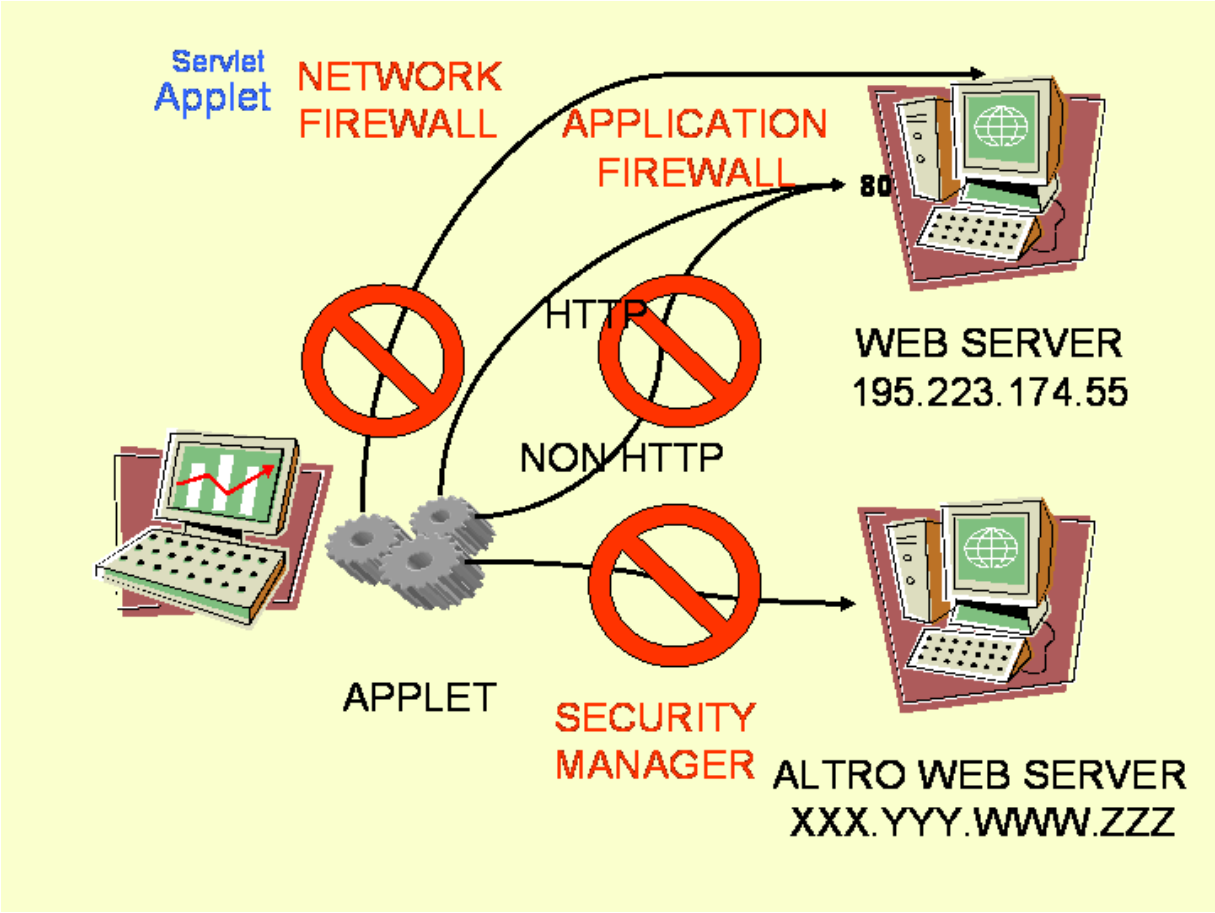
- `init()`: quando si apre un documento con un applet, viene invocato automaticamente il metodo `init()` per l'inizializzazione dell'applet. Può essere usato per inizializzare le variabili. Questo

metodo è chiamato solo una volta per applet;

- `start()`: quando un documento con un applet viene aperto, viene chiamato il metodo `start()` automaticamente dopo `init()`, per iniziare l'applet. Questo metodo, ad esempio, è invocato ogni qual volta l'applet viene "rivisitato".
- `stop()`: viene invocato automaticamente quando l'utente si sposta fuori dalla pagina su cui è situato l'applet . Serve per bloccare attività che possono rallentare il sistema quando l'utente non sta utilizzandole (es. animazioni) . I metodi start e stop formano una coppia: start attiva un comportamento, stop lo disattiva.
- `destroy()` : Viene invocato quando l'applet viene dismesso (ad es. quando viene chiuso il browser), e in questo caso viene prima invocato il metodo stop().
  Tutte le risorse legate all'applet vengono rilasciate.
- `paint()` : Questo metodo è invocato direttamente dal browser dopo che `init()` è stato completamente eseguito e dopo che `start()` abbia iniziato l'esecuzione. E' invocato ogni qual volta l'applet deve essere ridipinto
- `repaint()` : invoca il metodo `update()` al più presto (non da overridden) `update()` permette di "ri-dipingere" l'applet, e può essere riscritto. Per default, "pulisce" lo sfondo e chiama il metodo `paint()`.

Comunicazione Servlet/Applet

Uno dei problemi che possono essere risolti da un'applet è quello di permettera una più immediata e tempestiva comunicazione tra server e browser. La limitazione del protocollo http non permette di avere un meccanismo semplice con cui il server in maniera "attiva" avvisi il client che nuove informazioni sono disponibili o più in generale di comunicare col client stesso. Vediamo ora una soluzione applet/servlet per ottenera una comunicazione in cui il server avvisa il client al verificarsi di certi eventi.



La Sandbox presente nel browser impedisce all'applet di aprire connessione con server diversi da quello dalla quale l'applet stessa è stata scaricata. La presenza di Firewall potrebbe impedire di aprire una connessione TCP su porte diverse dalla porta 80 (porta che deve per forza essere aperta in quanto porta standard del protocollo HTTP). Quindi l'unica porta sicuramente aperta è la porta 80 dove è in attesa di connessioni il server HTTP. Il fatto che ci sia il server HTTP in attesa di connessioni HTTP e dal fatto che ci potrebbe essere un firewall a livello applicativo che impedisci connessioni diverse da HTTP suggerisce di usare come schema iniziale di comunicazione tra Applet e Servlet il protocollo HTTP stesso.

Il fatto di usare come schema iniziale di comunicazione tra Applet e Server quello standard HTTP ha anche il vantaggio che è possibile usare la stessa Servlet sia per comunicare interattivamente con un Applet che per fornire una risposta statica generando una pagina HTML.
La servlet rimane “simile” a quelle viste.
Azioni da compiere nell'Applet per invocare una servlet.

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;
public class PushApplet extends Applet implements ActionListener
{
  TextArea taResults;
  Button btnStart;
  public void init()
  {
    setLayout(new FlowLayout(FlowLayout.LEFT));
    btnStart = new Button("Start");
    btnStart.addActionListener(this);
    add("North", btnStart);
    taResults = new TextArea(10, 80);
    add("Center", taResults);
  }
  public void execute()
  {
    char[] buf = new char[256];
    try
    {
      URL url = new URL(getDocumentBase(),"PushServlet");
      URLConnection uc = url.openConnection();
      uc.setDoInput(true);
      uc.setUseCaches(false);

      InputStreamReader in = new InputStreamReader(uc.getInputStream());
      int len = in.read(buf);
      while (len != -1 )
      {
        taResults.replaceRange(new String(buf,0,len),0,len);
        len = in.read(buf);
      }
      in.close();
    }
    catch(MalformedURLException e)
    {
      taResults.setText(e.toString());
    }
    catch(IOException e)
    {
      taResults.setText(e.toString());
    }
  }
  public void actionPerformed(ActionEvent ae)
  {
    execute();
  }
```

```
}
```

Nota: l'applet rimane bloccata in attesa di dati da parte del server. Andrebbe gestita la richiesta e l'attesa dei dati in un thred secondario (vedi esempio testo scorrevole).

Possiamo utilizzare un'applet per ricaricare tutta la pagina che contiene l'applet utilizzando la seguente istruzioni:

```
getAppletContext().showDocument(new
URL(getDocumentBase(),"AppletServletCommunication.html"));
```

Testo scorrevole

Un'esempio tipico di funzionalità ottenibili con le applet è il testo scorrevole:

```
public class Scroll extends Applet implements Runnable
{
Thread t=null;
String temp;
String text = "Questo testo scorre";
long velocitaBattitura=100;

boolean cont = true;
public void start()
{
t = new Thread(this);
cont = true;
t.start();
}
public void stop()
{
cont = false;
t.interrupt();
t=null;
}
public void run()
{
while (cont)
{
for(int j=0; j<=text.length(); j++)
{
temp = text.substring(j)+ " " + text.substring(0,j);
repaint();
try{t.sleep(velocitaBattitura);}
catch(InterruptedException eint){if (!cont) break;}
}
}
}
public void paint(Graphics g)
{
Font f = new Font("Arial",0,16);
FontMetrics fm = g.getFontMetrics(f);
int w = fm.stringWidth(temp);
g.setColor(Color.white);
g.fillRect(0,0,w+10,30);
g.setColor(Color.black);
g.setFont(f);
```

```
g.drawString(temp, 5, 20);
}

}
```

Sintassi del tag APPLET

```
<applet
[archive=ListaArchivio] // classi o risorse che saranno preloaded. Si
puo' indicare una lista di file JAR (Java Archive) dalla quale
estrarre l'applet
code=MioFile.class // nome del file che contiene il compilato,
relativo alla stessa URL
width=pixels heigh=pixels // dimensioni iniziali del display
dell'applet
[codebase=codebaseURL] // directory che contiene il codice dell'applet

[alt=testoAlternativo] // se il browser legge l'applet tag ma non può
eseguire l'applet
[name=nomeIstanzaApplet] // permette di identificare diversi applet
nella stessa pagina per esempio, permette ad applet di comunicare tra
loro…
[align=allineamento]// left right top texttop middle baseline bottom
[vspace=pixels] [hspace=pixels]
>
[<param name=attributo1 value=valore>]// valori specificati
dall'esterno
[<param name=attributo2 value=valore>]// ci sarà un getParameter()
nell'applet
. . .
</applet>
```

L'uso dei parametri permette in modo molto semplice di personalizzare il comportamente dell'applet.
Esempio:

```
public void init(){
String param = getParameter("testo");
...
}
```

Nel file HTML possiamo usare l'applet con due parametri diversi:

```
<html>
<title>Etichetta</title>
<body>
<applet code = "Etichetta.class"
width=500 height=100
name=primo
align = top
vspace=30
>
<param name = testo value ="testo primo">
</applet>

<p><hr><p>

<applet code = "Etichetta.class"
width=500 height=100
name=secondo
align = right
```

```
vspace=30
>
<param name = testo value ="testo diverso">

</applet>
</body>
</html>
```