Java

creare applets

G. Prencipe
prencipe@di.unipi.it

Applets

- Java offre la capacità di creare *applets*
- Sono piccoli programmi che possono essere eseguiti all'interno di un browser Web
- Dato che questi programmi *devono* essere *sicuri*, le applets sono limitate in quello che possono fare

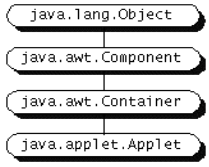
Restrizioni delle applets

- La programmazione con le applets è talmente restrittiva che spesso viene riferita come programmazione "inside the sandbox"
  - Area recintata con della sabbia dove spesso giocano i bambini
- Infatti, c'è sempre qualcuno (il sistema di sicurezza a run-time di Java) che controlla quello che viene fatto
  - Comunque, è sempre possibile uscire dalla *sandbox* e scrivere normali applicazioni che possono accedere tutte le caratteristiche offerte dal sistema operativo

Restrizioni delle applets

- Lo scopo delle applet è quello di estendere le funzionalità di una pagina Web in un browser
- È per questo che alle applet vengono imposte alcune restrizioni
  - Una applet non può accedere al disco locale
    - Non in lettura, non in scrittura, dato che non vogliamo che una applet trasmetta dati locali su Internet senza permesso
  - Le applet possono impiegare maggiore tempo per essere visualizzate
    - Bisogna scaricare tutta la applet ogni volta (il browser può usare caching, ma non è garantito)
    - È quindi conveniente impacchettare tutte le componenti di una applet in un archivio JAR

Gerarchia grafica



JFrame, **JApplet**, **JDialog**, **JWindow** sono le componenti *Swing heavyweight*

Sono le componenti che sono *in cima* a qualsiasi gerarchia Swing (sono detti *top-level container*)

Sono utilizzati per contenere le componenti *lightweight* (bottoni, testi, ecc.)

Le Applet Java

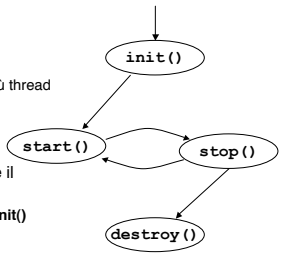
- Per le applicazioni Java, il primo metodo eseguito è il **`main()`** di una classe qualunque
- Per le applet, il ciclo di vita è più complicato:
  - l'applet deve essere sottoclasse di **`Applet`** (o di **`JApplet`**, che è una sua sottoclasse)
  - al caricamento, viene chiamato **`init()`**, poi **`start()`**
  - se l'utente cambia pagina e poi vi ritorna, vengono chiamati **`stop()`** e **`start()`** – anche più volte
  - alla fine, viene chiamato **`stop()`**, poi **`destroy()`**

Le applets

- Le applets si costruiscono ereditando dalla classe **Applet** o **JApplet** e riscrivendo i metodi appropriati
- Ci sono alcuni metodi che controllano la creazione e l'esecuzione di una applets in una pagina Web
  - **init()**: invocata automaticamente la prima volta che la applet viene caricata per effettuare l'inizializzazione della applet
  - **start()**: invocata ogni volta che viene lanciata la applet
  - **stop()**: invocata per bloccare la applet. Invocata anche subito prima di **destroy()**
  - **destroy()**: invocata quando la applet viene scaricata dalla pagina. Avviene il rilascio delle risorse utilizzate dalla applet

Ciclo di vita delle Applet

- **init()** per le inizializzazioni
  - serve come un costruttore
- **start()** per avviare i lavori
  - direttamente o creando uno o più thread
- **stop()** per sospendere o chiudere
  - si fermano i thread
- **destroy()** quando l'utente chiude il browser
  - si liberano le risorse allocate in **init()**



Esempio

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

public class Applet1 extends JApplet {
  public void init() {
    getContentPane().add(new JLabel("Applet!"));
  }
} ///:~
```

- Lo scopo è di inserire una etichetta di testo su una applet, utilizzando la classe **JLabel**
  - Nelle vecchie AWT il nome era **Label**. La **J** compare dinanzi a molti componenti delle *Swing*
  - Il costruttore prende una Stringa e crea l'etichetta

Esempio

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

public class Applet1 extends JApplet {
  public void init() {
    getContentPane().add(new JLabel("Applet!"));
  }
} ///:~
```

- Il metodo **Component add(Component comp)** è nella classe **java.awt.Container**
  - **JLabel** è un **Component**
  - **add()** aggiunge una componente a un contenitore, e restituisce la stessa componente **comp**

Esempio

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

public class Applet1 extends JApplet {
  public void init() {
    getContentPane().add(new JLabel("Applet!"));
  }
} ///:~
```

- Il metodo **getContentPane()** è in **JApplet** e restituisce un Container (che potrà contenere tutte le componenti che aggiungiamo)
- Proviamo a scrivere questo pezzo di codice in Eclipse ed eseguiamolo come Applet

Eseguire applets in Eclipse

- È sufficiente selezionare *Applet* dal menù di *Run As*

Eseguire applets in un browser

- Per eseguire questa applet da un Web browser, bisogna inserirla in una pagina Web utilizzando alcuni tag speciali nel codice HTML che specificano come caricare ed eseguire la applet

  <applet code=Applet1 width=100 height=50>

  </applet>
  - Chiaramente il file **Applet1.class** deve essere nel CLASSPATH altrimenti il browser non lo trova e non esegue la applet
  - Provare a creare la pagina html e invocare la applet
- Scaricando il plugin di Eclipse *SDK for Web Standard Tools (WST*) è possibile utilizzare anche un editor HTML

Utilizzare l'editor HTML

- Dall'interno di un progetto, selezionare New, Other, Web, HTML
- Viene creata una pagina HTML vuota all'interno di un editor HTML
- Per caricare questa pagina, attivare il menù di contesto (pulsante destro) del file HTML e selezionare *Open With Web Browser*
  - Viene aperto il browser integrato di Eclipse

Visualizzatore di Applets

- È possibile visualizzare le applet anche tramite il visualizzatore di applet contenuto nel SDK della Sun
  - **appletviewer**
- Questo tool considera *solo* i tag APPLET e li esegue, ignorando il resto del codice HTML
  - In questo modo è sufficiente inserire questi tag direttamente nel codice .java come commenti e invocare **appletviewer Applet1.java**
  - In questo modo, l'**appletviewer** eseguirà solo i tag APPLET (ignorando il resto del codice Java)

Esempio

```
// <applet code=Applet1 width=100 height=50></applet>
import javax.swing.*;
import java.awt.*;

public class Applet1 extends JApplet {
  public void init() {
    getContentPane().add(new JLabel("Applet!"));
  }
} ///:~
```

- Eseguire (dalla shell)
  appletviewer Applet1.java

Applet da linea di comando

- È possibile anche creare una classe che possa essere invocata sia come applicazione “standard” che come applet
- Per fare questo, è sufficiente aggiungere il **main()** al normale codice scritto per una applet
  - In questo caso il **main** deve provvedere a inizializzare e lanciare la applet

Applet da linea di comando

- Chiaramente in questo modo non otteniamo lo stesso comportamento che avremmo lanciando la applet da un browser
  - In questo caso infatti si invocano anche **stop()** e **destroy()**

Esempio

```
// <applet code=Applet1c width=100 height=50></applet>
import javax.swing.*; import java.awt.*;
public class Applet1c extends JApplet {
  public void init() {
    getContentPane().add(new JLabel("Applet!"));
  }
  // Il main() per l'aaplicazione
  public static void main(String[] args) {
    JApplet applet = new Applet1c();
    JFrame frame = new JFrame("Applet1c");
// Per chiudere l'applicazione
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.getContentPane().add(applet);
    frame.setSize(100,50);
    applet.init();
    applet.start();
    frame.setVisible(true);
  }} ///:~
```

Utilizzare i file JAR

- Un importante utilizzo dei file JAR è per ottimizzare il caricamento delle applet
- In Java 1.0 la tendenza era di inserire tutto il codice riguardante una applet in una sola classe
  - In questo modo l'utente caricava l'intera applet con un solo click
- Il caricamento risultava comunque lento, dato che bisognava caricare tutta la applet

Utilizzare i file JAR

- I file JAR risolvono il problema comprimendo tutti i file **.class** in un unico file che viene caricato dal browser
- In questo modo è possibile mantenere la progettazione delle classi senza preoccuparsi del numero di classi create
- Inoltre, conseguenza della compressione è un minor tempo di scaricamento della applet

Esercizio

- Scrivere una applet contenente un **JPasswordField**
- Aggiungere un **JOptionPane** che restituisce un messaggio all'utente
  - *OK* se la password inserita è corretta
  - *Riprova* altrimenti
- Provare a eseguire il programma da Eclipse

Esercizio -- continua

- Comprimere il codice in un file **Password.jar**
  - Utilizzare la funzione di *Export* di Eclipse
- Creare un file html da cui si invoca **Password.jar**
  - La struttura html per invocare la applet è

```
<applet code=Password.class
        archive=Password.jar
        width=100 height=50> </applet>
```

Applet firmate

- In genere, per motivi di sicurezza, le applet possono fare molto poco
  - Ad esempio, non possono accedere al file system
- Per ovviare a questo problema, bisogna firmare le applet (*signed applet*)
  - Con una applet firmata è possibile verificare che la persona che ha creato la applet lo ha fatto davvero, e che il contenuto del file jar non è stato modificato da quando ha lasciato il server

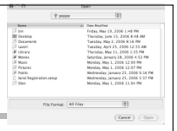
Applet firmate

- Il processo della firma di una applet è stato notevolmente semplificato in seguito al rilascio del plugin Java
- In precedenza era necessario firmare un file **.jar** con un tool di Netscape (per i clienti Netscape) o un file **.cab** con un tool Microsoft (per i clienti Explorer), e creare un tag html specifico per la piattaforma su cui doveva essere eseguita la applet

Applet firmate

- Il plugin fornisce un approccio standard al processo di firma delle applet
- Inoltre permette di automatizzare il processo

Esercizio



- Supponiamo di voler creare una applet che accede al file system del cliente, e che apre un **JFileChooser**
  - **FileAccessApplet.java**
  - Creare due bottoni (**Open** e **Save**), e alla pressione di si attiva il **JFileChooser**, e si sceglie un file
  - La scelta del file provoca la scrittura del nome del file su una area di testo
- Se la applet non è firmata, il metodo **showOpenDialog()** che apre la finestra di dialogo per navigare il file system genera una **SecurityException**

Esempio

- Dopo aver scritto la nostra applet (che appare come una normalissima applet), bisogna firmarla
- Per fare questo, bisogna
  - Mettere il codice in un file **.jar**
    - **fileaccess.jar**
  - Firmare il file **.jar**

Esempio -- firmare il JAR

- Per creare un certificato o una chiave con cui firmare il file **.jar**, bisognerebbe registrarsi presso un'autorità competente (tipo Verisign o Thawte), che rilascerebbero un certificato
  - Va pagato!!
- È possibile comunque creare un certificato per scopi di testing utilizzando **keytool** distribuito con Java

Esempio -- firmare il JAR

- Il comando per generare la firma è

  keytool -genkey -alias <keyname> -keystore <url>
- **keyname** è l'alias che vogliamo dare alla chiave, ad es. **miaChiave**
- **url** è la locazione del file che memorizza le chiavi
  - È tipicamente in un file chiamato **cacerts** in **jre/lib/security**

Esempio -- firmare il JAR

- Ecco il risultato del dialogo per la generazione della chiave

```
keytool -genkey -alias fileaccess -keystore ChiaviEsempioFileAccessApplet
Enter keystore password:  pippo
Keystore password is too short - must be at least 6 characters
Enter keystore password:  pippolo
What is your first and last name?
  [Unknown]:  Giuseppe Prencipe
What is the name of your organizational unit?
  [Unknown]:  Dipartimento Informatica
What is the name of your organization?
  [Unknown]:  Universita Pisa
What is the name of your City or Locality?
  [Unknown]:  Pisa
```

Esempio -- firmare il JAR

```
What is the name of your State or Province?
  [Unknown]:  Pisa
What is the two-letter country code for this unit?
  [Unknown]:  IT
Is CN=Giuseppe Prencipe, OU=Dipartimento Informatica, O=Universita
    Pisa, L=Pisa, ST=Pisa, C=IT correct?
  [no]:  yes
Enter key password for <fileaccess>
        (RETURN if same as keystore password):
giuseppe-prencipes-ibook-g4:~/Documents/EclipseWorkspace/Applets
    peppe$
```

Esempio -- firmare il JAR

- Per firmare il file **.jar**, eseguire
  jarsigner -keystore <url> <jarfile><keyname>
- Ecco il risultato:

jarsigner -keystore ChiaviEsempioFileAccessApplet fileaccess.jar fileaccess
Enter Passphrase for keystore: pippolo

Warning: The signer certificate will expire within six months.

- Ora la nostra applet è firmata!!

JNLP

- Le applet firmate sono un tool efficace, ma necessitano di un web browser per essere eseguite
- Il *Java Network Launch Protocol* (*JNLP*) risolve questo problema, senza sacrificare i vantaggi delle applet
- Con una applicazione JNLP è possibile scaricare una applicazione Java *standalone* sulla macchina del cliente
  - Può essere eseguita da shell, da una icona sul desktop, ecc.

JNLP

- Anche le applicazioni JNLP sono soggette alle restrizioni di sicurezza imposte dalla sandbox
- Come le applet, possono essere racchiuse in un file JAR, fornendo all'utente la possibilità di verificare il firmatario
- Al contrario delle applet, anche se sono in unn JAR non firmato, possono richiedere l'utilizzo di alcune risorse al sistema cliente (che l'utente deve comunque autorizzare durante l'esecuzione)

JNLP e Java Web Start

- JNLP descrive un protocollo e non un'implementazione
- L'implementazione di riferimento per JNLP è Java Web Start della Sun
  - Gratuita

JNLP

- Creare una applicazione JNLP non è difficile
- Si crea un'applicazione standard, che si archivia in un JAR
- Poi si fornisce un file d'esecuzione (*launch file*), che è un semplice file XML che fornisce al cliente tutte le informazioni necessarie per scaricare e installare l'applicazione

Java

creare applets

fine