

Java applets

Sergio Luján Mora



Departamento de Lenguajes y
Sistemas Informáticos



Universitat d'Alacant
Universidad de Alicante

Contents

- What is an applet?
- The life cycle of an applet
- Painting
- Handling events
- Applet API
- Security restrictions

What is an applet?

- An applet is a special kind of Java program that a browser enabled with Java technology can download from the internet and run
- An applet is typically embedded inside a web-page and runs in the context of the browser
- An applet must be a subclass of the `java.applet.Applet` class, which provides the standard interface between the applet and the browser environment.

The life cycle of an applet

- By calling certain methods, a browser manages an applet life cycle, if an applet is loaded in a web page:
 - `init`: This method is intended for whatever initialization is needed for your applet. It is called after the param attributes of the applet tag.
 - `start`: This method is automatically called after `init` method. It is also called whenever user returns to the page containing the applet after visiting other pages.
 - `stop`: This method is automatically called whenever the user moves away from the page containing applets. You can use this method to stop an animation.
 - `destroy`: This method is only called when the browser shuts down normally.

The life cycle of an applet

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

public class LifeCycle extends Applet {

    StringBuffer buffer;

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

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

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

The life cycle of an applet

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

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

public void paint(Graphics g) {
    //Draw a Rectangle around the applet's display area.
    g.drawRect(0, 0,
        getWidth() - 1,
        getHeight() - 1);

    //Draw the current string inside the rectangle.
    g.drawString(buffer.toString(), 5, 15);
}
}
```

The life cycle of an applet

```
<!-- lifecycle.html -->
<html>
<head>
<title>The Life Cycle of an Applet
</title>
</head>
<body>
<applet code="LifeCycle.class" width="500"
  height="20">
</applet>
</body>
</html>
```

The life cycle of an applet

- For development and testing purposes, you can run your applet using the lightweight `appletviewer` application that comes with the JDK:

```
appletviewer lifecycle.html
```

Painting

- To draw the applet's representation within a browser page, you use the `paint` method

```
public void paint(Graphics g) {  
    //Draw a Rectangle around the applet's display area.  
    g.drawRect(0, 0,  
        getWidth() - 1,  
        getHeight() - 1);  
  
    //Draw the current string inside the rectangle.  
    g.drawString(buffer.toString(), 5, 15);  
}
```

Handling events

- To react to an event, an applet must override the appropriate event-specific method
- **Interfaces:** `KeyListener`, `MouseListener`, `TextListener`, `WindowListener`, ...
- **Events:** `KeyEvent`, `MouseEvent`, `TextEvent`, `WindowEvent`, ...

Handling events

```
import java.awt.event.MouseListener;
import java.awt.event.MouseEvent;

public class MouseEvents extends Applet
    implements MouseListener {

    public void mouseEntered(MouseEvent event) {
    }
    public void mouseExited(MouseEvent event) {
    }
    public void mousePressed(MouseEvent event) {
    }
    public void mouseReleased(MouseEvent event) {
    }
    public void mouseClicked(MouseEvent event) {
    }
}
```

```
import java.awt.event.MouseListener;
import java.awt.event.MouseEvent;
import java.applet.Applet;
import java.awt.Graphics;

public class MouseEvents extends Applet
    implements MouseListener {

    StringBuffer buffer;

    public void init() {
        addMouseListener(this);
        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) {
    //Draw a Rectangle around the applet's display area.
    g.drawRect(0, 0,
        getWidth() - 1,
        getHeight() - 1);

    //Draw the current string inside the rectangle.
    g.drawString(buffer.toString(), 5, 15);
}

public void mouseEntered(MouseEvent event) {
    addItem("enter!... ");
}
public void mouseExited(MouseEvent event) {
    addItem("exit!... ");
}
public void mousePressed(MouseEvent event) {
    addItem("press!... ");
}
public void mouseReleased(MouseEvent event) {
    addItem("release!... ");
}

public void mouseClicked(MouseEvent event) {
    addItem("click!... ");
}
}

```

Applet API

- `getCodeBase`:
 - Gets the base URL. This is the URL of the directory which contains this applet
- `getDocumentBase`:
 - Gets the URL of the document in which this applet is embedded
- `getLocale`:
 - Gets the locale of the applet
 - It allows the applet to maintain its own locale separated from the locale of the browser or appletviewer

Applet API

- Implement:

- `getAppletInfo`:

- Returns information about this applet
 - An applet should override this method to return a `String` containing information about the author, version, and copyright of the applet
 - The implementation of this method provided by the `Applet` class returns `null`

Applet API

- Implement:

- `getParameterInfo`:

- Returns information about the parameters that are understood by this applet
 - An applet should override this method to return an array of `Strings` describing these parameters
 - Each element of the array should be a set of three strings containing the name, the type, and a description. For example:

```
String pinfo[][] = { {"fps", "1-10", "frames per second"},  
                    {"repeat", "boolean", "repeat image loop"},  
                    {"imgs", "url", "images directory"} };
```

- The implementation of this method provided by the `Applet` class returns `null`

```

import java.applet.Applet;
public class AppletAPI extends Applet {
    public void init() {
    }
    public void start() {
        System.out.println("getCodeBase(): " + getCodeBase());
        System.out.println("getDocumentBase(): " + getDocumentBase());
        System.out.println("getLocale().getCountry(): " + getLocale().getCountry());
        System.out.println("getLocale().getDisplayCountry(): " + getLocale().getDisplayCountry());
        System.out.println("getLocale().getLanguage(): " + getLocale().getLanguage());
        System.out.println("getLocale().getDisplayLanguage(): " +
getLocale().getDisplayLanguage());
        System.out.println("getLocale().getDisplayName(): " + getLocale().getDisplayName());
        System.out.println("source: " + getParameter("source"));
        System.out.println("target: " + getParameter("target"));
    }
    public void stop() { }
    public void destroy() { }
    public String getAppletInfo()
    { return "Calling getAppletInfo()"; }
    public String[][] getParameterInfo() {
        String[][] info = {
            // Parameter Name Kind of Value Description
            {"source", "URL", "the source directory"},
            {"target", "URL", "the target directory"},
        };
        return info; }
    }
}

```

Applet API

```

getCodeBase():
file:/D:/Profe/Cursos/2007/Argelia/Applet/AppletAPI/
getCodeBase():
file:/D:/Profe/Cursos/2007/Argelia/Applet/AppletAPI/a
ppletapi.html
getLocale().getCountry(): ES
getLocale().getDisplayCountry(): España
getLocale().getLanguage(): es
getLocale().getDisplayLanguage(): español
getLocale().getDisplayName(): español (España)

```

Security restrictions

- Existing applet viewers (including Web browsers) impose the following restrictions:
 - **Applets cannot load libraries or define native methods.**
 - Applets can use only their own Java code and the Java API the applet viewer provides. At a minimum, each applet viewer must provide access to the API defined in the java.* packages.
 - **An applet cannot ordinarily read or write files on the host that is executing it.**
 - The JDK Applet Viewer actually permits some user-specified exceptions to this rule, but older browsers generally do not. Applets in any applet viewer *can* read files specified with full URLs, instead of by a filename. A workaround for not being able to write files is to have the applet forward data to an application on the host the applet came from. This application can write the data files on its own host
 - **An applet cannot make network connections except to the host that it came from.**
 - The workaround for this restriction is to have the applet work with an application on the host it came from. The application can make its own connections anywhere on the network.

Security restrictions

- Existing applet viewers (including Web browsers) impose the following restrictions:
 - **An applet cannot start any program on the host that is executing it.**
 - Again, an applet can work with a server-side application instead.
 - **An applet cannot read certain system properties.**
 - **Windows that an applet brings up look different than windows that an application brings up.**
 - You can identify the Applet window by the name 'Java Applet Window', which is displayed at the bottom of the window. Application window would not have any name at its bottom. This helps the user distinguish applet windows from those of trusted applications.