Java Applet

Applet is a special type of program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side.

Advantage of Applet

There are many advantages of applet. They are as follows:

* It works at client side so less response time.
* Secured
* It can be executed by browsers running under many plateforms, including Linux, Windows, Mac Os etc.

Drawback of Applet

* Plugin is required at client browser to execute applet.

There are some important differences between an applet and a standalone Java application, including the following −

* An applet is a Java class that extends the java.applet.Applet class.
* A main() method is not invoked on an applet, and an applet class will not define main().
* Applets are designed to be embedded within an HTML page.
* When a user views an HTML page that contains an applet, the code for the applet is downloaded to the user's machine.
* A JVM is required to view an applet. The JVM can be either a plug-in of the Web browser or a separate runtime environment.
* The JVM on the user's machine creates an instance of the applet class and invokes various methods during the applet's lifetime.
* Applets have strict security rules that are enforced by the Web browser. The security of an applet is often referred to as sandbox security, comparing the applet to a child playing in a sandbox with various rules that must be followed.

Hierarchy of Applet

|  |
| --- |
| As displayed in the above diagram, Applet class extends Panel. Panel class extends Container which is the subclass of Component. |

Lifecycle of Java Applet

1. Applet is initialized.
2. Applet is started.
3. Applet is painted.
4. Applet is stopped.
5. Applet is destroyed.

Lifecycle methods for Applet:

The java.applet.Applet class 4 life cycle methods and java.awt.Component class provides 1 life cycle methods for an applet.

java.applet.Applet class

For creating any applet java.applet.Applet class must be inherited. It provides 4 life cycle methods of applet.

1. **public void init():** is used to initialized the Applet. It is invoked only once.
2. **public void start():** is invoked after the init() method or browser is maximized. It is used to start the Applet.
3. **public void stop():** is used to stop the Applet. It is invoked when Applet is stop or browser is minimized.
4. **public void destroy():** is used to destroy the Applet. It is invoked only once.

java.awt.Component class

The Component class provides 1 life cycle method of applet.

1. **public void paint(Graphics g):** is used to paint the Applet. It provides Graphics class object that can be used for drawing oval, rectangle, arc etc.

Who is responsible to manage the life cycle of an applet?

Java Plug-in software.

How to run an Applet?

There are two ways to run an applet

1. By html file.
2. By appletViewer tool (for testing purpose).

Applet class

Development and execution of a simple applet

1.writing the applet’s code(.java file)

2. compiling the applet code(.class file will be ready after the compilation)

3. Writing the html code(.html file)

4. Invoking teh applet viewer utility from a command shell window

(appletviewer helloapplet.html)

Step 1:

Writing the code in .java (hellioapplet.java)

|  |
| --- |
| import java.awt.\*;  import java.applet.\*;  public class helloapplet extends Applet  {  public void paint(Graphics g)  {  g.drawString("hello welcome",65,35);  }  } |

Step 2 : compile the .java file

Javac helloapplet.java

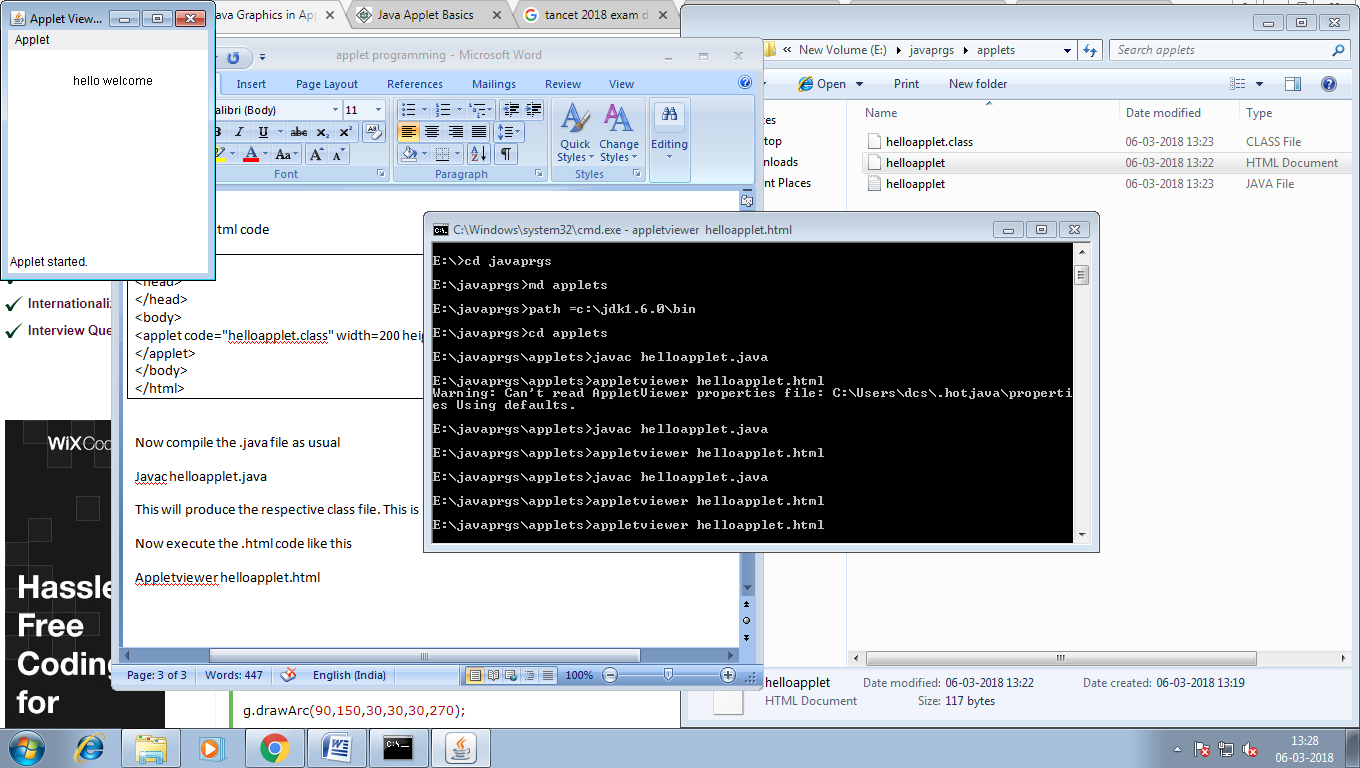
This will create helloapplet.class file. This is the file you mention in the applet tag of html file

Step 4:Writing the .html code

|  |
| --- |
| <html>  <head>  </head>  <body>  <applet code="helloapplet.class" width=200 height =200>  </applet>  </body>  </html> |

Now execute the .html code like this

Appletviewer helloapplet.html



The output will be like above

# Displaying Graphics in Applet

java.awt.Graphics class provides many methods for graphics programming.

## **Commonly used methods of Graphics class:**

1. **public abstract void drawString(String str, int x, int y):** is used to draw the specified string.
2. **public void drawRect(int x, int y, int width, int height):** draws a rectangle with the specified width and height.
3. **public abstract void fillRect(int x, int y, int width, int height):** is used to fill rectangle with the default color and specified width and height.
4. **public abstract void drawOval(int x, int y, int width, int height):** is used to draw oval with the specified width and height.
5. **public abstract void fillOval(int x, int y, int width, int height):** is used to fill oval with the default color and specified width and height.
6. **public abstract void drawLine(int x1, int y1, int x2, int y2):** is used to draw line between the points(x1, y1) and (x2, y2).
7. **public abstract boolean drawImage(Image img, int x, int y, ImageObserver observer):** is used draw the specified image.
8. **public abstract void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle):** is used draw a circular or elliptical arc.
9. **public abstract void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle):** is used to fill a circular or elliptical arc.
10. **public abstract void setColor(Color c):** is used to set the graphics current color to the specified color.

**public abstract void setFont(Font font):** is used to set the graphics current font to the specified font.

Graphicsdemo.java

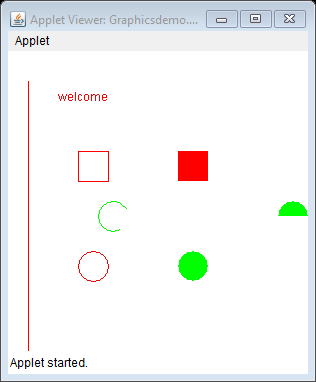
|  |
| --- |
| import java.applet.Applet;  import java.awt.\*;  public class Graphicsdemo extends Applet  {  public void paint(Graphics g)  {  g.setColor(Color.red);  g.drawString("welcome",50,50);  g.drawLine(20,30,20,300);  g.drawRect(70,100,30,30);  g.fillRect(170,100,30,30);  g.drawOval(70,200,30,30);  g.setColor(Color.green);  g.fillOval(170,200,30,30);  g.drawArc(90,150,30,30,30,270);  g.fillArc(270,150,30,30,0,180);  }  } |

Graphicsdemo.html

|  |
| --- |
| <html>  <body>  <applet code="Graphicsdemo.class" width=300 height=300></applet>  </body>  </html> |

To run

|  |
| --- |
| E:\javaprgs\applets>javac Graphicsdemo.java  E:\javaprgs\applets>appletviewer Graphicsdemo.html |



Another example to draw a polygon

drawpolygon.java

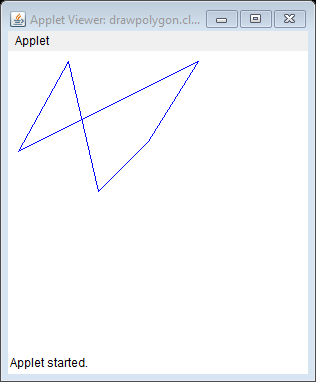
|  |
| --- |
| import java.applet.Applet;  import java.awt.\*;  public class drawpolygon extends Applet  {  public void paint(Graphics g)  {  int n=5;  int xdata[]={10,60,90,140,190};  int ydata[]={100,10,140,90,10};  g.setColor(Color.blue);  g.drawPolygon(xdata,ydata,n);  }  } |

drawpolygon.html

|  |
| --- |
| <html>  <body>  <applet code="drawpolygon.class" width=300 height=300></applet>  </body>  </html> |

Running the above codes

|  |
| --- |
| E:\javaprgs\applets>javac drawpolygon.java  E:\javaprgs\applets>appletviewer drawpolygon.html |



Displaying image in Applet

Applet is mostly used in games and animation. For this purpose image is required to be displayed. The java.awt.Graphics class provide a method drawImage() to display the image.

## **Syntax of drawImage() method:**

|  |
| --- |
| 1. **public abstract boolean drawImage(Image img, int x, int y, ImageObserver observer):** is used draw the specified image. |

## **How to get the object of Image:**

|  |
| --- |
| The java.applet.Applet class provides getImage() method that returns the object of Image. Syntax: |

1. public Image getImage(URL u, String image){}

## **Other required methods of Applet class to display image:**

|  |
| --- |
| 1. **public URL getDocumentBase():** is used to return the URL of the document in which applet is embedded. 2. **public URL getCodeBase():** is used to return the base URL. |

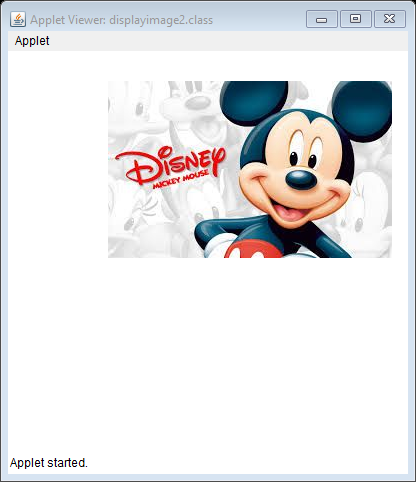
Displayimage2.java

|  |
| --- |
| import java.applet.\*;  import java.awt.\*;  public class displayimage2 extends Applet  {  Image pic;  public void init()  {  pic=getImage(getDocumentBase(),"file:/e:/javaprgs/applets/mickey.jpg");  }  public void paint(Graphics g)  {  int width=pic.getWidth(this);  int height=pic.getHeight(this);  g.drawImage(pic,100,30,width,height,this);  }  } |

Displayimage2.html

|  |
| --- |
| <html>  <body>  <applet code="displayimage1.class" width="400" height="400">  </applet>  </body>  </html> |

output



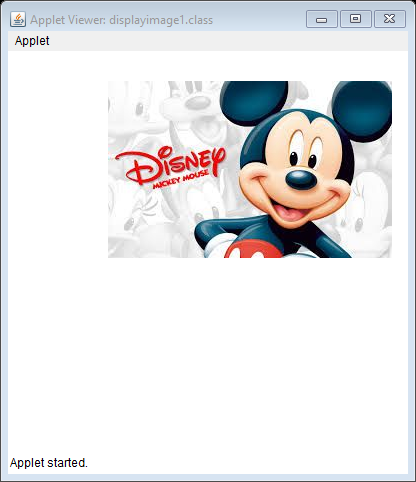
With parameters

displayimage1.java

|  |
| --- |
| import java.applet.\*;  import java.awt.\*;  public class displayimage1 extends Applet  {  Image pic;  public void init()  {  pic=getImage(getDocumentBase(),getParameter("image1"));  }  public void paint(Graphics g)  {  g.drawImage(pic,100,30,this);  }  } |

Displayimage1.html

|  |
| --- |
| <html>  <body>  <applet code="displayimage1.class" width="400" height="400">  <param name="image1" value="mickey.jpg">  </applet>  </body>  </html> |



Another example where we pass string as parameter

appletparam1.java

|  |
| --- |
| // applet prgrm for parameter  import java.awt.\*;  import java.applet.\*;  public class appletparam1 extends Applet  {  public void paint(Graphics g)  {  String str=getParameter("message");  g.drawString(str,100,100);  }  } |

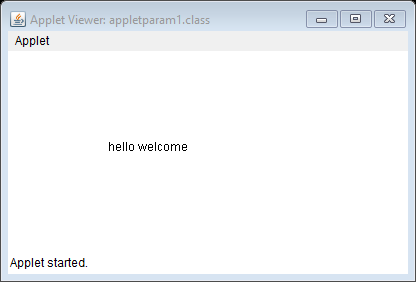
appletparam1.html

|  |
| --- |
| <html>  <body>  <applet code="appletparam1.class" width=400 height=200>  <param name="message" value="hello welcome">  </applet>  </body>  </html> |

To run the program

|  |
| --- |
| E:\javaprgs\applets>javac -cp . appletparam1.java  E:\javaprgs\applets>appletviewer appletparam1.html |

Output



Example

appletparam2.java

|  |
| --- |
| // applet to compute a numeric value and return it -prgrm for parameter  import java.awt.\*;  import java.applet.\*;  public class appletparam2 extends Applet  {  public void paint(Graphics g)  {  int a=11;  int b=29;  int sum=a+b;  String str="sum "+String.valueOf(sum);  g.drawString(str,100,100);  }  } |

appletparam2.html

|  |
| --- |
| <html>  <body>  <applet code="appletparam2.class" width=400 height=200>  </applet>  </body>  </html> |