**Java Applet基础**

applet是一种Java程序。它一般运行在支持Java的Web浏览器内。因为它有完整的Java API支持,所以applet是一个全功能的Java应用程序。

如下所示是独立的Java应用程序和applet程序之间重要的不同：

* Java中applet类继承了 java.applet.Applet类
* Applet类没有定义main()，所以一个 Applet程序不会调用main()方法，
* Applets被设计为嵌入在一个HTML页面。
* 当用户浏览包含Applet的HTML页面，Applet的代码就被下载到用户的机器上。
* 要查看一个applet需要JVM。 JVM可以是Web浏览器的一个插件，或一个独立的运行时环境。
* 用户机器上的JVM创建一个applet类的实例，并调用Applet生命周期过程中的各种方法。
* Applets有Web浏览器强制执行的严格的安全规则，applet的安全机制被称为沙箱安全。
* applet需要的其他类可以用Java归档（JAR）文件的形式下载下来。

**Applet的生命周期**

Applet类中的四个方法给你提供了一个框架，你可以再该框架上开发小程序：

* **init:**该方法的目的是为你的applet提供所需的任何初始化。在Applet标记内的param标签被处理后调用该方法。
* **start:** 浏览器调用init方法后，该方法被自动调用。每当用户从其他页面返回到包含Applet的页面时，则调用该方法。
* **stop:**当用户从包含applet的页面移除的时候，该方法自动被调用。因此，可以在相同的applet中反复调用该方法。
* **destroy:**此方法仅当浏览器正常关闭时调用。因为applets只有在HTML网页上有效，所以你不应该在用户离开包含Applet的页面后遗漏任何资源.
* **paint:** 该方法在start()方法之后立即被调用，或者在applet需要重绘在浏览器的时候调用。paint()方法实际上继承于java.awt。

**"Hello, World" Applet:**

下面是一个简单的Applet程序HelloWorldApplet.java:

import java.applet.\*;

import java.awt.\*;

public class HelloWorldApplet extends Applet

{

public void paint (Graphics g)

{

g.drawString ("Hello World", 25, 50);

}

}

这些import语句将以下类导入到我们的applet类中：

java.applet.Applet.

java.awt.Graphics.

没有这些import语句，Java编译器就识别不了Applet和Graphics类。

**Applet 类**

每一个applet都是java.applet.Applet 类的子类，基础的Applet类提供了供衍生类调用的方法,以此来得到浏览器上下文的信息和服务。

这些方法做了如下事情：

* 得到applet的参数
* 得到包含applet的HTML文件的网络位置
* 得到applet类目录的网络位置
* 打印浏览器的状态信息
* 获取一张图片
* 获取一个音频片段
* 播放一个音频片段
* 调整此 applet 的大小

除此之外，Applet类还提供了一个接口，该接口供Viewer或浏览器来获取applet的信息，并且来控制applet的执行。

Viewer可能是：

* 请求applet作者、版本和版权的信息
* 请求applet识别的参数的描述
* 初始化applet
* 销毁applet
* 开始执行applet
* 结束执行applet

Applet类提供了对这些方法的默认实现，这些方法可以在需要的时候重写。

"Hello，World"applet都是按标准编写的。唯一被重写的方法是paint方法。

**Applet的调用**

applet是一种Java程序。它一般运行在支持Java的Web浏览器内。因为它有完整的Java API支持,所以applet是一个全功能的Java应用程序。

<applet>标签是在HTML文件中嵌入applet的基础。以下是一个调用"Hello World"applet的例子；

<html>

<title>The Hello, World Applet</title>

<hr>

<applet code="HelloWorldApplet.class" width="320" height="120">

If your browser was Java-enabled, a "Hello, World"

message would appear here.

</applet>

<hr>

</html>

**注意:**你可以参照HTML Applet标签来更多的了解从HTML中调用applet的方法。

<applet>标签的属性指定了要运行的Applet类。Width和height用来指定applet运行面板的初始大小。applet必须使用</applet>标签来关闭。

如果applet接受参数，那么参数的值需要在标签里添加，该标签位于<applet>和</applet>之间。浏览器忽略了applet标签之间的文本和其他标签。

不支持Java的浏览器不能执行<applet>和</applet>。因此，在标签之间显示并且和applet没有关系的任何东西，在不支持的Java的浏览器里是可见的。

Viewer或者浏览器在文档的位置寻找编译过的Java代码，要指定文档的路径，得使用<applet>标签的codebase属性指定。

如下所示：

<applet codebase="http://amrood.com/applets"

code="HelloWorldApplet.class" width="320" height="120">

如果applet所在一个包中而不是默认包，那么所在的包必须在code属性里指定，例如：

<applet code="mypackage.subpackage.TestApplet.class"

width="320" height="120">

**获得applet参数**

下面的例子演示了如何使用一个applet响应来设置文件中指定的参数。该Applet显示了一个黑色棋盘图案和第二种颜色。

第二种颜色和每一列的大小通过文档中的applet的参数指定。

CheckerApplet 在init()方法里得到它的参数。也可以在paint()方法里得到它的参数。然而，在applet开始得到值并保存了设置，而不是每一次刷新的时候都得到值，这样是很方便，并且高效的。

applet viewer或者浏览器在applet每次运行的时候调用init()方法。在加载applet之后，Viewer立即调用init()方法（Applet.init()什么也没做），重写该方法的默认实现，添加一些自定义的初始化代码。

Applet.getParameter()方法通过给出参数名称得到参数值。如果得到的值是数字或者其他非字符数据，那么必须解析为字符串类型。

下例是CheckerApplet.java的梗概：

import java.applet.\*;

import java.awt.\*;

public class CheckerApplet extends Applet

{

int squareSize = 50;// 初始化默认大小

public void init () {}

private void parseSquareSize (String param) {}

private Color parseColor (String param) {}

public void paint (Graphics g) {}

}

下面是CheckerApplet类的init()方法和私有的parseSquareSize()方法：

public void init ()

{

String squareSizeParam = getParameter ("squareSize");

parseSquareSize (squareSizeParam);

String colorParam = getParameter ("color");

Color fg = parseColor (colorParam);

setBackground (Color.black);

setForeground (fg);

}

private void parseSquareSize (String param)

{

if (param == null) return;

try {

squareSize = Integer.parseInt (param);

}

catch (Exception e) {

// 保留默认值

}

}

该applet调用parseSquareSize()，来解析squareSize参数。parseSquareSize()调用了库方法Integer. parseInt()，该方法将一个字符串解析为一个整数，当参数无效的时候，Integer.parseInt()抛出异常。

因此，parseSquareSize()方法也是捕获异常的，并不允许applet接受无效的输入。

Applet调用parseColor()方法将颜色参数解析为一个Color值。parseColor()方法做了一系列字符串的比较，来匹配参数的值和预定义颜色的名字。你需要实现这些方法来使applet工作。

**指定applet参数**

如下的例子是一个HTML文件，其中嵌入了CheckerApplet类。HTML文件通过使用标签的方法给applet指定了两个参数。

<html>

<title>Checkerboard Applet</title>

<hr>

<applet code="CheckerApplet.class" width="480" height="320">

<param name="color" value="blue">

<param name="squaresize" value="30">

</applet>

<hr>

</html>

**注意:**参数名字大小写不敏感。

**应用程序转换成Applet**

将图形化的Java应用程序（是指，使用AWT的应用程序和使用java程序启动器启动的程序）转换成嵌入在web页面里的applet是很简单的。

下面是将应用程序转换成applet的几个步骤：

* 编写一个HTML页面，该页面带有能加载applet代码的标签。
* 编写一个JApplet类的子类，将该类设置为public。否则，applet不能被加载。
* 消除应用程序的main()方法。不要为应用程序构造框架窗口，因为你的应用程序要显示在浏览器中。
* 将应用程序中框架窗口的构造方法里的初始化代码移到applet的init()方法中，你不必显示的构造applet对象，浏览器将通过调用init()方法来实例化一个对象。
* 移除对setSize()方法的调用，对于applet来讲，大小已经通过HTML文件里的width和height参数设定好了。
* 移除对 setDefaultCloseOperation()方法的调用。Applet不能被关闭，它随着浏览器的退出而终止。
* 如果应用程序调用了setTitle()方法，消除对该方法的调用。applet不能有标题栏。（当然你可以给通过html的title标签给网页自身命名）
* 不要调用setVisible(true),applet是自动显示的。

**事件处理**

Applet类从Container类继承了许多事件处理方法。Container类定义了几个方法，例如：processKeyEvent()和processMouseEvent()，用来处理特别类型的事件，还有一个捕获所有事件的方法叫做processEvent。

为了响应一个事件，applet必须重写合适的事件处理方法。

import java.awt.event.MouseListener;

import java.awt.event.MouseEvent;

import java.applet.Applet;

import java.awt.Graphics;

public class ExampleEventHandling extends Applet

implements MouseListener {

StringBuffer strBuffer;

public void init() {

addMouseListener(this);

strBuffer = new StringBuffer();

addItem("initializing the apple ");

}

public void start() {

addItem("starting the applet ");

}

public void stop() {

addItem("stopping the applet ");

}

public void destroy() {

addItem("unloading the applet");

}

void addItem(String word) {

System.out.println(word);

strBuffer.append(word);

repaint();

}

public void paint(Graphics g) {

//Draw a Rectangle around the applet's display area.

g.drawRect(0, 0,

getWidth() - 1,

getHeight() - 1);

//display the string inside the rectangle.

g.drawString(strBuffer.toString(), 10, 20);

}

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) {

addItem("mouse clicked! ");

}

}

如下调用该applet：

<html>

<title>Event Handling</title>

<hr>

<applet code="ExampleEventHandling.class"

width="300" height="300">

</applet>

<hr>

</html>

最开始运行，applet显示 "initializing the applet. Starting the applet."，然后你一点击矩形框，就会显示"mouse clicked" 。

**显示图片**

applet能显示GIF,JPEG,BMP等其他格式的图片。为了在applet中显示图片，你需要使用java.awt.Graphics类的drawImage()方法。

如下实例演示了显示图片的所有步骤：

import java.applet.\*;

import java.awt.\*;

import java.net.\*;

public class ImageDemo extends Applet

{

private Image image;

private AppletContext context;

public void init()

{

context = this.getAppletContext();

String imageURL = this.getParameter("image");

if(imageURL == null)

{

imageURL = "java.jpg";

}

try

{

URL url = new URL(this.getDocumentBase(), imageURL);

image = context.getImage(url);

}catch(MalformedURLException e)

{

e.printStackTrace();

// Display in browser status bar

context.showStatus("Could not load image!");

}

}

public void paint(Graphics g)

{

context.showStatus("Displaying image");

g.drawImage(image, 0, 0, 200, 84, null);

g.drawString("www.javalicense.com", 35, 100);

}

}

如下调用该applet：

<html>

<title>The ImageDemo applet</title>

<hr>

<applet code="ImageDemo.class" width="300" height="200">

<param name="image" value="java.jpg">

</applet>

<hr>

</html>

**播放音频**

Applet能通过使用java.applet包中的AudioClip接口播放音频。AudioClip接口定义了三个方法：

* **public void play():** 从一开始播放音频片段一次。
* **public void loop():**循环播放音频片段
* **public void stop():**停止播放音频片段

为了得到AudioClip对象，你必须调用Applet类的getAudioClip()方法。无论URL指向的是否是一个真实的音频文件，该方法都会立即返回结果。

直到要播放音频文件时，该文件才会下载下来。

如下实例演示了播放音频的所有步骤：

import java.applet.\*;

import java.awt.\*;

import java.net.\*;

public class AudioDemo extends Applet

{

private AudioClip clip;

private AppletContext context;

public void init()

{

context = this.getAppletContext();

String audioURL = this.getParameter("audio");

if(audioURL == null)

{

audioURL = "default.au";

}

try

{

URL url = new URL(this.getDocumentBase(), audioURL);

clip = context.getAudioClip(url);

}catch(MalformedURLException e)

{

e.printStackTrace();

context.showStatus("Could not load audio file!");

}

}

public void start()

{

if(clip != null)

{

clip.loop();

}

}

public void stop()

{

if(clip != null)

{

clip.stop();

}

}

}

如下调用applet：

<html>

<title>The ImageDemo applet</title>

<hr>

<applet code="ImageDemo.class" width="0" height="0">

<param name="audio" value="test.wav">

</applet>

<hr>

你可以使用你电脑上的test.wav来测试上面的实例。