[**Java的SystemTray类的使用方法**](http://kayo.iteye.com/blog/737360)

**博客分类：**

* [JAVA体系](http://kayo.iteye.com/category/28406)

[Java](http://www.iteye.com/blogs/tag/Java)[Swing](http://www.iteye.com/blogs/tag/Swing)

Java的SystemTray类的使用方法

JDK1.6版增加了java.awt.SystemTray类，该类表示桌面的系统托盘，这对大多数系统托盘应用已经足够。如果这个类还是不能满足您的需求，可以试一试JDIC （JDesktop Integration Components）。  
JDK1.6的API文档，给出了这个类的用法。有人愿意把创建系统托盘的代码直接放在主应用代码中，笔者更倾向于把创建系统托盘的责任分派给一个单独的类，这符合类的设计原则。在主应用中直接调用该类的工厂方法就可以了，结构清晰，便于维护，改动或增加系统托盘的功能，不必改动主应用代码。  
创建系统托盘的类(TrayFactory.java)的代码如下：

[[Copy to clipboard]](http://kayo.iteye.com/admin/blogs/###) [[ - ]](http://kayo.iteye.com/admin/blogs/###)

CODE:

//package com.jnmc.liang.systemtray;  
  
import java.awt.\*;  
import java.awt.event.\*;  
  
public class TrayFactory {  
        private TrayIcon trayIcon = null;  
        private Component comp = null;  
        private String trayImageName = null;  
        private String tooltip = null;  
          
        private TrayFactory(Component comp, String trayImageName, String tooltip) {  
                this.comp = comp;  
                this.trayImageName = trayImageName;  
                this.tooltip = tooltip;  
                  
                init();  
        }  
          
        private void init() {  
                if (SystemTray.isSupported()) {  
                        SystemTray tray = SystemTray.getSystemTray();  
                        Image image  = Toolkit.getDefaultToolkit().getImage(trayImageName);  
                        PopupMenu popupMenu = new PopupMenu();  
                        MenuItem restoreItem = new MenuItem("还原");   
                        MenuItem exitItem  = new MenuItem("退出");  
                          
                        restoreItem.addActionListener(new RestoreItemActionListener());                          
                        exitItem.addActionListener(new ExitItemActionListener());  
                          
                        popupMenu.add(restoreItem);  
                        popupMenu.addSeparator();  
                        popupMenu.add(exitItem);  
                          
                        trayIcon = new TrayIcon(image, tooltip,  popupMenu);  
                        trayIcon.addMouseListener(new TrayIconMouseListener());  
                        //trayIcon.addActionListener(new TrayIconActionListener());  
                                                  
                        try {  
                                tray.add(trayIcon);  
                        }  
                        catch (AWTException e) {  
                                System.err.println(e);  
                        }  
                }  
                else {  
                        //..........  
                }  
        }  
  
        private class RestoreItemActionListener implements ActionListener {  
                public void actionPerformed(ActionEvent ae) {  
                        comp.setVisible(true);  
                }  
        }  
                  
        private class ExitItemActionListener implements ActionListener {  
                public void actionPerformed(ActionEvent ae) {  
                        System.exit(0);  
                }  
        }  
          
        private class TrayIconMouseListener extends MouseAdapter {  
                public void mousePressed(MouseEvent me) {  
                        if (me.getButton() == MouseEvent.BUTTON1) {  
                                comp.setVisible(true);  
                        }  
                }  
        }  
/\*  
        private class TrayIconActionListener implements ActionListener {  
                public void actionPerformed(ActionEvent ae) {  
                        comp.setVisible(true);  
                }  
        }  
\*/  
        public static void createTray(Component comp, String trayImageName, String tooltip) {  
                new TrayFactory(comp, trayImageName, tooltip);  
        }  
}

这个类的构造方法是private的，因为创建该类的实例不是我们的目的，另一方面是为了方便系统托盘的创建。  
构造方法有3个参数：  
第1个参数是一个Component，这就是您所要显示或隐藏的主窗体。当然，您把这个参数写成Component的子类也无妨，这里是为了增加类的通用性。  
第2个参数是一个String，它是一个image文件的名称，将来用这个文件创建托盘图标TrayIcon。  
第3个参数也是一个String，它是工具提示字符串，当您把鼠标放到托盘图标上的时候，托盘图标将弹出这个字符串的内容。  
创建系统托盘的具体代码在方法private void init()中。  
创建系统托盘的任务分为两个步骤：  
首先创建托盘图标（TrayIcon），其次把托盘图标添加到系统托盘(SystemTray)中。  
Java.awt.TrayIcon提供3种构造方法创建TrayIcon实例，这里采用第三种构造方法：  
                trayIcon = new TrayIcon(image, tooltip,  popupMenu);  
其中，image对象由TrayFactory类构造方法的第二个参数trayImageName创建：  
Image image  = Toolkit.getDefaultToolkit().getImage(trayImageName);  
tooltip就是TrayFactory类构造方法的第三个参数  
由于不能朝前引用，PopupMenu对象popupMenu必须在创建trayIcon对象之前创建，并给popup菜单项添加动作监听器来处理菜单项的动作（点击）事件，由于处理动作事件比较简单，这里不做具体分析，只对TrayIcon的鼠标事件做出说明。  
我们希望在托盘图标上按下鼠标左键时显示主窗体，所以必须处理托盘图标的鼠标按下事件。我们用TrayIconMouseListener私有内部类（private inner class）来处理鼠标按下事件。不论按下鼠标左键还是右键，都会产生鼠标事件，都会引起mousePressed方法的调用，所以在这个方法中只处理左键而忽略右键，把右键留给popup菜单：  
                        if (me.getButton() == MouseEvent.BUTTON1) {  
                                comp.setVisible(true);  
                        }  
这时，就可以给init()方法中创建的TrayIcon对象trayIcon添加鼠标监听器了：  
trayIcon.addMouseListener(new TrayIconMouseListener());  
利用TrayFactory类对主应用创建系统托盘时，只需在主应用中调用该类的工厂方法即可：  
TrayFactory.createTray(comp, trayImageName, tooltip);  
注意这一句，一定要放在适当的位置：主应用从开始启动到启动完毕，必须有机会执行这一句。所以建议把这一句放在主应用的构造方法中，或者干脆放在显示主窗体的前一个语句的位置：  
  
//其他语句  
               JFrame mainFrame = new JFrame("测试主窗体");  
                mainFrame.getContentPane().add(otherVisibleObj, BorderLayout.CENTER);  
                mainFrame.pack();  
                **TrayFactory.createTray(mainFrame, "images/icon.png", "这是工具提示");**          
                mainFrame.setVisible(true);  
下面给出一个完整的例子。  
主应用（MainApp.java）源代码如下：

[[Copy to clipboard]](http://kayo.iteye.com/admin/blogs/###) [[ - ]](http://kayo.iteye.com/admin/blogs/###)

CODE:

import java.awt.\*;  
import javax.swing.\*;  
  
public class MainApp {  
        private static void createAndShowGUI() {  
                JFrame mainFrame = new JFrame("测试主窗体");  
                JLabel jLabel = new JLabel("这是一个标签", SwingConstants.CENTER);  
                jLabel.setPreferredSize(new Dimension(400, 300));  
    mainFrame.getContentPane().add(jLabel, BorderLayout.CENTER);  
      
                mainFrame.pack();  
                TrayFactory.createTray(mainFrame, "images/icon.png", "你好，Java!\nHello,Java!");  
                mainFrame.setVisible(true);  
        }  
      
        public static void main(String[] args) {  
                String lf = UIManager.getSystemLookAndFeelClassName();  
                try {  
                        UIManager.setLookAndFeel(lf);  
                }   
                catch(Exception e) {  
                }  
                  
                javax.swing.SwingUtilities.invokeLater(new Runnable() {  
                                public void run() {  
                                        createAndShowGUI();  
                                }  
    });  
        }      
}

为简单起见，可以把TrayFactory.java和MainApp.java放在同一目录下，把icon.png文件放在当前目录的images目录下。