搭建Java桌面应用程序原型(三) PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/253/2021_2022__E6_90_AD_E5_BB_BAJava_c97_253056.htm 做技术决定 在开发过程中，我不得不去解决一些技术问题并且要做一些技术决定。下面的代码片段仅仅简单的进行了解释，但是他们将在我以后的文章中被详细描述。在这里重要的是去理解原型充当的角色。用你的原型去寻找技术问题的解决方案，去测试不常用的APIs，并且保证你的应用程序的性能。 用多层Panels 构建一个例如windows中的画板的图形应用程序不是非常复杂的任务。你必须处理鼠标事件、画线、画矩形和画椭圆。还要处理变形功能，比如从一个基础应用程序到一个专业级的图形编辑器要具有对图片的移动，缩放，重新排序、删除、复制、剪切和粘贴等更多的工作。你也可以想要包含一个可以进行编辑、重新控制大小和文字包装功能的文字框等等。构建自己的风格文本编辑器是没有必要的，因为Swing已经提供了一些文本组件。 你怎么将Swing的文本编辑器和你自己的绘图组件相集成？我考虑了两个解决方案。一个是实现一个类似于JTable所用的cell编辑器，但是如果你想改变文本框大小或者移动它就需要一点技巧了。另个一个解决方案是用JDesktopPane，把文本组件放在JInternalFrame之内。 用第二种解决方案的话，Swing已经提供了改变大小和移动功能，但是下面的问题是你怎么在包含文本注释的内置frame下绘制图象？并且你怎么在JDesktopPane上绘制其他简单图形，例如直线、矩形和椭圆？幸运的是，有一个简单的解决方案，因为JDesktopPane是真正的多层Panel。原型的MainPane类扩展

了JDesktopPane，有两层。它们中的一个包含PaintView自定义组件，允许你绘制简单图形。另一层包含文本注释。当然，如果一个注释图片不能被程序获得，那么这个解决办法是没有意义的。MainPanel的getAnnotatedImage()方法利用下面的代码做这件事：BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB).Graphics2D g = image.createGraphics().printAll(g).g.dispose(). 在paint()外部进行绘制 Swing组件的绘制通常都是在paint()内部进行或者在paint()内部调用绘制方法。当用鼠标在屏幕上绘制一个对象，可是，你不想重绘其他组件，因为这将引起应用程序运行效率低下。例如，用户用铅笔进行绘制，每个鼠标事件都让应用绘制一个小线段。在MOUSE_PRESSED和MOUSE_RELEASED之间有上百个MOUSE_DRAGGED事件。 当用户在屏幕仅仅绘制了一些图形时，重绘PaintView组件几百次这样的事情是不能被接受的。注意一下，PaintView处理大多数的绘制操作并且一个repaint需要所有注释包括文本注释进行重绘。正确的解决方案是当每个鼠标事件被处理时在paint()外部利用getGraphics()获得图形上下文。protected void toolAction(MouseEvent e) { e.consume(). Graphics2D g2 = (Graphics2D) getGraphics(). float zoomFactor = model.getZoomFactor(). g2.scale(zoomFactor, zoomFactor). float x = e.getX() / zoomFactor. float y = e.getY() / zoomFactor. currentTool.action(e.getID(), x, y, g2). g2.dispose().} PaintView组件利用鼠标监听去处理鼠标事件。上面的方法被每一个事件所调用，委托绘制currentTool对象。当鼠标释放的时候，repaint()被调用去请求整个组件的刷新。因此，用户完成图

形对象绘制后paint()仅仅被调用一次。这是注册鼠标监听的代码：protected void registerListeners() { addMouseListener(new MouseAdapter() { public void mousePressed(MouseEvent e) { if (SwingUtilities.isLeftMouseButton(e)) { requestFocus(). currentTool = model.createTool(AbstractTool.DRAW_STYLE). toolAction(e). } } public void mouseReleased(MouseEvent e) { if (SwingUtilities.isLeftMouseButton(e)) { toolAction(e). model.setLastTool(currentTool). currentTool = null. repaint(). } }}).addMouseMotionListener(new MouseMotionAdapter() { public void mouseDragged(MouseEvent e) { if (SwingUtilities.isLeftMouseButton(e)) toolAction(e). }})....} PaintView类的完整代码将在以后的文章中讲述。上面代码片段仅仅展示了怎么利用原型去做技术决定。 总结 原型在应用程序开发过程中有着重要的角色，允许你测试你的想法并且尽早的获得用户反馈。我没有把原型看成是当“ 真正 ”开发开始时可以被丢弃的代码片段。反而，原型应该是你产品或者着应用的基础。这意味着你应该小心的对它进行编码，尽管你的类或方法在以后要被重写。