

# 实验三：软件需求的跟踪分析

选定的开源 IDE 项目：IDEA。

实验过程：

明确提出需求 R 的文本，获取需求 R 的有关讨论文本；

需求 R 提出：Unable to toggle fullscreen view on linux。

项目地址: <https://github.com/JetBrains/intellij-community/pull/69>

相关文本：

Draft of supporting of FullScreen Mode at Linux. Checking on JRE 1.7, 1.8 XFCE with SawFish WM.

实现需求的代码：

```
platform/platform-api/src/com/intellij/openapi/wm/WindowManager.java
@@ -128,6 +128,6 @@ public StatusBar getStatusBar(@NotNull Component c, @Nullable Project project) {
128 128     * @return <code>true</code> if full screen mode is supported in current OS.
129 129     */
130 130     public boolean isFullScreenSupportedInCurrentOS() {
131 131     - return SystemInfo.isMacOSLion || SystemInfo.isWindows;
132 132     + return SystemInfo.isMacOSLion || SystemInfo.isWindows || (SystemInfo.isLinux && SystemInfo.isJavaVersionAtLeast("1.7"));
133 133     }
```

```
platform/platform-impl/src/com/intellij/openapi/wm/impl/IdeFrameImpl.java
@@ -359,7 +359,7 @@ private void installDefaultProjectStatusBarWidgets(@NotNull final Project projec
359 359
360 360     final EncodingPanel encodingPanel = new EncodingPanel(project);
361 361     statusBar.addWidget(encodingPanel, "after Position");
362 362     -
363 363     +
364 364     final LineSeparatorPanel lineSeparatorPanel = new LineSeparatorPanel(project);
365 365     statusBar.addWidget(lineSeparatorPanel, "before " + encodingPanel.ID());
@@ -480,7 +480,7 @@ public boolean isInFullScreen() {
480 480     if (SystemInfo.isMacOSLion) {
481 481         return myFrameDecorator != null && myFrameDecorator.isInFullScreen();
482 482     }
483 483     - if (SystemInfo.isWindows) {
484 484     + if (SystemInfo.isWindows || SystemInfo.isLinux) {
485 485         GraphicsDevice device = ScreenUtil.getScreenDevice(getBounds());
486 486         return (device != null && device.getDefaultConfiguration().getBounds().equals(getBounds()) && isUndecorated());
}
```

```
15 platform/platform-impl/src/com/intellij/openapi/wm/impl/WindowManagerImpl.java
833 833     }
834 834
835 835     - if (SystemInfo.isWindows) {
836 836     + if (SystemInfo.isWindows || SystemInfo.isLinux) {
837 837         GraphicsDevice device = ScreenUtil.getScreenDevice(frame.getBounds());
838 838         if (device == null) return;
839 838         try {
840 840     @@ -842,12 +842,19 @@ public void setFullScreen(IdeFrameImpl frame, boolean fullScreen) {
841 841     }
842 842     frame.dispose();
843 843     frame.setUndecorated(fullScreen);
844 844     +
845 845     + if (fullScreen) {
846 846     +     device.setFullScreenWindow(frame);
847 846     + } else {
848 847     +     device.setFullScreenWindow(null);
849 848     + }
850 849     }
851 850     }
852 851     finally {
853 852     if (fullScreen) {
854 853     - frame.setBounds(device.getDefaultConfiguration().getBounds());
855 854     - }
856 855     - else {
857 856     + if (SystemInfo.isWindows)
858 857     +     frame.setBounds(device.getDefaultConfiguration().getBounds());
859 858
860 859     } else {
861 860     Object o = frame.getRootPane().getClientProperty("oldBounds");
862 861     if (o instanceof Rectangle) {
863 862     frame.setBounds((Rectangle)o);
864 863     }
865 864     }
866 865     }
867 866     }
868 867     }
869 868     }
870 869     }
871 870     }
872 871     }
873 872     }
874 873     }
875 874     }
876 875     }
877 876     }
878 877     }
879 878     }
880 879     }
881 880     }
882 881     }
883 882     }
884 883     }
885 884     }
886 885     }
887 886     }
888 887     }
889 888     }
890 889     }
891 890     }
892 891     }
893 892     }
894 893     }
895 894     }
896 895     }
897 896     }
898 897     }
899 898     }
900 899     }
901 900     }
902 901     }
903 902     }
904 903     }
905 904     }
906 905     }
907 906     }
908 907     }
909 908     }
910 909     }
911 910     }
912 911     }
913 912     }
914 913     }
915 914     }
916 915     }
917 916     }
918 917     }
919 918     }
920 919     }
921 920     }
922 921     }
923 922     }
924 923     }
925 924     }
926 925     }
927 926     }
928 927     }
929 928     }
930 929     }
931 930     }
932 931     }
933 932     }
934 933     }
935 934     }
936 935     }
937 936     }
938 937     }
939 938     }
940 939     }
941 940     }
942 941     }
943 942     }
944 943     }
945 944     }
946 945     }
947 946     }
948 947     }
949 948     }
950 949     }
951 950     }
952 951     }
953 952     }
954 953     }
955 954     }
956 955     }
957 956     }
958 957     }
959 958     }
960 959     }
961 960     }
962 961     }
963 962     }
964 963     }
965 964     }
966 965     }
967 966     }
968 967     }
969 968     }
970 969     }
971 970     }
972 971     }
973 972     }
974 973     }
975 974     }
976 975     }
977 976     }
978 977     }
979 978     }
980 979     }
981 980     }
982 981     }
983 982     }
984 983     }
985 984     }
986 985     }
987 986     }
988 987     }
989 988     }
990 989     }
991 990     }
992 991     }
993 992     }
994 993     }
995 994     }
996 995     }
997 996     }
998 997     }
999 998     }
1000 999     }
```

vknavets on 9 Apr 2013 Author Contributor  
This is not needed for Linux since this was done with device.setFullScreenWindow  
Reply...

可以看出代码主要是增加了对操作系统的判断和处理  
同时作者表示在 java1.6 下存在该全屏问题，仅在 java1.7 版本后支持全屏，并且并没有能力在 Gnome3/2 或 KDE 桌面引擎下检测。  
文本：  
Since there is some issue with full screen in 1.6 of Java... Supporting only from 1.7... which is checks in the code...  
I don't have ability to check in Gnome3/2 or KDE...  
同时作者对之前全屏相关代码两次次修改，对系统进行检测（第二次主要为了缩短代码量）

```
10 platform/platform-impl/src/com/intellij/openapi/wm/impl/WindowManagerImpl.java
@@ -843,10 +843,12 @@ public void setFullScreen(IdeFrameImpl frame, boolean fullScreen) {
    843      frame.dispose();
    844      frame.setUndecorated(fullScreen);
    845
    846      - if (fullScreen) {
    847      -     device.setFullScreenWindow(frame);
    848      - } else {
    849      -     device.setFullScreenWindow(null);
    850      + if (SystemInfo.isLinux) {
    851      +     if (fullScreen) {
    852      +         device.setFullScreenWindow(frame);
    853      +     } else {
    854      +         device.setFullScreenWindow(null);
    855      +     }
    856      }
    857
    858 }
```

```
6 platform/platform-impl/src/com/intellij/openapi/wm/impl/WindowManagerImpl.java
@@ -844,11 +844,7 @@ public void setFullScreen(IdeFrameImpl frame, boolean fullScreen) {
    844      frame.setUndecorated(fullScreen);
    845
    846      if (SystemInfo.isLinux) {
    847      -     if (fullScreen) {
    848      -         device.setFullScreenWindow(frame);
    849      -     } else {
    850      -         device.setFullScreenWindow(null);
    851      -     }
    852      + device.setFullScreenWindow(fullScreen ? frame : null);
    853      }
    854
    855 }
```

之后有人回复表示必须在所有桌面管理引擎上进行 check，作者回复会在 VM 上测试然后给出了测试结果

Phanteon (Elementary Desktop/OS) - Passed

KDE - Passed

Gnome Classic - don't working at all =( (( It seems issue with compozite manager, need to check on the latest version

Gnome Classic (no effect) - Passed

Gnome Shell - Passed (issue with top bar depth, after switch (alt-tab) or Active IDEA menu e.g. Alt+F everything is ok)

XFCE - Passed with the same issue with Gnome Shell (after switching (alt-tab) or activating IDEA menu e.g. Alt-F works well)

Ubuntu Unity - Passed. works well if window is not maximized.

Thus I think in most of cases this code working well, but need to think about some tweaks for Unity and XFCE and Gnome Shell...

Any suggestion how I can check which WM is used?

之后又给出了 Gnome Classic 上不工作的原因

Problem with Gnome Classic is found. Related to Compiz.... To be able to have full screen

support in Compiz need to turn on Legacy Fullscreen Support in Workaraund settings.  
并且修改代码对不支持的 WM 给出提示信息

```
31 31 import com.intellij.openapi.diagnostic.Logger;
32 32 import com.intellij.openapi.project.Project;
33 33 import com.intellij.openapi.project.ProjectManager;
34 + import com.intellij.openapi.ui.Messages;
35 35 import com.intellij.openapi.ui.popup.JBPopup;
36 36 import com.intellij.openapi.util.Disposer;
37 37 import com.intellij.openapi.util.NamedJDOMExternalizable;

@@ -835,6 +836,10 @@ public void setFullScreen(IdeFrameImpl frame, boolean fullScreen) {
835 836     if (SystemInfo.isWindows || SystemInfo.isLinux) {
836 837         GraphicsDevice device = ScreenUtil.getScreenDevice(frame.getBounds());
837 838         if (device == null) return;
839 +         if (!device.isFullScreenSupported()) {
840 +             Messages.showWarningDialog("Sorry but yours Window Manager is not support Fullscreen mode", "Unsupported Window Manager");
841 +             return;
842 +         }
838 843         try {
839 844             frame.getRootPane().putClientProperty(ScreenUtil.DISPOSE_TEMPORARY, Boolean.TRUE);
840 845             if (fullScreen) {
@@ -844,7 +849,9 @@ public void setFullScreen(IdeFrameImpl frame, boolean fullScreen) {
844 849             frame.setUndecorated(fullScreen);
845 850
846 851             if (SystemInfo.isLinux) {
852 +                 frame.setResizable(!fullScreen);
847 853                 device.setFullScreenWindow(fullScreen ? frame : null);
854 +                 frame.validate();
848 855             }
849 856
850 857         }
@@ -858,7 +865,7 @@ public void setFullScreen(IdeFrameImpl frame, boolean fullScreen) {
858 865             frame.setBounds((Rectangle)o);
859 866         }
860 867     }
861 -     frame.setVisible(true);
862 +     if (!frame.isVisible()) frame.setVisible(true);
862 869     frame.getRootPane().putClientProperty(ScreenUtil.DISPOSE_TEMPORARY, null);
863 870 }
864 871 }
```

之后又对该段代码再次修改仅对 Linux 系统进行消息提示

```
platform/platform-impl/src/com/intellij/openapi/wm/impl/WindowManagerImpl.java
@@ -836,7 +836,7 @@ public void setFullScreen(IdeFrameImpl frame, boolean fullScreen) {
836 836     if (SystemInfo.isWindows || SystemInfo.isLinux) {
837 837         GraphicsDevice device = ScreenUtil.getScreenDevice(frame.getBounds());
838 838         if (device == null) return;
839 -         if (!device.isFullScreenSupported()) {
839 +         if (SystemInfo.isLinux && !device.isFullScreenSupported()) {
840 840             Messages.showWarningDialog("Sorry but yours Window Manager is not support Fullscreen mode", "Unsupported Window Manager");
841 841             return;
842 842         }
@@ -842,7 +842,7 @@ public void setFullScreen(IdeFrameImpl frame, boolean fullScreen) {
842 842     }
```

之后又修改了全屏模式的方法，使用了原生的 X11 调用

```

17 platform/platform-impl/src/com/intellij/openapi/wm/impl/WindowManagerImpl.java
@@ -31,7 +31,6 @@
31 31 import com.intellij.openapi.diagnostic.Logger;
32 32 import com.intellij.openapi.project.Project;
33 33 import com.intellij.openapi.project.ProjectManager;
34 - import com.intellij.openapi.ui.Messages;
35 34 import com.intellij.openapi.ui.popup.JBPopup;
36 35 import com.intellij.openapi.util.Disposer;
37 36 import com.intellij.openapi.util.NamedJDOMExternalizable;

@@ -42,6 +41,7 @@
42 41 import com.intellij.openapi.wm.ex.WindowManagerEx;
43 42 import com.intellij.openapi.wm.impl.welcomeScreen.WelcomeFrame;
44 43 import com.intellij.ui.ScreenUtil;
44 + import com.intellij.ui.X11FullscreenHelper;
45 45 import com.intellij.util.Alarm;
46 46 import com.intellij.util.EventDispatcher;
47 47 import com.intellij.util.messages.MessageBus;

@@ -836,24 +836,27 @@ public void setFullScreen(IdeFrameImpl frame, boolean fullScreen) {
836 836 if (SystemInfo.isWindows || SystemInfo.isLinux) {
837 837 GraphicsDevice device = ScreenUtil.getScreenDevice(frame.getBounds());
838 838 if (device == null) return;
839 - if (SystemInfo.isLinux && !device.isFullScreenSupported()) {
840 - Messages.showWarningDialog("Sorry but yours Window Manager is not support Fullscreen mo
841 - return;
842 - }
843 839 try {
844 840 frame.getRootPane().putClientProperty(ScreenUtil.DISPOSE_TEMPORARY, Boolean.TRUE);
845 841 if (fullScreen) {
846 842 frame.getRootPane().putClientProperty("oldBounds", frame.getBounds());
847 843 }
844 + // setUndecorated working only with not created window yet
848 845 frame.dispose();
849 846 frame.setUndecorated(fullScreen);
850 847
851 848 if (SystemInfo.isLinux) {
849 + // prevent resize of fullscreen window, to make sure that nothing bad will not happen
852 850 frame.setResizable(!fullScreen);
853 - device.setFullScreenWindow(fullScreen ? frame : null);
851 + // Set window bounds to screen size
852 + frame.setBounds(device.getDefaultConfiguration().getBounds());
853 + // Since we take from frame it's peer we need to make sure that it's was created
854 + // for this we create frame and reinitialize internal stuff by calling validate
855 + frame.setVisible(true);
854 856 frame.validate();
857 + // going to fullscreen and store result of operation in fullScreen state
858 + fullScreen = X11FullscreenHelper.setFullScreenWindow(frame, fullScreen);
855 859 }
856 -
857 860 }
858 861 finally {
859 862 if (fullScreen) {

```

增加新文件 platform/platform-impl/src/com/intellij/ui/X11FullscreenHelper.java 使用 X11 调用实现全屏。

新代码:



```

129 platform/platform-impl/src/com/intellij/ui/X11FullscreenHelper.java
... @@ -0,0 +1,129 @@
1  /*
2   * Copyright 2000-2013 JetBrains s.r.o.
3   *
4   * Licensed under the Apache License, Version 2.0 (the "License");
5   * you may not use this file except in compliance with the License.
6   * You may obtain a copy of the License at
7   *
8   * http://www.apache.org/licenses/LICENSE-2.0
9   *
10  * Unless required by applicable law or agreed to in writing, software
11  * distributed under the License is distributed on an "AS IS" BASIS,
12  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13  * See the License for the specific language governing permissions and
14  * limitations under the License.
15  */
16  package com.intellij.ui;
17
18  import com.sun.jna.Native;
19  import com.sun.jna.NativeLong;
20  import com.sun.jna.platform.unix.X11;
21  import sun.awt.X11.XAtom;
22  import sun.awt.X11.XBaseWindow;
23
24  import java.awt.*;
25
26  /**
27   * Created by IntelliJ IDEA.
28   * Author: Vladimir Kravets
29   * E-Mail: vova.kravets@gmail.com
30   * Date: 4/19/13
31   * Time: 12:19 AM
32   * based on code from VLCJ
33   * http://code.google.com/p/vlcj/source/browse/trunk/vlcj/src/main/java/uk/co/caprica/vlcj/runtime
34   */
35
36  public class X11FullscreenHelper {
37      /**
38       * Ask the window manager to make a window full-screen.
39       * <p>
40       * This method sends a low-level event to an X window to request that the
41       * window be made 'real' full-screen - i.e. the window will be sized to fill
42       * the entire screen bounds, and will appear <em>above</em> any window
43       * manager screen furniture such as panels and menus.
44       * <p>
45       * This method should only be called on platforms where X is supported.
46       * <p>
47       * The implementation makes use of the JNA X11 platform binding.
48       *
49       * @param w window to make full-screen
50       * @param fullScreen <code>true</code> to make the window full-screen; <code>false</code> to
51       * @return <code>true</code> if the message was successfully sent to the window; <code>false</code>
52       */
53      public static boolean setFullscreenWindow(Window w, boolean fullScreen) {
54          // Use the JNA platform X11 binding
55          X11 x = X11.INSTANCE;
56          X11.Display display = null;
57          try {
58              // Open the display
59              display = x.XOpenDisplay(null);
60              // Send the message
61
62              // Send change property before going to Fullscreen to make sure that WM will know that win
63              // Workaround for http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=7057287
64              XAtom.get("_NET_WM_STATE").setAtomListProperty(((XBaseWindow)w.getPeer()), new XAtom[] {XAtom
65
66              int result = sendClientMessage(
67                  display,
68                  Native.getWindowID(w),
69                  "_NET_WM_STATE",
70                  new NativeLong(fullScreen ? _NET_WM_STATE_ADD : _NET_WM_STATE_REMOVE),
71                  x.XInternAtom(display, "_NET_WM_STATE_FULLSCREEN", false),
72                  x.XInternAtom(display, "_NET_WM_STATE_ABOVE", false)
73              );
74              return result != 0;
75          }
76          finally {
77              if (display != null) {
78                  // Close the display
79                  x.XCloseDisplay(display);
80              }
81          }
82      }
83
84      /**
85       * Helper method to send a client message to an X window.
86       *
87       * @param display display
88       * @param wid native window identifier
89       * @param msg type of message to send
90       * @param data0 message data
91       * @param data1 message data
92       * @return <code>1</code> if the message was successfully sent to the window; <code>0</code>
93       */
94      private static int sendClientMessage(X11.Display display, long wid, String msg, NativeLong dat
95          // Use the JNA platform X11 binding
96          X11 x = X11.INSTANCE;
97          // Create and populate a client-event structure
98          X11.XEvent event = new X11.XEvent();
99          event.type = X11.ClientMessage;
100         // Select the proper union structure for the event type and populate it
101         event.setType(X11.XClientMessageEvent.class);
102         event.xclient.type = X11.ClientMessage;
103         event.xclient.serial = new NativeLong(0L);
104         event.xclient.send_event = 1;
105         event.xclient.message_type = x.XInternAtom(display, msg, false);
106         event.xclient.window = new X11.Window(wid);
107         event.xclient.format = 32;
108         // Select the proper union structure for the event data and populate it
109         event.xclient.data.setType(NativeLong[].class);
110         event.xclient.data.l[0] = data0;
111         event.xclient.data.l[1] = data1;
112         event.xclient.data.l[2] = data2;
113         event.xclient.data.l[3] = new NativeLong(0L);
114         event.xclient.data.l[4] = new NativeLong(0L);
115
116         // Send the event
117         NativeLong mask = new NativeLong(X11.SubstructureRedirectMask | X11.SubstructureNotifyMask);
118         int result = x.XSendEvent(display, x.DefaultRootWindow(display), 0, mask, event);
119         // Flush, since we're not processing an X event loop
120         x.XFlush(display);
121         // Finally, return the result of sending the event
122         return result;
123     }
124
125     // X window message definitions
126     private static final int _NET_WM_STATE_REMOVE = 0;
127     private static final int _NET_WM_STATE_ADD = 1;
128
129 }

```

然后应为使用了新的行为实现全屏功能，进行了小重构：

对 Linux 系统单独处理使用新的行为实现制在 linux 系统下的全屏：

```
5 platform/platform-impl/src/com/intellij/openapi/wm/impl/IdeFrameImpl.java ...

@@ -480,10 +480,13 @@ public boolean isInFullScreen() {
480 480     if (SystemInfo.isMacOSLion) {
481 481         return myFrameDecorator != null && myFrameDecorator.isInFullScreen();
482 482     }
483 - if (SystemInfo.isWindows || SystemInfo.isLinux) {
483 + if (SystemInfo.isWindows) {
484 484         GraphicsDevice device = ScreenUtil.getScreenDevice(getBounds());
485 485         return (device != null && device.getDefaultConfiguration().getBounds().equals(getBounds()))
486 486     }
487 + if (SystemInfo.isLinux) {
488 +     return X11FullscreenHelper.isInFullscreen();
489 + }
487 490     return false;
488 491 }
489 492
```

```

35 platform/platform-impl/src/com/intellij/ui/X11FullscreenHelper.java
18 18 import com.sun.jna.Native;
19 19 import com.sun.jna.NativeLong;
20 20 import com.sun.jna.platform.unix.X11;
21 - import sun.awt.X11.XAtom;
22 - import sun.awt.X11.XBaseWindow;
23 21
24 22 import java.awt.*;
25 23

* @@ -34,6 +32,9 @@
34 32 */
35 33
36 34 public class X11FullscreenHelper {
37 35 +
38 36 + private static boolean isFullscreenMode = false;
39 37 +
40 38 /**
41 39 * Ask the window manager to make a window full-screen.
42 40 * <p>
43 41
44 42 @@ -59,19 +60,20 @@ public static boolean setFullscreenWindow(Window w, boolean fullScreen) {
45 59 display = x.XOpenDisplay(null);
46 60 // Send the message
47 61
48 62 // Send change property before going to Fullscreen to make sure that WM will know that wir
49 63 // Workaround for http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=7057287
50 64 XAtom.get("_NET_WM_STATE").setAtomListProperty((XBaseWindow)w.getPeer(), new XAtom[] {XAtom
51 65
52 66 int result = sendClientMessage(
53 67 display,
54 68 Native.getWindowID(w),
55 69 "_NET_WM_STATE",
56 70 new NativeLong(fullScreen ? _NET_WM_STATE_ADD : _NET_WM_STATE_REMOVE),
57 71 x.XInternAtom(display, "_NET_WM_STATE_FULLSCREEN", false),
58 72 x.XInternAtom(display, "_NET_WM_STATE_ABOVE", false)
59 73 +
60 74 new NativeLong[]{
61 75 new NativeLong(fullScreen ? _NET_WM_STATE_ADD : _NET_WM_STATE_REMOVE),
62 76 x.XInternAtom(display, "_NET_WM_STATE_FULLSCREEN", false),
63 77 x.XInternAtom(display, "_NET_WM_STATE_ABOVE", false),
64 78 new NativeLong(0L),
65 79 new NativeLong(0L)
66 80 };
67 81 );
68 82 return result != 0;
69 83 isFullscreenMode = (result != 0) && fullScreen;
70 84 return (result != 0);
71 85 }
72 86 finally {
73 87 if(display != null) {
74 88 @@ -91,8 +93,9 @@ public static boolean setFullscreenWindow(Window w, boolean fullScreen) {
75 91 * @param data1 message data
76 92 * @return <code>1</code> if the message was successfully sent to the window; <code>0</code> c
77 93 */
78 94 - private static int sendClientMessage(X11.Display display, long wid, String msg, NativeLong dat
79 95 + private static int sendClientMessage(X11.Display display, long wid, String msg, NativeLong[] c
80 96 // Use the JNA platform X11 binding
81 97 + assert (data.length < 5);
82 98 X11 x = X11.INSTANCE;
83 99 // Create and populate a client-event structure
84 100 X11.XEvent event = new X11.XEvent();
85 101
86 102 @@ -107,11 +110,7 @@ private static int sendClientMessage(X11.Display display, long wid, String ms
87 107 event.xclient.format = 32;
88 108 // Select the proper union structure for the event data and populate it
89 109 event.xclient.data.setType(NativeLong[].class);
90 110 event.xclient.data.l[0] = data0;
91 111 event.xclient.data.l[1] = data1;
92 112 event.xclient.data.l[2] = data2;
93 113 event.xclient.data.l[3] = new NativeLong(0L);
94 114 event.xclient.data.l[4] = new NativeLong(0L);
95 115 + System.arraycopy(data, 0, event.xclient.data.l, 0, 5);
96 116
97 117 // Send the event
98 118 NativeLong mask = new NativeLong(X11.SubstructureRedirectMask | X11.SubstructureNotifyMask);
99 119 @@ -122,6 +121,10 @@ private static int sendClientMessage(X11.Display display, long wid, String ms
100 122 return result;
101 123 }
102 124
103 125 + public static boolean isInFullscreen() {
104 126 + return isFullscreenMode;
105 127 + }
106 128
107 129 // X window message definitions
108 130 private static final int _NET_WM_STATE_REMOVE = 0;
109 131 private static final int _NET_WM_STATE_ADD = 1;
110 132
111 133

```



```
32 platform/platform-impl/src/com/intellij/openapi/wm/impl/WindowManagerImpl.java ...

@@ -51,6 +51,7 @@
51 51 import org.jetbrains.annotations.NonNls;
52 52 import org.jetbrains.annotations.NotNull;
53 53 import org.jetbrains.annotations.Nullable;
54 + import sun.awt.X11.XToolkit;
55
56 56 import javax.swing.*;
57 57 import java.awt.*;

@@ -833,7 +834,7 @@ public void setFullScreen(IdeFrameImpl frame, boolean fullScreen) {
833 834 return;
834 835 }
835 836

836 - if (SystemInfo.isWindows || SystemInfo.isLinux) {
837 + if (SystemInfo.isWindows) {
837 838 GraphicsDevice device = ScreenUtil.getScreenDevice(frame.getBounds());
838 839 if (device == null) return;
839 840 try {

@@ -844,34 +845,31 @@ public void setFullScreen(IdeFrameImpl frame, boolean fullScreen) {
844 845 // setUndecorated working only with not created window yet
845 846 frame.dispose();
846 847 frame.setUndecorated(fullScreen);
847 -
848 - if (SystemInfo.isLinux) {
849 - // prevent resize of fullscreen window, to make sure that nothing bad will not happen
850 - frame.setResizable(!fullScreen);
851 - // Set window bounds to screen size
852 - frame.setBounds(device.getDefaultConfiguration().getBounds());
853 - // Since we take from frame it's peer we need to make sure that it's was created
854 - // for this we create frame and reinitialize internal stuff by calling validate
855 - frame.setVisible(true);
856 - frame.validate();
857 - // going to fullscreen and store result of operation in fullScreen state
858 - fullScreen = X11FullscreenHelper.setFullScreenWindow(frame, fullScreen);
859 - }
860 848 }
861 849 finally {
862 850 if (fullScreen) {
863 - if (SystemInfo.isWindows)
864 - frame.setBounds(device.getDefaultConfiguration().getBounds());
865 + frame.setBounds(device.getDefaultConfiguration().getBounds());
866 852 } else {
867 853 Object o = frame.getRootPane().getClientProperty("oldBounds");
868 854 if (o instanceof Rectangle) {
869 855 frame.setBounds((Rectangle)o);
870 856 }
871 857 }
872 - if (!frame.isVisible()) frame.setVisible(true);
873 + frame.setVisible(true);
874 859 frame.getRootPane().putClientProperty(ScreenUtil.DISPOSE_TEMPORARY, null);
875 860 }
876 861 }
877 + if (SystemInfo.isLinux) {
878 + // going to fullscreen using native X11 bindings
879 + // make sure that AWT thread will do nothing with window while it's going to fullscreen
880 + XToolkit.awtLock();
881 + try {
882 + X11FullscreenHelper.setFullScreenWindow(frame, fullScreen);
883 + } finally {
884 + // unlock AWT thread after finishing fullscreen switch
885 + XToolkit.awtUnlock();
886 + }
887 + }
888 873 }
889 874 finally {
890 875 frame.storeFullScreenStateIfNeeded(fullScreen);
891 876 }
```

重构之后又修复了一个关于 assert 的小 bug

```
platform/platform-impl/src/com/intellij/ui/X11FullscreenHelper.java
@@ -95,7 +95,7 @@ public static boolean setFullScreenWindow(Window w, boolean fullScreen) {
95 95  /*
96 96  private static int sendClientMessage(X11.Display display, long wid, String msg, NativeLong[] de
97 97  // Use the JNA platform X11 binding
98 98  - assert (data.length < 5);
98 98  + assert (data.length == 5);
99 99  X11 x = X11.INSTANCE;
100 100  // Create and populate a client-event structure
101 101  X11.XEvent event = new X11.XEvent();
```

之后作者表示在各个桌面管理引擎上测试都通过了, 并且由于最后的小重构使得全屏功能在 java1.6 上也能使用, 所以修改了最初的改动将 java 版本判断去掉了

```
,6 @@ public StatusBar getStatusBar(@NotNull Component c, @Nullable Project project) {
<code>true</code> if full screen mode is supported in current OS.

lean isFullScreenSupportedInCurrentOS() {
/systemInfo.isMacOSLion || SystemInfo.isWindows || (SystemInfo.isLinux && SystemInfo.isJavaVersionAtLeast("1.7"));
/systemInfo.isMacOSLion || SystemInfo.isWindows || SystemInfo.isLinux;
```

作者文本:

Finally with the last commit, should work with all WMs which support \_NET\_WM\_STATE\_FULLSCREEN.

By now check successfully with:

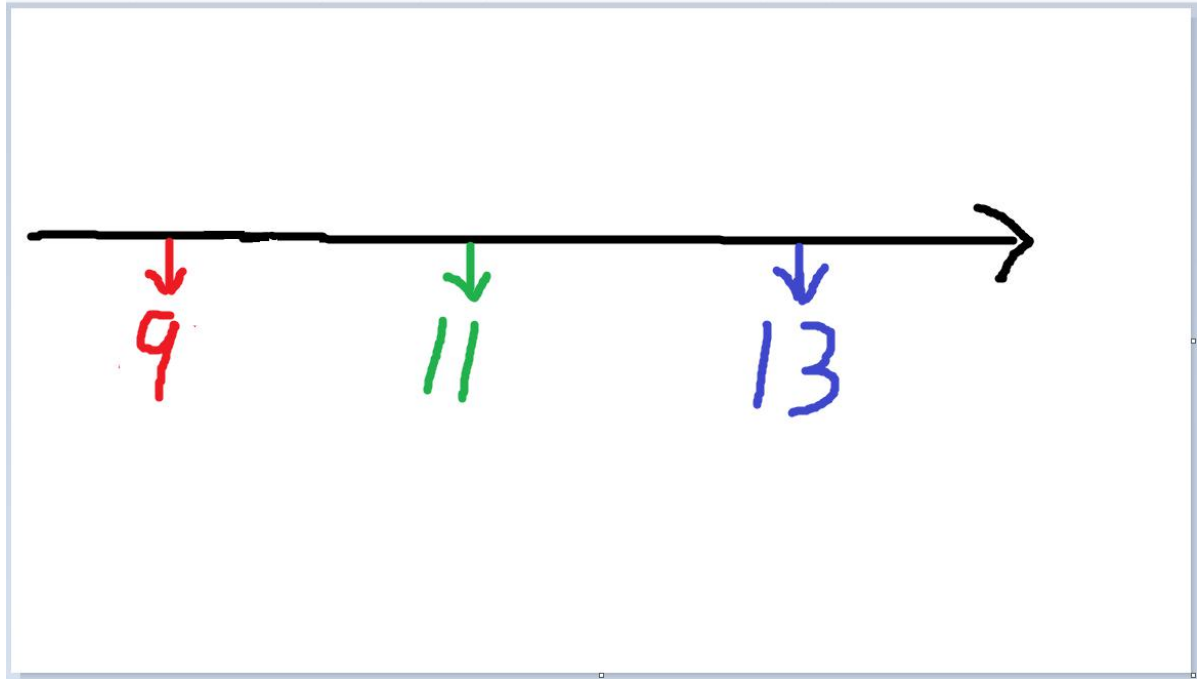
Ubuntu Unity  
GNOME3 Shell  
SawFish  
Enjoy!

Working everywhere. Update status of where its was checked:

Unity  
Unity2D  
Gnome Classic (Compiz)  
Gnome Classic (metacity)  
Gnome Shell (Gnome3)  
KDE  
SawFish  
XFCE/XFW  
Please review and merge with master

Last behavior of going to Fullscreen mode also support 1.6, 1.7 and 1.8 JRE. Thus checking runtime JRE was removed.

需求变更时间线



9: 4月9日修改, 给出了基本的判断代码, 并对修改代码进行了一些修改, 以完成工程开发的需要

11: 4月11日发现 Gnome Classic 上修改不工作的问题, 修改了在 Gnome Classic 上修改不工作的问题, 并相应地重构了代码

13: 4月13日修改, 完成了在各个平台上的全屏测试, 并且得益于4月11日的修改, 将代码试用范围推广至 java1.6