Swing URCap Toolbar Contributions

Universal Robots A/S

Version 1.11.0

Abstract

As of URCap API version 1.4, support for URCap toolbar contributions with a Swing-based user interface is introduced for version 5.0.0 of PolyScope. This document will describe how to contribute a toolbar view to the toolbar in PolyScope for a URCap.

Contents

1 Introduction

This document describes the purpose of URCap toolbar contributions in PolyScope and how to work with them through the URCap API.

2 Purpose of Toolbar Contributions

For an advanced user interface (UI), it can be an advantage to be able to display information from the URCap in a toolbar panel of its own. The panel can contain information retrieved from the URCap, e.g. gripper status or it could be buttons bound to URCap actions like "start welding".

Multiple URCaps from different providers can be installed on the robot at the same time and the toolbar functionality will allow URCaps to provide toolbar panels that will coexists with panels from other URCaps.

The individual panels are Swing JPanels and PolyScope provides functionality to bind these panels together and make them available via a toolbar of contributed URCap panels.

3 Contribution of a Panel to the Toolbar

A URCap can contribute one or more panels, available via the toolbar. Each panel is a Swing `JPanel` customizable by the URCap provider by means of other Java Swing GUI components.

3.1 Creating the Contributed Panel

To contribute a panel to the URCap, implement the `SwingToolbarContribution` interface. The interface has three methods that must be implemented:

- `void buildUI(JPanel)`: Called once at start up. The UI is built on the provided `JPanel`. Listing 1 shows how Swing GUI components are added to the `JPanel` to form the contribution. The provided panel will have a Swing UI manager already set, meaning that components without additional styling will look as native PolyScope ones including the correct font types. In order to resemble PolyScope the components should therefore only use limited styling, such as font sizes or component sizes (if the applied ones are not suitable). *Note: PolyScope will restrict the given `JPanel` in some ways to prevent a URCap from breaking the PolyScope UI and help the URCap provider to align with PolyScope UI styling.*
- `void openView()`: Called each time the user selects this URCap toolbar contribution in the toolbar
- `void closeView()`: Called each time the user exits this URCap toolbar contribution in the toolbar

Listing 1: MyToolbar toolbar contribution

```
1 package com.ur.urcap.examples;
2
3 import com.ur.urcap.api.contribution.toolbar.ToolbarContext;
4 import com.ur.urcap.api.contribution.toolbar.swing.SwingToolbarContribution;
5
6 import javax.swing.BorderFactory;
```

```
7  import javax.swing.Box;
8  import javax.swing.BoxLayout;
9  import javax.swing.JLabel;
10 import javax.swing.JPanel;
11 import java.awt.Component;
12 import java.awt.Dimension;
13 import java.awt.Font;
14 import java.util.Date;
15 import java.util.Locale;
16 import java.util.Random;
17
18 class MyToolbarContribution implements SwingToolbarContribution {
19   private static final int VERTICAL_SPACE = 10;
20   private static final int HEADER_FONT_SIZE = 24;
21
22   private final ToolbarContext context;
23   private JLabel demoToolStatus;
24
25   MyToolbarContribution(ToolbarContext context) {
26     this.context = context;
27   }
28
29   @Override
30   public void openView() {
31     demoToolStatus.setText("<HTML>" + get3rdPartyStatus() + "</HTML>");
32   }
33
34   @Override
35   public void closeView() {
36   }
37
38   public void buildUI(JPanel jPanel) {
39     jPanel.setLayout(new BoxLayout(jPanel, BoxLayout.Y_AXIS));
40
41     jPanel.add(createHeader());
42     jPanel.add(createVerticalSpace());
43     jPanel.add(createInfo());
44   }
45
46   private Box createHeader() {
47     Box headerBox = Box.createHorizontalBox();
48     headerBox.setAlignmentX(Component.CENTER_ALIGNMENT);
49
50     JLabel header = new JLabel("My␣Swing␣Toolbar");
51     header.setFont(header.getFont().deriveFont(Font.BOLD, HEADER_FONT_SIZE));
52     headerBox.add(header);
53     return headerBox;
54   }
55
56   private Box createInfo() {
57     Box infoBox = Box.createVerticalBox();
58     infoBox.setAlignmentX(Component.CENTER_ALIGNMENT);
59     JLabel pane1 = new JLabel();
60     pane1.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
61     pane1.setText("<HTML>This␣is␣a␣sample␣URCap␣Toolbar␣contribution.␣Feel␣
           free␣to␣use␣this␣as␣an␣example␣for␣creating␣new␣contributions.</HTML>"
           );
62     pane1.setBackground(infoBox.getBackground());
63     infoBox.add(pane1);
64
65     JLabel pane2 = new JLabel();
66     Locale locale = context.getAPIProvider().getSystemAPI().getSystemSettings
           ().getLocalization().getLocale();
67     pane2.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
```

```
68      pane2.setText("<HTML>Currently␣the␣robot␣is␣configured␣to␣use␣the␣Locale:
            ␣" + locale.getDisplayName() + "</HTML>");
69      infoBox.add(pane2);
70
71      demoToolStatus = new JLabel();
72      demoToolStatus.setText("<HTML>" + get3rdPartyStatus() +"</HTML>");
73      demoToolStatus.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
74      infoBox.add(demoToolStatus);
75      return infoBox;
76    }
77
78    private Component createVerticalSpace() {
79      return Box.createRigidArea(new Dimension(0, VERTICAL_SPACE));
80    }
81
82    private String get3rdPartyStatus() {
83      Date now = new Date();
84      int number = new Random().nextInt(10) + 20;
85      return  String.format("Tool␣status␣reading:␣%d,␣read␣at␣%tF␣%tT.", number,
            now, now);
86    }
87  }
```

3.2 Making the Toolbar Contribution available to PolyScope

To hook the toolbar panel into PolyScope, implement the `SwingToolbarService` interface. It provides means to attach an icon to identify the panel in the toolbar and configure the contribution:

- `Icon getIcon()`: Returns the icon for visual identification of the contributed panel in the toolbar
- `void configureContribution(ToolbarConfiguration)`: Use the argument supplied to configure the contribution. The configuration will already have default values for its properties (see default values in the Javadoc). If the default values are appropriate, leave this method empty. The values of the `ToolbarConfiguration` will be read once immediately after this method call.
- `SwingToolbarContribution createToolbar(ToolbarContext)`: Creates a new toolbar contribution instance. The `ToolbarContext` object gives access to services and APIs provided by PolyScope relevant to the toolbar contribution.

All methods are only called once on startup.

The `SwingToolbarService` needs to be registered in the OSGi BundleContext in a bundle activator. Listing 2 shows an implementaion of a `SwingToolbarService`.

Listing 2: MyToolbar toolbar contribution service

```
1  package com.ur.urcap.examples;
2
3  import com.ur.urcap.api.contribution.toolbar.ToolbarConfiguration;
4  import com.ur.urcap.api.contribution.toolbar.ToolbarContext;
5  import com.ur.urcap.api.contribution.toolbar.swing.SwingToolbarContribution;
6  import com.ur.urcap.api.contribution.toolbar.swing.SwingToolbarService;
7
8  import javax.swing.Icon;
9  import javax.swing.ImageIcon;
10
```

```
11  public class MyToolbarService implements SwingToolbarService {
12
13    @Override
14    public Icon getIcon() {
15      return new ImageIcon(getClass().getResource("/icons/acme_logo.png"));
16    }
17
18    @Override
19    public void configureContribution(ToolbarConfiguration configuration) {
20      configuration.setToolbarHeight(200);
21    }
22
23    @Override
24    public SwingToolbarContribution createToolbar(ToolbarContext context) {
25      return new MyToolbarContribution(context);
26    }
27  }
```

3.3 Layout of Toolbar Contributions

The layout of the content in the contributed toolbar panel must conform to the Universal Robot style guide to receive UR+ certification. The panel has a fixed width of 400px. The height must be in the range 48px to 600px and is set via the `setToolbarHeight()` method in the `ToolbarConfiguration` interface.

The toolbar contribution is accessed via the toolbar and this toolbar is accessed via the UR+ toggle button. When the UR+ toggle button is disabled, none of the toolbar contributions will be shown as depicted in figure 1.

Figure 1: URCap toolbar not enabled

When enabled, the provided toolbar panels will be accessible via a tabbed toolbar panel overlaying the right hand side of PolyScope.

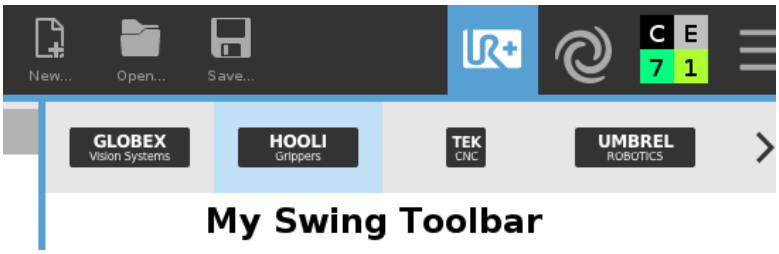

Figure 2: URCap toolbar enabled with multiple toolbar contributions

Pressing the icon identifying the individual toolbar in the tabbed panel will display the contributed toolbar panel. Figure 2 shows a toolbar with multiple toolbar contributions.

Like the content of the toolbar panel, the icons used to identify the panels in the toolbar should conform to the Universal Robots style guide. The icons are restricted to a width of 89px and a height of 30px.

Figure 3: Logo dimensions