Demo: Embedding Java GUI elements inside a Windows Presentation Foundation application

Version 10.1

jnbridge.com

JNBridge, LLC
jnbridge.com

August 13, 2019

Introduction

This document shows how a Java GUI element (an AWT or Swing component – SWT is not currently supported in this scenario) can be embedded inside a Windows Presentation Foundation (WPF) application. If you are unfamiliar with JNBridgePro, we recommend that you work through one of the other demos first. We recommend working through the "Log demo," which will work through the entire process of generating proxies and setting up, configuring, and running an interop project. This current document assumes such knowledge, and is mainly a guided tour of the code and configuration information necessary to embed Java GUI elements inside GUI-based WPF applications.

The Java GUI component

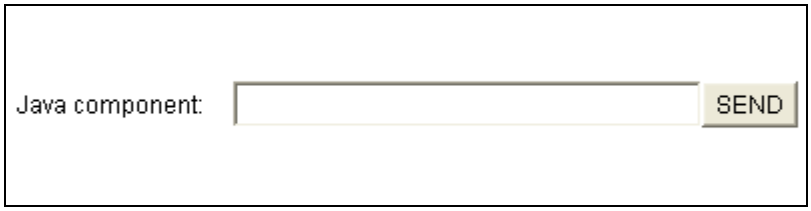
In this example, we have provided a simple AWT-based Java component, jInN.JavaComponent. (Swing components will work as well, but embedding SWT components inside WPF applications is not currently supported.)

```
public class JavaComponent extends Panel {
  public TextField javaTextBox;
  Button btn;

  public JavaComponent() {
    setSize(400, 100);
    setLayout(new GridBagLayout());
    add(new Label("Java component: "));
    javaTextBox = new TextField(30);
    btn = new Button("SEND");
    add(javaTextBox);
    add(btn);
  }

  public void addActionListener(ActionListener l)
  {
    btn.addActionListener(l);
  }
}
```

Any Java GUI component to be embedded inside a WinForm must be derived from java.awt.Component. The JavaComponent class above is derived from java.awt.Panel, which is a subclass of java.awt.Component. JavaComponent contains a TextField and a button; the TextField is public, so outside code can get and set the field's text, and a public method makes it possible to add an ActionListener for the button. JavaComponent should look like the following:

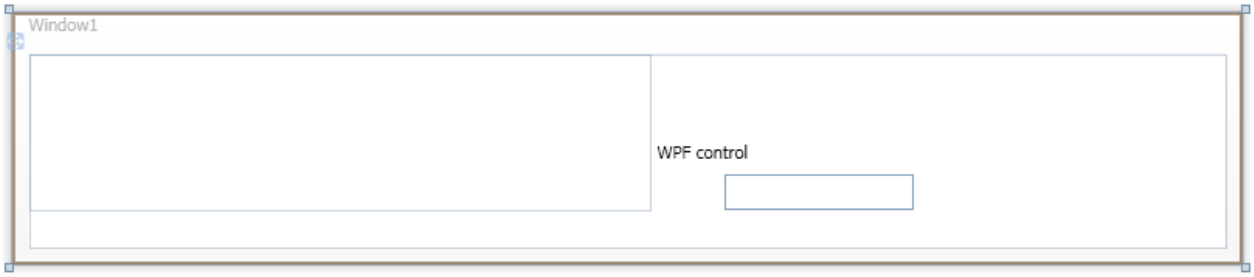


Generating the proxies

We have provided a proxy assembly javaComponentProxies.dll, which contains the proxies for jInN.JavaComponent, plus all supporting classes. However, it is straightforward to generate the proxies oneself.

Embedding the Java component inside the Windows Form

We have prepared a WPF application to contain the embedded Java component:



Note that the Window contains a WPF WrapPanel component, named javaHolder, which is the same size as the Java component, which will be embedded inside the panel.

Inside the window's constructor, we have added the following lines:

```
jInN.JavaComponent jcp = new jInN.JavaComponent();
this.javaHolder.Children.Add(new JavaWPFControl(jcp, 400, 100));
```

The code first instantiates the proxy for the Java component (JavaComponent), then embeds it inside a special wrapper, com.jnbridge.embedding.JavaWPFControl, which inherits from System.Windows.Forms.Integration.WindowsFormsHost, and which allows the Java component to be used wherever a WPF Control is expected. Note that the Java component's size (400 pixels wide and 100 pixels high) must be supplied to the JavaControl constructor. Finally, the wrapped Java component is added to the javaHolder panel.

We have also created a callback class that implements java.awt.event.ActionListener, and which will be executed whenever the Java component's button is clicked:

```
[AsyncCallback("java.awt.event.ActionListener")]
      public class DotNetActionListener : ActionListener
{
      private java.awt.TextField javaTextBox;
      private TextBox wpfTextBox;

      public DotNetActionListener(java.awt.TextField jTB, TextBox dNTB)
      {
            javaTextBox = jTB;
            wpfTextBox = dNTB;
      }

      private delegate void setTextBox(string s);
      private void setTextBoxProc(string s)
      {
            wpfTextBox.Text = s;
      }
```

```
        public void actionPerformed(ActionEvent p1)
        {
              string s = javaTextBox.getText();
              wpfTextBox.Dispatcher.Invoke(
                    new setTextBox(setTextBoxProc), new object[] { s });
        }
    }
```

The callback code extracts the text from the Java TextField, and writes it to the .NET TextBox. Note that the assignment to wpfTextBox must be done inside a call to Invoke() so that it is executed within wpfTextBox's owning thread. This is important, and an exception will result if the assignment is made directly within the listener and the listener thread. The callback is instantiated and registered with the Java component as an ActionListener by the following line in the form's constructor:

```
jcp.addActionListener(
     new DotNetActionListener(jcp.javaTextBox, wpfTextBox));
```

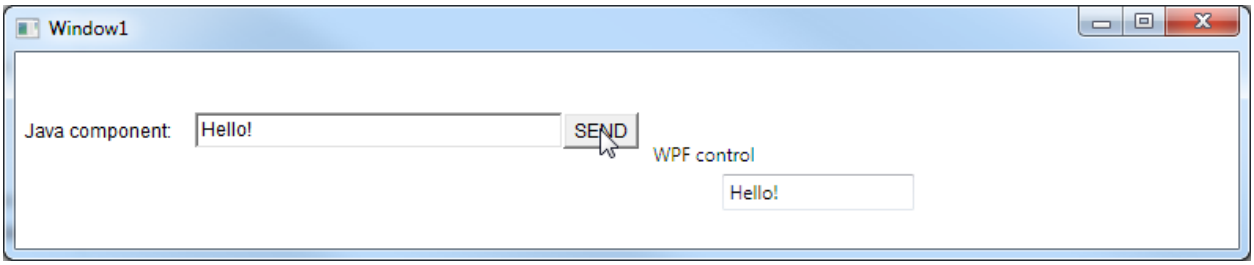
Configuring and running the application

The project is constructed in the same way as other .NET-to-Java interop projects. The communications mechanism must be shared memory:

```
  <jnbridge>
    <dotNetToJavaConfig scheme="sharedmem"
     jvm="C:/Program Files (x86)/Java/jre6/bin/client/jvm.dll"
     jnbcore="C:/Program Files (x86)/JNBridge/JNBridgePro v10.1/jnbcore/jnbcore.jar"
     bcel="C:/Program Files (x86)/JNBridge/JNBridgePro v10.1/jnbcore/bcel-5.1-
jnbridge.jar"
     classpath="../../../Java"
    />
  </jnbridge>
```

The one other special consideration is that copies of the files jawt.dll and jnbjavaentry.dll must be placed in the same folder as the .NET .exe file. If either is not there, an exception will be thrown. A copy of jawt.dll can be found in the jre\bin folder (if you are using a JDK) or in the bin folder (if you are using a JRE).

When the .NET application is run, the Java component appears embedded in the WinForms application, and when text is entered in the Java component's text field and the SEND button is clicked, the text will appear in the .NET form's text box, illustrating how the .NET and Java GUI elements communicate.



Summary

The above example shows how simple it is to embed a Java GUI component inside a WPF application. This embedding can be accomplished in four steps:

- Proxy the Java component and the supporting classes
- Create a panel of the proper size in the WPF application.
- Write code to wrap the Java component's proxy in the special JavaWPFControl wrapper class, and add that JavaWPFControl object to the WPF panel.
- Create .NET classes to implement any ActionListeners, and register them with the Java component's proxy.