



<http://www.progressivebusinesstechnologytraining.com/>

Writing Your own Twitter Application Using Java, Swing, and Twitter4j

By: Cesar Otero

Introduction

This article is brief introduction to using the twitter4j API, which is used for creating your own Twitter applications in Java. This API gives access to the Twitter infrastructure, and allows you to do such tasks as updating your status, getting a list of your followers, and many other useful tasks all from your Java program.

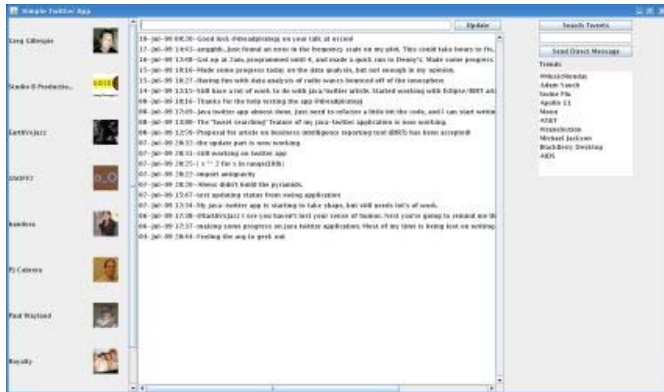
A good understanding of Java and Swing gui's is needed. Our goal is to build a basic desktop application using Swing as our view technology, that will:

- * Login through a pop-up
- * Display the users followers
- * Display the users updates
- * Allow the user to update his status on Twitter (send a tweet)
- * Search for Tweets
- * Send direct tweets

This will serve as a skeleton for your own applications. Use your imagination, and feel free to embellish on the example we create here. Although you could use any view technology, such as JSP, I've decided to use Swing in order to avoid entering into too many additional topics.

Currently, there are three Java Twitter API's; Twitter4J by Yusuke Yamamoto, Java-Twitter by DeWitt Clinton, and JTwitter by Daniel Winterstein. At the time of this writing Twitter4J is at version 2.0.8, Java-Twitter is at version 0.9, and JTwitter is at version 1.2. We'll be using Twitter4J since it's the most mature of the three API's.

Before getting started, you will need a Twitter account; this takes all of 3 minutes to complete at most. You will also need to understand the basics. What a tweet is, following others, and sending direct tweets. And off we go!



The Twitter4J API

Download Twitter4J from <http://repo1.maven.org/maven2/net/homeip/yusuke/twitter4j/>. You can also download the Java docs from here which are enormously helpful. As always, make sure the .jar file is on your classpath. Here is a sample session utilizing Twitter4J:

```
import twitter4j.*;

public class Test {
    public static void main(String[] args) throws TwitterException {
        Twitter twitter = new Twitter("username", "password"); // login to service
        twitter.updateStatus("Tweeting!"); // update your status
    }
}
```

In the preceding code, we first need to import the twitter4j classes. Inside of the main method, an instance of the Twitter class is created. The user name and password are passed in as strings to the constructor. If the login is successful, the users status is updated with the call to the method

updateStatus

. You can verify that the update worked by logging in to your Twitter account.

Since a login is obviously required to do anything, our application should first display a dialog which requires the necessary data to login. This dialog will have two

JLabels

, a

JTextField

, a

JPasswordField

, a

confirmation JButton

, and a

cancel JButton

in case the user changes their mind. Just to keep it clean, we'll create a class for our login dialog.

```
import javax.swing.*;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.*;

class LoginDialog extends JDialog implements ActionListener {
    JLabel labelName;
    JLabel labelPass;
    JTextField textName;
    JPasswordField passField;
    JButton okButton;
    JButton cancelButton;

    JDialog dialog;

    public LoginDialog() {
        JPanel panelOne = new JPanel();
        labelName = new JLabel("Name");
        textName = new JTextField(15);
        panelOne.add(labelName);
        panelOne.add(textName);

        JPanel panelTwo = new JPanel();
        labelPass = new JLabel("Password");
        passField = new JPasswordField(15);
        panelTwo.add(labelPass);
        panelTwo.add(passField);

        JPanel panelThree = new JPanel();
        okButton = new JButton("OK");
        cancelButton = new JButton("Cancel");
        okButton.addActionListener(this);
        cancelButton.addActionListener(this);
        panelThree.add(okButton);
```

```

panelThree.add(cancelButton);

dialog = new JDialog();
dialog.setResizable(false);
dialog.getContentPane().add(panelOne);
dialog.getContentPane().add(panelTwo);
dialog.getContentPane().add(panelThree);
dialog.setTitle("Login in to Twitter");
dialog.getContentPane().setLayout(new FlowLayout());
dialog.setSize(350, 150);
dialog.setLocationRelativeTo(null); // place in center of screen
dialog.setModal(true);
dialog.setVisible(true);

}

public void actionPerformed(ActionEvent e) {
    if (e.getSource() == okButton) {
        dialog.dispose();
    } else if (e.getSource() == cancelButton) {
        System.exit(0);
    }
}

public String getUsername() {
    return textName.getText();
}

public String getPassword() {
    return String.valueOf(passField.getPassword());
}
}

```

Up to now, this is all pretty straight forward, old fashion Java code. Here we are using a FlowLayout, and we're inheriting from

JDialog

. Our implemented action is to either close the dialog window if the ok button is pressed, or to exit the program all together. There are two getter methods;

getUsername()

and

getPassword()

. If the ok button is pressed, we will pass on the information through the get methods. The actual login will occur in the main program which will hold our Twitter instance, and pass it around as needed. If you want to test this you can add a main to this class such as this:

```
public static void main(String[] args) throws TwitterException{
    LoginDialog login = new LoginDialog();
    String userName = login.getUserName();
    String password = login.getPassword();

    try{
        Twitter twitter = new Twitter(userName, password);
        twitter.verifyCredentials();
        JOptionPane.showMessageDialog(login, "Login successful!");
    } catch(TwitterException e){
        JOptionPane.showMessageDialog(login, "Unable to login");
    }
}
```

Creating a new Twitter instance does not throw a

TwitterException

, but calling the method

verifyCredentials()

does. Upon a successful login, you should see the message in figure 2.

