MÓDULO 3: Desarrollo de aplicaciones basadas en Servicios

Cliente de Twitter

Alejandro Fernández-Montes González
afdez@us.es

Máster en Desarrollo de Aplicaciones para Internet y Dispositivos Móviles

Contenido

Ejercicio Entregable: Creación de un cliente de escritorio para Twitter

Profesor: Alejandro Fernández-Montes González

Fecha recomendada de entrega: 12/abril/2015

Fecha límite de entrega: 15/abril/2015

Enunciado:

Crear un cliente gráfico de Twitter que al menos consulte el "Friends_timeline" y tenga la posibilidad de enviar un tweet.

Extra: Ampliación de funcionalidades

Agregue nuevas funcionalidades a su cliente de Twitter. Elija aquellas que considere más interesantes. He aquí algunos ejemplos:

- Mostrar en la barra de estado información sobre lo que se muestra en el cliente. Por ejemplo: número de tweets descargados, tiempo desde la última actualización, tiempo para la próxima actualización, etc.
- Permita al usuario elegir la periodicidad entre las actualizaciones del timeline, y cuantos elementos quiere cargar.
- Muestre los TrendingTopics de su localización. Para ello tome el woeid de España que es: 23424950
  https://dev.twitter.com/docs/api/1/get/trends/%3Awoeid

Material a entregar:

Proyecto con el cliente de twitter con las funcionalidades básicas listo para ser lanzado, y con funcionalidades extras para obtener una mayor calificación.

Cómo entregar: La solución debe entregarse por Blackboard a través de la actividad que podrá encontrar en la carpeta "Contenidos/Modulo 3/Servicios web"

Tiempo estimado: **7 horas.**

En este ejercicio vamos a crear un cliente gráfico basado en *REST* para Twitter, que mostrará el *Timeline* de las actualizaciones de estado y para ver y actualizar tu propio estado. La aplicación usa *Swing* para presentar la interfaz gráfica por lo que veremos también como crear una GUI simple.

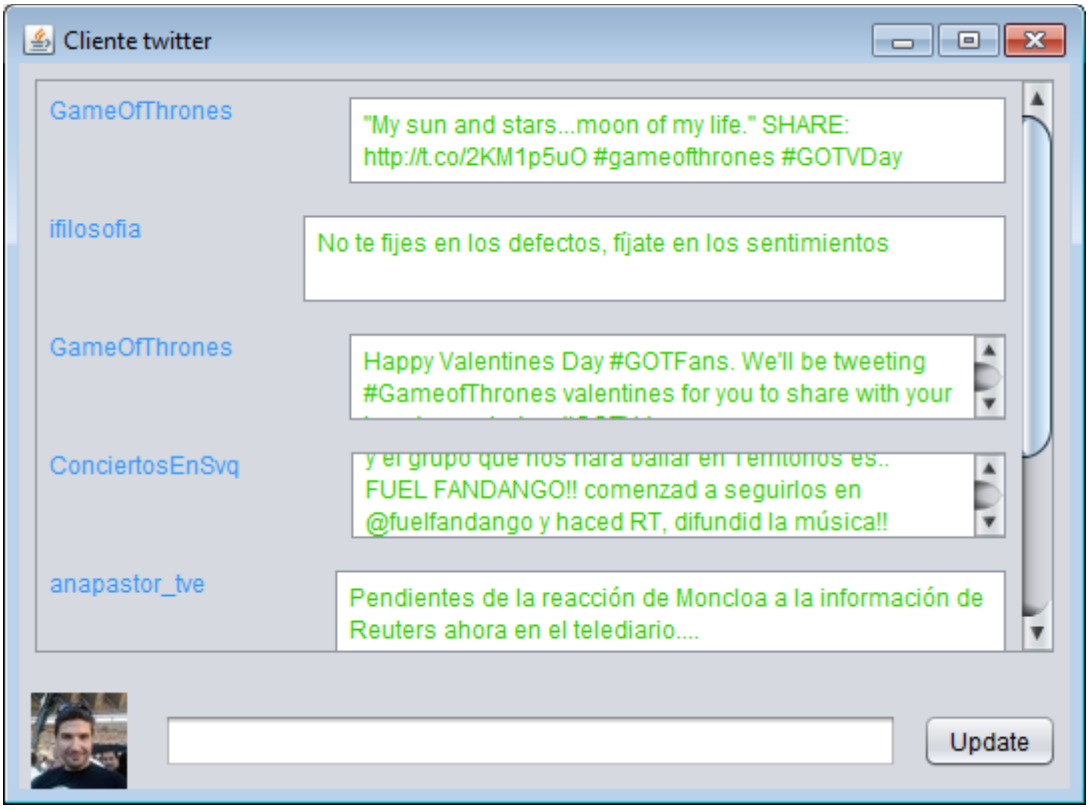


Figura 1: Aspecto que presentará el cliente de twitter.

Si no tienes una cuenta de Twitter, date de alta en https://twitter.com/signup.

Diseñando la ventana. JFrame.

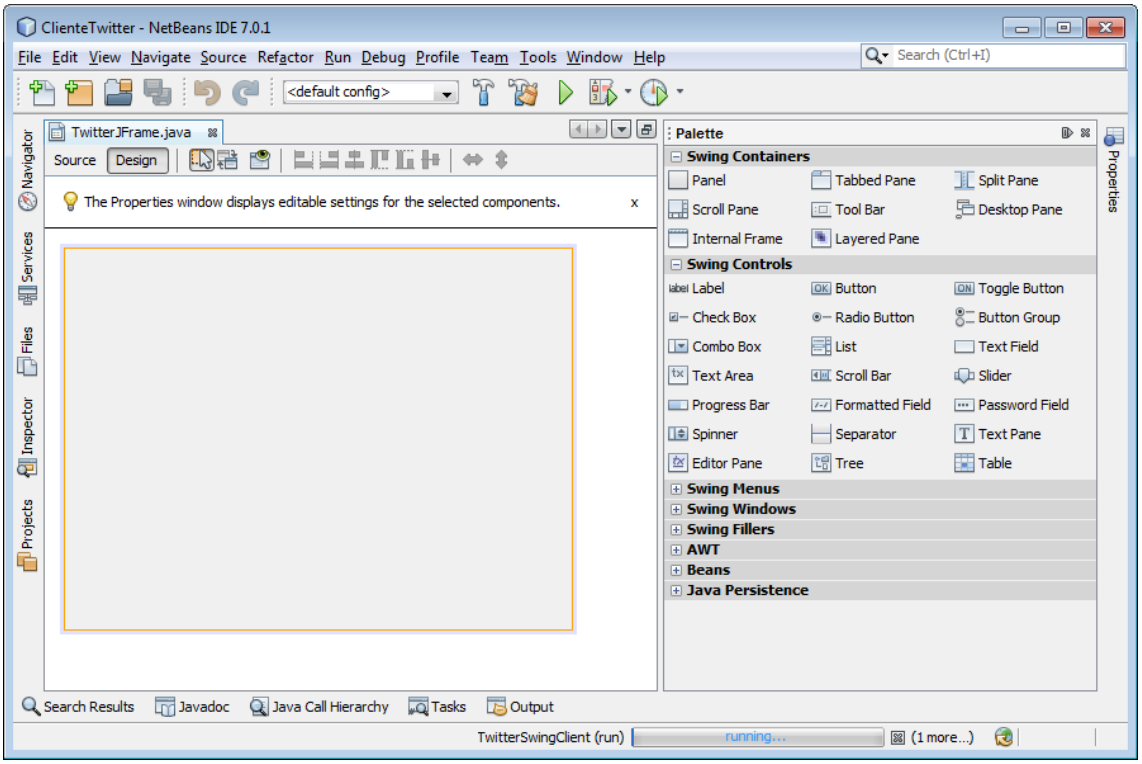
En este primer paso vamos a crear los elementos gráficos que mostrarán el *timeline* de tus amigos de Twitter, tu avatar de usuario, y el espacio donde lees y publicas tu estado. Todos estos elementos se encontraran dentro de un `JFrame`. Por supuesto que no es obligatorio que organices los contenidos como se propone en este ejercicio y también puedes usar las fuentes y estilos que desees.

Abra el proyecto Java que se le suministra `ClienteTwitterAlumnos`

A continuación vamos a crear un nuevo `JFrame`. Para ello diríjase a New → JFrame Form (New → Other → Swing GUI Forms → JFrame Form). Aparecerá el asistente para la creación de un nuevo `JFrame`.

Nombre a la clase como `TwitterJFrame`, indique se cree en el paquete `es.us.mwm.twitter.gui` y pulse "Finish".

Se abrirá la vista de diseño para la clase `TwitterJFrame`. A la derecha puedes encontrar todos los componentes *Swing* que puedes arrastrar sobre la ventana `JFrame`.



Arrastra un nuevo botón sobre el `JFrame` (se encuentra en la sección *Swing Controls*). Suéltalo en la esquina inferior derecha. Si te fijas el botón muestra el texto *jButton1*, que corresponde con el nombre del objeto `JButton` declarado en la clase.

Haz click derecho sobre el botón, y selecciona "Edit text". Cambia el nombre por "Tweet". Este botón servirá para publicar un nuevo *tweet*.

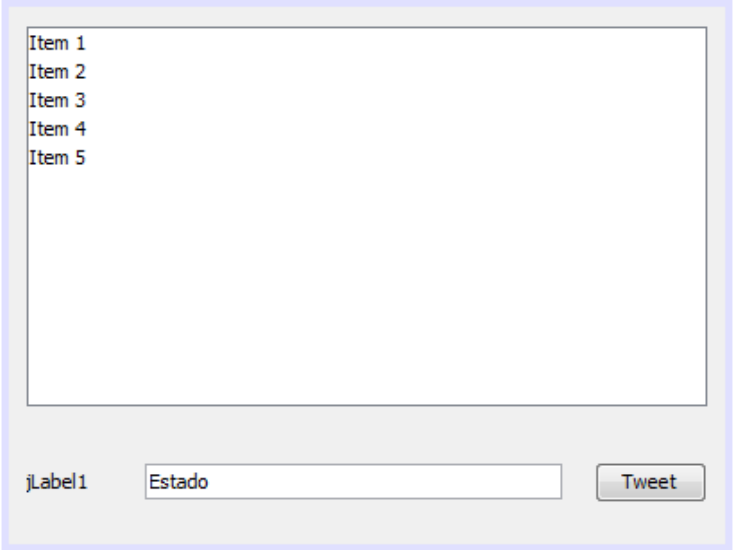
Arrastra un nuevo elemento de tipo `Label` (`jLabel1`) a la esquina inferior derecha del `JFrame`. Cambia el nombre por "Avatar". Tu avatar de usuario será mostrado en esta área.

Arrastra un nuevo elemento de tipo Text Field (JTextField1) entre el `Label` y el botón. Cambia su texto a "Estado". Haz click en el extremo derecho del campo de texto para cambiar su tamaño y que de esta manera ocupe todo el espacio inferior de la ventana. Sírvase de las guías que aparecen para alinearlo con los elementos anteriores.

Haz click derecho sobre el `jLabel1`, y cambia las propiedades. Establece la propiedad "*labelFor*" para que apunte al `jTextField1`. Esta acción mejor la accesibilidad. Cambia también las propiedades "*Maximum Size*", "*Minimun Size*" y "*Preferred Size*" a [48x48], para que tenga el mismo tamaño que los avatares de Twitter.

Por último, arrastra un *Scroll Pane* en la parte superior del `JFrame`. Cambia su tamaño para ocupar todo el espacio superior de la ventana y arrastra un *List* sobre el *Scroll Pane*. Aparecerá una lista de elementos. Guarda el fichero.

El `JFrame` tendrá ahora un aspecto similar al que se muestra en la siguiente figura:



Ahora ya tienes los componentes básicos para el cliente de Twitter. A continuación vamos a añadir operaciones de Twitter.

Añadiendo el *timeline* de los contactos

Para recoger las actualizaciones de estado de los contactos vamos a usar el método `TwitterClient.getFriendsTimeline()`. Para ello vamos a sobrescribir el método `run` de `TwitterJFrame` incluyendo llamadas a `getFriendsTimeline`.

Necesitamos obtener un objeto de la clase `TwitterClient` y guardarlo como atributo de nuestro `TwitterJFrame`. Para ello, cree el objeto en el constructor y declare un atributo de nombre `client` donde guardarlo.

Para que las llamadas se realicen periódicamente, vamos a usar un objeto de tipo `TimerTask`. Para ello diríjase de nuevo al constructor de `TwitterJFrame` y cree un nuevo objeto de tipo `java.util.Timer` y use el método `scheduleAtFixedRate`. Este método recibe un objeto de tipo `TimerTask` que vamos a definir en la propia llamada. Además recibe dos parámetros de tipo `long`. El primero marca una espera (`delay`) antes de que comience la primera ejecución de la tarea, y el segundo marca el tiempo entre dos ejecuciones (`period`).

Use este código para la creación de esta automatización para actualizar el *timeline* periódicamente:

```
Timer timerTwitter = new Timer("TwitterTimer", false);
```

```
timerTwitter.scheduleAtFixedRate(new TimerTask() {
  @Override
  public void run() {
   throw new UnsupportedOperationException("Operacion no soportada.");
  }
}, 5000, 30000);
```

Observe que la clase de tipo `TimerTask` se está definiendo *en línea*. Esto es, incrustada en la llamada a un método. Observe también que se ha definido el método `run()` que sobrescribe (`@Override`). Esto es así ya que la clase `TimerTask` de Java es abstracta, así como su método `run`, por lo que hay que darle cuerpo al mismo.

Observe también que se ha establecido 5 segundos para la primera ejecución, y una espera de 30 segundos para cada actualización. Los primeros 5 segundos nos sirven para asegurarnos de que da tiempo de completar el proceso de logeo y autenticación. Puede probar a reducir estos tiempos para incrementar la frecuencia de actualización.

Antes de realizar las llamadas a `getFriendsTimeline` dentro del método `run` recién definido debemos crear la estructura de datos para que las actualizaciones de estado sean mostradas en el `JScrollPane` que creamos para tal efecto. Para ello declare un objeto de tipo `DefaultListModel` como atributo de la clase principal `TwitterJFrame` que representa a una lista que puede ser mostrada en el `JScrollPane`.

```
private DefaultListModel statusesListModel = new DefaultListModel();
```

Ahora vamos a darle cuerpo al método `run()` con el siguiente código:

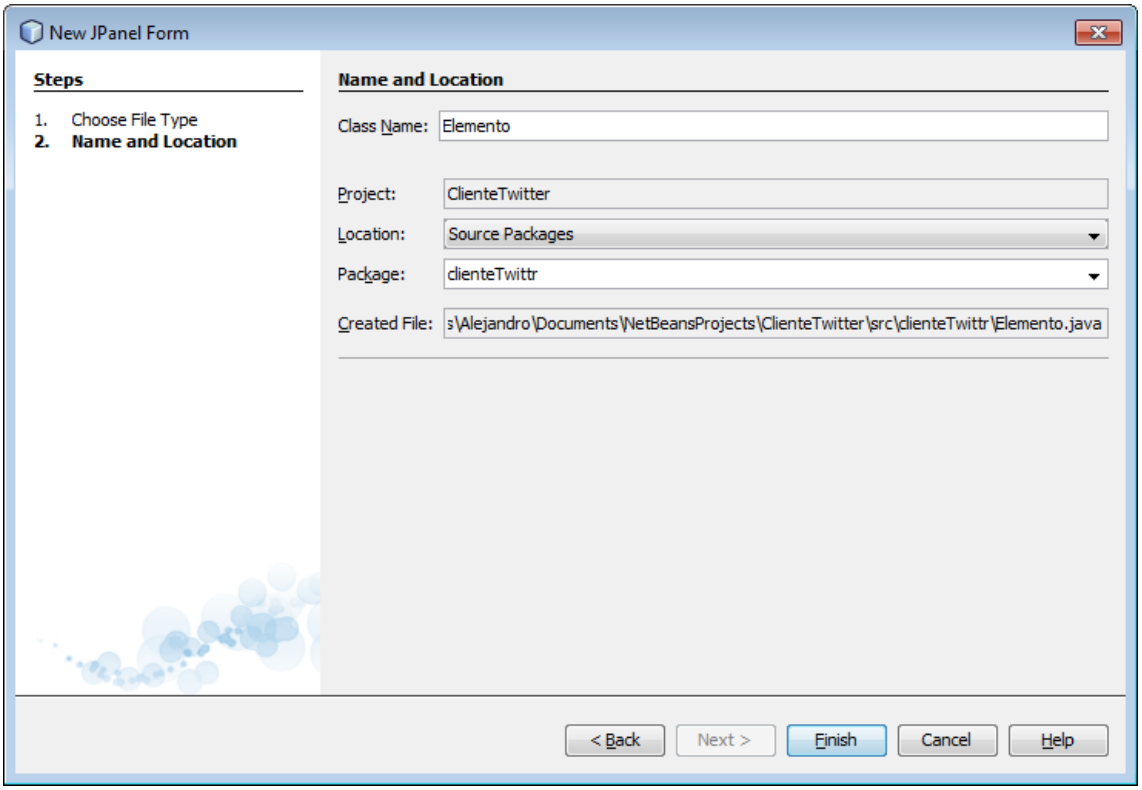
```
@Override
public void run() {
  System.out.println("Timer Task se está ejecutando");
  try {
    Response response = client.getFriendsTimeline();
    // Limpia la lista, de manera que los elementos anteriores no se
    //replican
    statusesListModel.clear();
List<Status> statuses =
                  response.readEntity(new GenericType<List<Status>>() {
                     });
    //Añade los elementos de tipo Status a la lista
    for (final Status st : statuses) {
      SwingUtilities.invokeLater(new Runnable() {
      public void run() {
        statusesListModel.addElement(st);
        }
      });
    }
    } catch (RuntimeException ex) {
      System.out.println("Excepcion llamando a getFriendsTimeline.
                          Detalles: " + ex.getResponse());
  }
}
```

En este método se llama a `getFriendsTimeline`, para recoger las últimas actualizaciones de estado. Después se itera sobre la respuesta para añadir elementos a la lista de estados. Nótese que estos elementos se agregan a través de una utilidad llamada `invokeLater`. Esto es así porque las actualizaciones de la GUI han de hacerse desde el hilo correspondiente, y para ello nos ayudamos de la clase `SwingUtilities`.

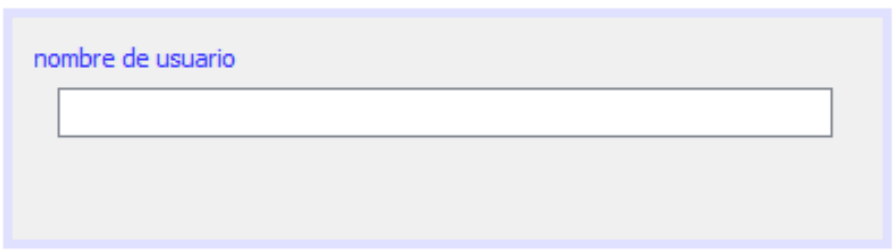
Creando un componente para renderizar la lista

Finalmente vamos a crear un componente para visualizar la lista de actualizaciones de tus contactos de Twitter. Para ello siga los siguientes pasos:

Haga click derecho en el proyecto y agregue un nuevo "JPanel Form". Llámelo `Elemento` y agréguelo en el paquete `es.us.mwm.twitter.gui`



Vaya a la vista de diseño y agregue un nuevo `JLabel` y un nuevo `JTextPane`. Intente colocar estos elementos de una manera similar a la que se muestra en la siguiente figura:



Haga click derecho sobre la etiqueta y acceda a sus propiedades para que la propiedad "labelFor" apunte al `JTextPane`. Además puede cambiar el color de la fuente estableciendo la propiedad "foreground".

Puede además abrir las propiedades del `JTextPane`, y experimentar con su apariencia.

Vaya a la vista de fuente, he indique que la clase `Elemento` implementa `ListCellRenderer<Status>`. La declaración de la clase debe ser por lo tanto:

```
public class Elemento extends javax.swing.JPanel
                      implements ListCellRenderer<Status>
```

Nótese que ahora necesitará definir al menos el método `getListCellRendererComponent` para implementar esa interfaz. Complete el método con este código:

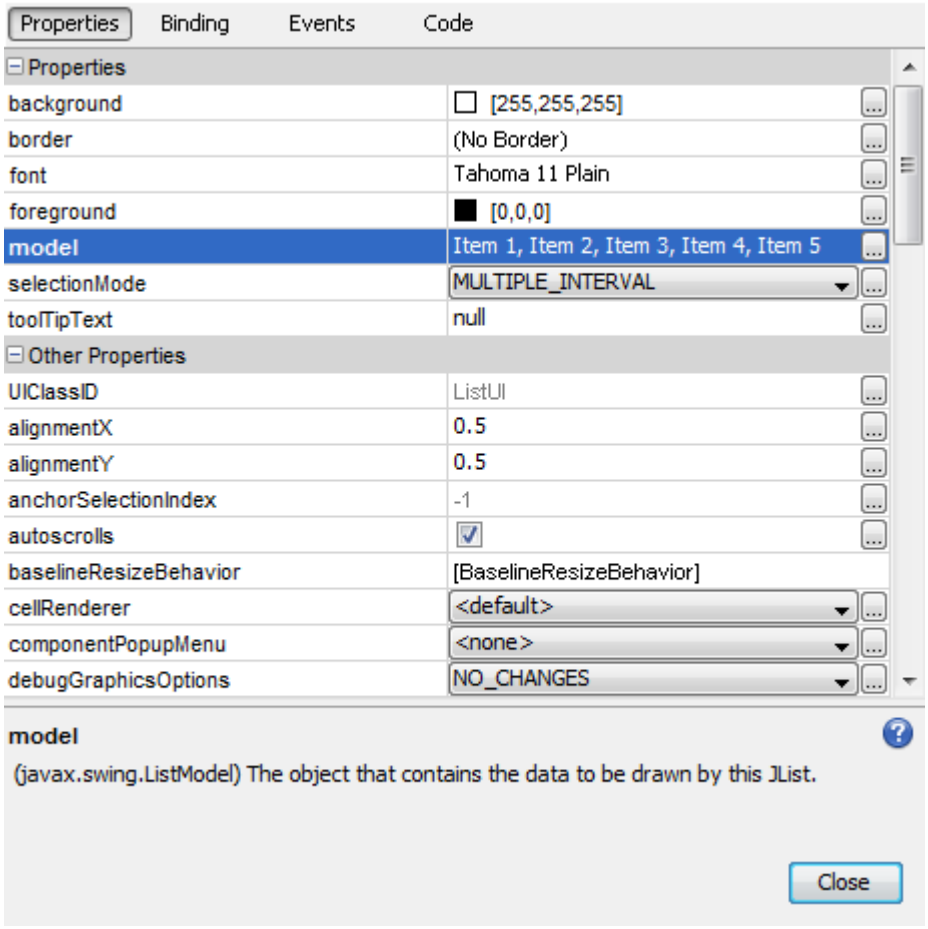
```
@Override
public Component getListCellRendererComponent(JList<? extends Status>
list, Status value, int index, boolean sel, boolean focus) {

   jTextPane1.setText(value.getText());
   jLabel1.setText("<html>" + value.getUser().getName() + "</html>");
   return this;
}
```

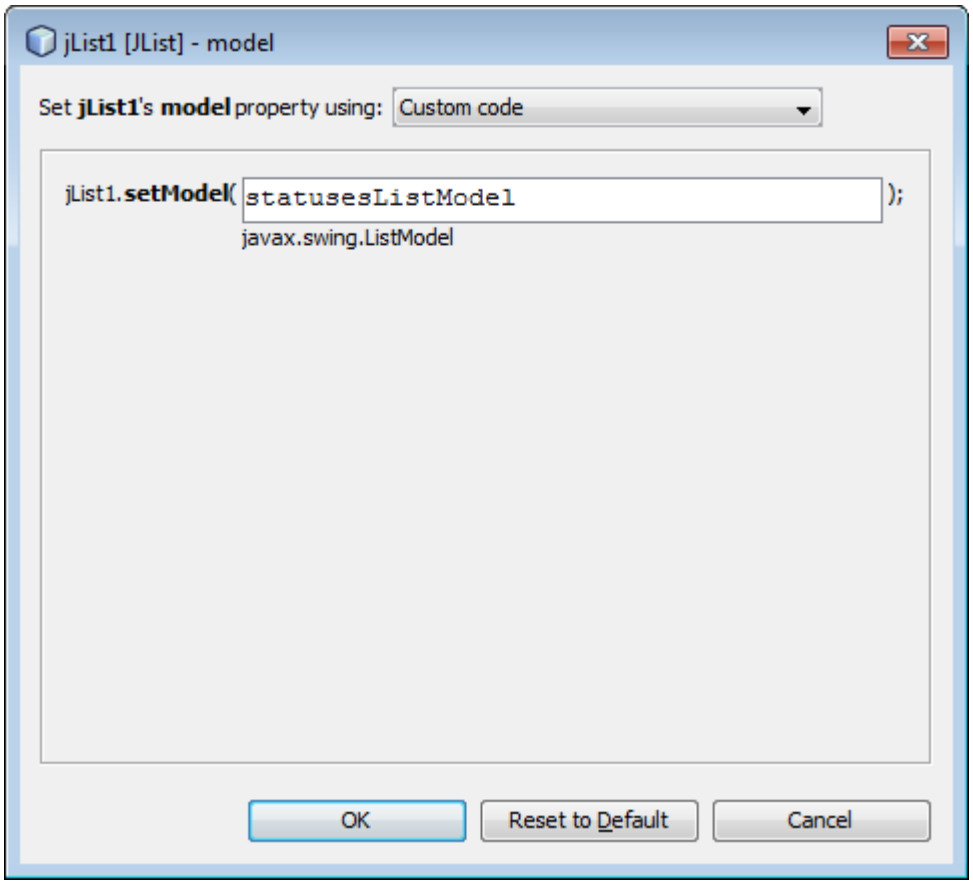
Este código remplaza los valores de la etiqueta y el texto del `jTextPane` con lo que se obtiene a través del objeto de tipo `Status`.

Ahora vamos a modificar la clase `TwitterJFrame` para que use estos componentes.

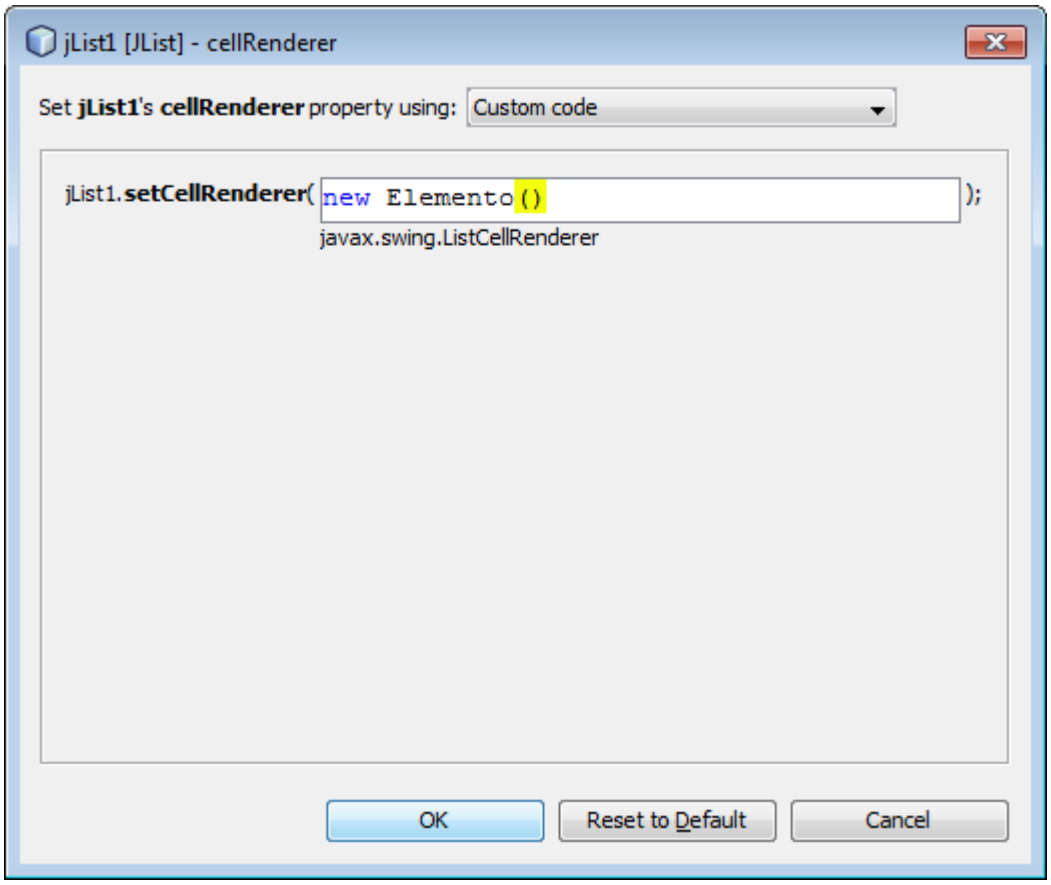
Abra la clase `TwitterJFrame` y muestre la vista de diseño. Haga click derecho sobre la lista del `ScrollPane` y abra sus propiedades.



Pulse sobre el botón ... que aparece a la derecha de la propiedad “model”. Seleccione en el desplegable la opción “CustomModel” y teclee el nombre del atributo de tipo `DefaultListModel` que nombramos como `statusesListModel`.



Modifique ahora la propiedad “cellRenderer” para que se use la clase `Elemento` que acabamos de crear.



Guarde todos los archivos.

Recoger las claves *OAuth* desde Twitter.

Para que la aplicación Java pueda acceder a los servicios de Twitter, necesitamos recoger las claves CUSTOMERKEY(API_KEY) y CUSTOMERSECRET(API_SECRET), además de la cadena de verificación desde Twitter. Twitter usa autorización *OAuth*, que requiere de estas claves. Sin embargo *OAuth* está pensado para ser utilizado desde una aplicación web desde un servidor web. Para conseguir estas claves vamos a seguir el procedimiento para registrar una aplicación web.

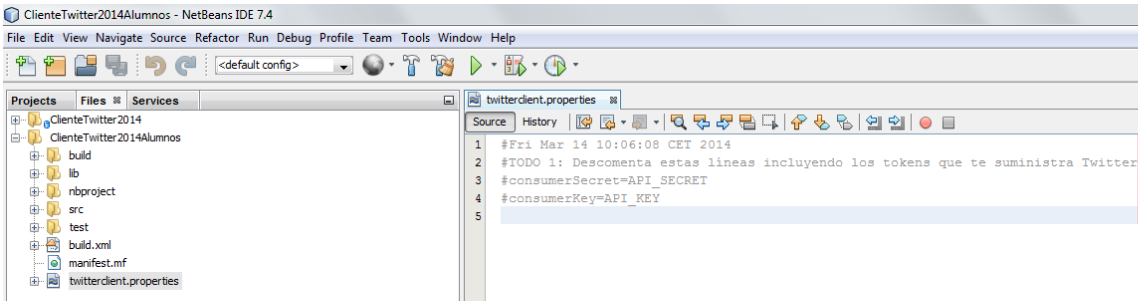
Abre el navegador, y dirígete a la página de aplicaciones de Twitter https://dev.twitter.com/apps y haz click en "Create a new Application". Necesitarás loguearte en Twitter para completar esta operación.

Escribe el nombre de la aplicación que desees, por ejemplo: Cliente de prueba de {tunombre}. Escribe también una descripción para la aplicación (es obligatorio) y una url en el campo "Website" por ejemplo: http://www.us.es. Además tienes que darle permisos de "Read, write, and direct messages", desde la pestaña "Settings".

Acepta las condiciones para el desarrollo de aplicaciones para Twitter y rellena los caracteres que aparecen en el "captcha"

Ahora se te presenta la página de información sobre la aplicación que acabas de crear y aparecen las claves que debemos ingresar en nuestra aplicación.

Copia y pega las claves CONSUMER_KEY y CONSUMER_SECRET en el fichero twitter.properties que encontrarás en la raíz de tu proyecto.



Guarda los cambios.

Ejecutando el proyecto.

Haz click derecho sobre el proyecto y selecciona Run.

En la primera ejecución deberá navegar hacia la URL que se le muestra por consola para poder autorizar la aplicación. Una vez autorizada, Twitter te mostrar un PIN de autorización, copia y pégalo en la consola de NetBeans y pulsa Enter.

Mostrando el estado del usuario

En esta sección vamos a crear un nuevo método y vamos a añadir la operación de Twitter *getUserTimeline*. Esta operación recupera tu avatar de usuario y tu estado actual, por lo que crearemos el código necesario para mostrar tu avatar y tu estado en los objetos `jLabel1` y `jTextField` respectivamente. Finalmente añadiremos una línea al constructor de `JFrame` para inicializar el método.

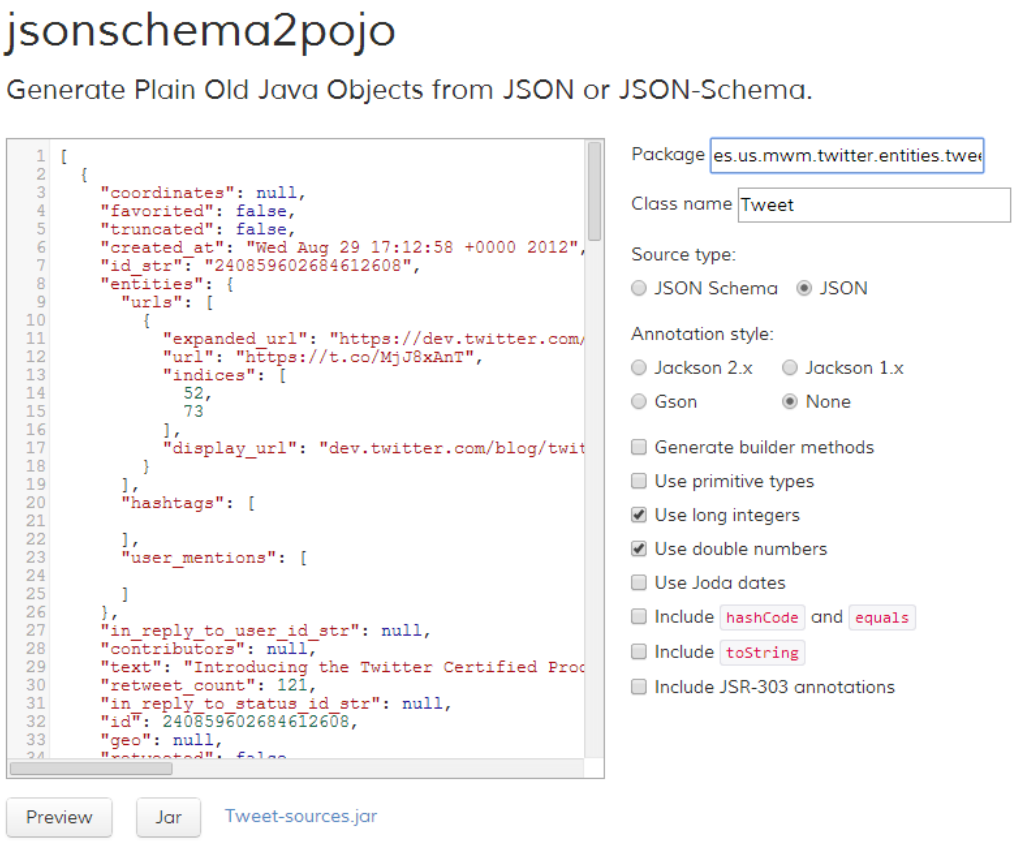
Añade la funcionalidad que se describe en la clase `TwitterClient`. Cree un método `getUserTimeline()` similar al método `getFriendsTimeline()`.

Este nuevo método tiene que hacer llamadas al servicio REST descrito en:

https://dev.twitter.com/docs/api/1.1/get/statuses/user_timeline

Necesitará crear POJOs para encapsular la respuesta. Para ello puede usar la siguiente herramienta web:

http://www.jsonschema2pojo.org/



A continuación vamos a crear un método en la clase `TwitterJFrame` llamado `initUserInfo()` con el siguiente código:

```
private void initUserInfo() throws MalformedURLException  {
 // Crea una instancia de la clase interna que ofrece el servicio
 Response responseUserTimeline = client.getUserTimeline();

 List<Tweet> tweets =
        responseUserTimeline.readEntity(new  GenericType<List<Tweet>>()
            {});

 if (!tweets.isEmpty()) {
   Tweet ultimoTweet = tweets.get(0);
   jTextField1.setText(ultimoTweet.getText());
   String iconSrc = ultimoTweet.getUser().getProfile_image_url();
   URL iconUrl = new URL(iconSrc);
   ImageIcon icon =
             new ImageIcon(iconUrl, ultimoTweet.getUser().getName());
   jLabel1.setIcon(icon);

   /*System.out.println("Mis Tweets:");
   for (Tweet tweet : tweets) {
      System.out.println(tweet.getText());
   }*/
 }
}
```

Observe que este método recoge el último estado publicado y lo muestra en el campo de texto y además recoge el avatar del usuario para mostrarlo en el `jLabel`.

Finalmente vamos a incluir la llamada a este método en el constructor de la clase `TwitterJFrame`. Para ello complete el constructor con este código:

```
public TwitterJFrame() {
    initComponents();

    try {
        initUserInfo();
    } catch (MalformedURLException ex) {
Logger.getLogger(TwitterJFrame.class.getName()).log(Level.SEVERE,
                                                    null, ex);
    }
 }
```

Usando otros servicios.

Siga la metodología anterior para añadir la funcionalidad de *twitear* un nuevo post.