

Automatización de scripting con Sikuli

# AYUDA VISUAL



Fred Goldstein, Fotolia

Usamos capturas de pantalla para generar scripts intuitiva y automáticamente con el entorno de scripting Sikuli. **POR DMITRI POPOV**

**C**on el uso de las capacidades de shell scripting de Linux podemos automatizar virtualmente cualquier tarea de nuestro sistema. Incluso si no somos un gurú de la programación, podemos escribir scripts que pueden hacerse cargo de las tareas

mundanas, desde montar unidades compartidas remotas a realizar copias de seguridad.

A pesar de toda su potencia, el shell scripting tiene una seria limitación: los scripts son buenos únicamente para controlar y automatizar herramientas

en línea de comandos y aplicaciones gráficas que soporten argumentos en línea de comandos. Por tanto, si queremos automatizar entornos de escritorio gráficos como Gnome o KDE, o automatizar aplicaciones basadas en interfaz gráfica, los scripts tradicionales no nos servirán.

Presentamos Sikuli [1], un entorno único de scripting que nos permite automatizar aplicaciones basadas en interfaz gráfica con gran facilidad. Sikuli no confía en ninguna interfaz de programación de aplicaciones y usa simples capturas de pantalla de elementos de interfaz gráfica de usuario como sus bloques constructivos. Básicamente, en lugar de describir dónde pulsamos o qué elemento del menú elegimos, simplemente alimentamos Sikuli con una captura de pantalla del área específica en el script.

Sikuli analiza el patrón de la imagen, encuentra el elemento apropiado en la interfaz gráfica, y ejecuta la acción especificada sobre ella. No hace falta decirlo, esto simplifica tremendamente el proceso de escribir un script. De hecho, el método de Sikuli es tan intuitivo

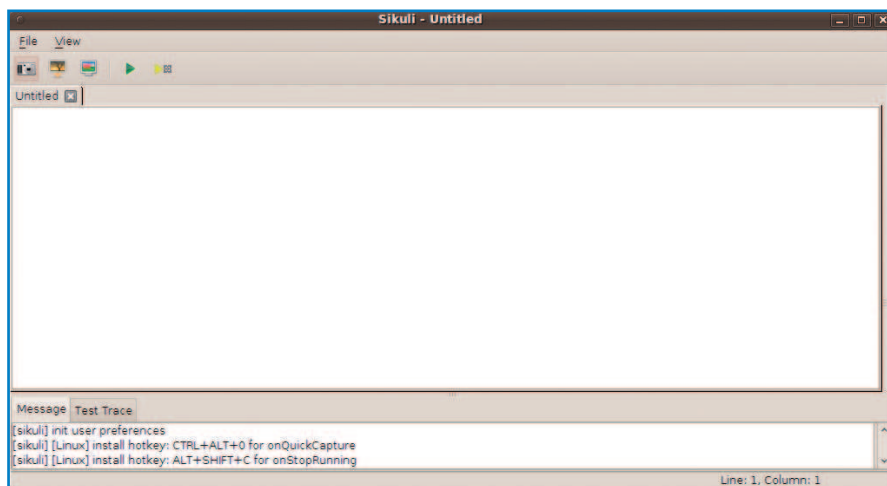


Figura 1: La IDE de Sikuli es minimalista y sólo tiene los botones más útiles en la barra de herramientas.

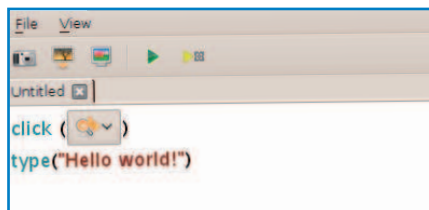


Figura 2: Script "Hello world!".

tivo que podemos comenzar a escribir scripts en cuestión de minutos, incluso si no tenemos ninguna experiencia como programador.

### Comenzar con Sikuli

Sikuli está escrito en Java, por lo que la primera tarea será configurar el Java Runtime Environment de nuestro sistema. Adicionalmente, tendremos que instalar un par de paquetes requeridos. En Ubuntu, esto puede hacerse ejecutando el siguiente comando en un terminal:

```
sudo apt-get install \
libcxxtools6 libcxxtools-dev \
libhighgui libhighgui-dev \
libcvi1
```

Una vez que todas las piezas están en su lugar, tomamos la última versión de Sikuli y desempaquetamos el archivo descargado en el directorio de nuestra elección (por ejemplo, nuestro directorio de usuario). En el terminal, nos ubicamos en el directorio resultante *Sikuli-IDE* y lanzamos la IDE de Sikuli con el

script *sikuli-ide.sh*. De forma alternativa, podemos iniciarlo haciendo doble clic en el archivo *sikuli-ide.jar*, suponiendo que el tipo de archivo *.jar* esté asociado con el motor Java Runtime.

La interfaz de Sikuli es muy fácil de entender. La barra de tareas principal ofrece un útil conjunto de botones que proporcionan acceso rápido a todas las funciones esenciales de Sikuli. Los tres botones más importantes son *Capture*, *Load* y *Run*. El botón *Capture* nos permite tomar una captura de pantalla de un elemento deseado o área de la interfaz gráfica, mientras que el botón *Load* nos permite insertar una captura de pantalla existente en el script actual. Como habrá imaginado, podemos usar el botón *Run* para ejecutar el script actualmente abierto. Gracias a la ayuda de las pestañas, podemos usar el IDE de Sikuli para abrir y administrar múltiples scripts al mismo tiempo.

Al trabajar con Sikuli deberíamos tener algunas cosas en mente. Por un lado, no se lleva bien con múltiples pantallas. Por tanto, si queremos asegurarnos de que nuestros scripts hacen lo que se supone que tienen que hacer, debemos cerciorarnos de que no tenemos monitores externos conectados a nuestra máquina.

Al principio, la mayor parte de mis scripts se negaban a funcionar correctamente. Resultó que debido a que había habilitado la opción izquierda del ratón, todas las acciones de clic se

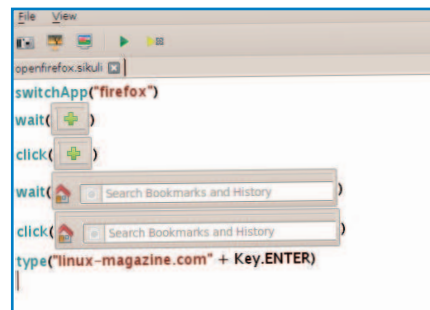


Figura 4: En lugar de acciones "click"...

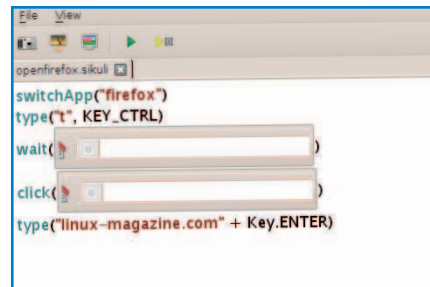


Figura 5: ... usamos una acción "type" con los modificadores de tecla para emular atajos de teclado.

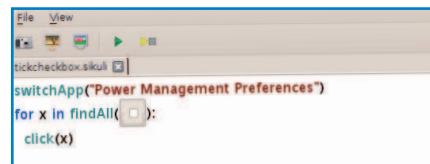


Figura 6: Marcamos todos los checkboxes.

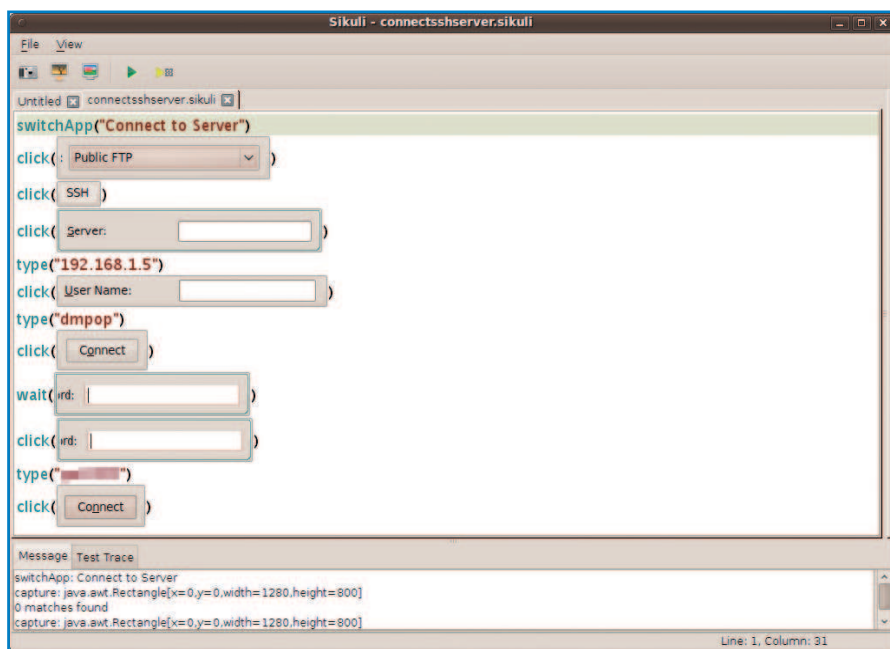


Figura 3: El script automatiza el proceso de montar una unidad compartida por SSH.

interpretaban como clics derechos. Por tanto, para todos los zurdos, la solución es sencilla: usar la acción del clic derecho en su lugar y viceversa.

La versión de Linux de Sikuli aún no soporta atajos de teclado, por lo que tenemos que usar el botón *Capture* para tomar capturas de pantalla. Alternativamente, podríamos usar una utilidad externa como *Shutter* para tomar las capturas de pantalla e importarla a nuestro script usando el botón *Load*. El uso de una herramienta de capturas de pantalla dedicada nos proporciona más control y acelera todo el proceso. Sin embargo, recuerde que todas las capturas de pantalla deben estar en formato PNG.

Para aprender lo básico, comenzaré creando un sencillo script que pulsa en el icono del applet de la barra de escritorio y teclea ¡Hello world! en el campo *Search*. Obviamente, para que funcione este script tenemos que instalar el paquete *deskbar-applet* previamente y añadirlo al panel. Un script Sikuli consiste en una serie de acciones como

*click*, *wait*, *type* y demás. Cada acción puede usar una captura de pantalla que especifica el elemento o área objetivo de la interfaz gráfica de usuario. Por ejemplo, si queremos que el script pulse en un botón específico de la aplicación deseada, añadimos la acción *click* seguida de la captura de pantalla de ese botón. En este caso, el script debería hacer dos cosas: pulsar en el applet de la barra de escritorio en el panel y teclear la cadena *¡Hello world!* en el campo *Search*.

Para conseguir esto tenemos que especificar dos pasos: *click*, con una captura de pantalla del botón de la barra de escritorio, y *type*, con la cadena *¡Hello world!* especificada, como se muestra en la Figura 2. Luego pulsamos el botón *Run* en la barra principal de Sikuli y observaremos cómo se ejecuta mágicamente por el script.

Ahora que he mostrado cómo funciona Sikuli, crearé un script que verdaderamente haga algo útil: por ejemplo, montar una unidad compartida remota vía SSH usando la herramienta *Places | Connect to Server*. El script completo se muestra en la Figura 3, y muchos de sus pasos son obvios.

Para rellenar los campos en el cuadro de diálogo se usaron una serie de acciones *click* y *type* y tecleamos las cadenas especificadas, como la dirección del servidor y el nombre de usuario. Sin embargo, hay dos pasos que requieren un estudio más cercano. Como sugiere el nombre, la acción *switchApp* apunta el script a una aplicación especificada (en este caso, la utilidad *Connect to Server*). Seleccionar un elemento desde la lista desplegable requiere dos acciones *click*: el primero hace clic en la propia lista desplegable (es decir, la lista *Server type*), mientras que el segundo hace clic en el elemento deseado de la lista (es decir, *SSH*). Para capturar el elemento de la lista usando Sikuli tenemos que ajustar el retardo de la captura, por lo que tenemos tiempo suficiente para hacer clic en la lista desplegable antes de que Sikuli entre en modo captura de pantalla. Para ajustar el retardo de la captura, elegimos *File | Preferences* y especificamos el retardo en segundos en el campo *Capture delay*.

Además de la habilidad de teclear cadenas específicas, Sikuli también

puede controlar pulsaciones de teclado y teclas modificadoras, lo que proporciona una manera más eficiente de automatizar aplicaciones. Por ejemplo, eche un vistazo al script de la Figura 4. Vemos que cambia al navegador Firefox, pulsa en el botón *New Tab*, introduce la URL especificada y dispara una pulsación de la tecla Enter. Para hacer esto último, el script usa el argumento *Key.ENTER*.

Debido a que Sikuli soporta modificadores de teclado, podemos reemplazar acciones que pulsen en el botón *New Tab* con el comando *type("t", KEY\_CTRL)* que emula el atajo de teclado Ctrl+T (véase la Figura 5). Otros modificadores de teclado soportados son *KEY\_ALT* (la tecla Alt), *KEY\_META* (la tecla ¡Meta! o Windows) y *KEY\_SHIFT* (la tecla Shift).

Usando la acción *click* también podemos marcar y desmarcar un checkbox en un cuadro de diálogo. Pero, ¿y si el cuadro de diálogo contiene varios checkboxes y queremos marcarlos todos al mismo tiempo? Aquí es donde la acción *findAll* resulta útil. Ésta encuentra todas las ocurrencias de una imagen específica. Por tanto, si usamos una captura de pantalla de un checkbox con la acción *findAll*, encontrará todos los checkboxes en el cuadro de diálogo especificado. A continuación sólo tenemos que agrupar esta acción en un bucle *for...in* para hacer que el script vaya pasando por todos los checkboxes encontrados y los vaya marcando.

El script de la Figura 6 marca todos los checkboxes del cuadro de diálogo de Preferencias de Administración de Energía. Si alguna vez ha intentado trabajar con Python, el código de este script le será familiar. No es coincidencia, ya que Sikuli usa Jython como base de su scripting. Jython es una implementación del lenguaje de programación Python escrito en Java, lo que explica por qué los scripts de Sikuli se parecen sospechosamente a los scripts habituales de Python. Aunque trabajar con Sikuli no requiere ninguna habilidad de programación, algo de conocimiento práctico de Python puede ayudarnos a conseguir usos avanzados con Sikuli.

A pesar de que Sikuli usualmente hace un buen trabajo reconociendo

patrones de imagen en capturas de pantalla, la IDE de Sikuli proporciona una útil funcionalidad que nos permite probar y ajustar la precisión del reconocimiento. Para hacer esto, pulsamos en la captura de pantalla deseada en el script, y Sikuli abre la ventana Preview, donde las áreas que coinciden con el patrón de la imagen en la captura de pantalla se marcan con rectángulos rojos. Esto puede ayudarnos a solucionar problemas con el script, por si se da el caso de que no pulsa el lugar adecuado en la interfaz especificada. De igual modo, podemos usar la barra deslizadora *Similarity* para hacer que el reconocimiento sea más o menos preciso. Una vez está listo el script, podemos exportarlo como un paquete ejecutable *.skl* eligiendo el comando *Export executable* desde el menú File. A continuación, podemos ejecutar el script sin tener que abrirlo en la IDE de Sikuli tecleando el siguiente comando en el terminal:

```
ruta/a/sikuli/sikuli-ide.sh z
script.skl
```

Reemplazamos *path/to/sikuli/* con la verdadera ruta al directorio del IDE de Sikuli y *script.skl* con el nombre del script que queremos ejecutar.

## Consideraciones Finales

Sikuli es un proyecto fascinante con un potencial tremendo. Aún está en su infancia, por lo que todavía hay lugar para la mejora. Hay que destacar que aún falta la documentación que cubre sus funcionalidades. Por ahora, la mejor manera de hacerse con las riendas de Sikuli es probar con él directamente y descubrir sus posibilidades escribiendo scripts. Para empezar, podemos usar la sección "Sikuli Script Commands for Jython" [2] de la documentación y el blog del proyecto [3]. ■

## RECURSOS

- [1] Sikuli: <http://groups.csail.mit.edu/uid/sikuli/>
- [2] Comandos de script de Sikuli para Jython: [http://sikuli.org/documentation.shtml# doc/pythonoc-python.edu.mit.csail.uid.Sikuli.html](http://sikuli.org/documentation.shtml#doc/pythonoc-python.edu.mit.csail.uid.Sikuli.html)
- [3] Blog oficial de Sikuli: <http://blog.sikuli.org/>