Identifying Elements with FEST

When someone is using FEST and has problems identifying the UI Elements.

I was working on a project which aimed to create a Swing Driver just like web driver so people wee versed in selenium can automate the Swing UI tests. So Ihad to decide on something to identify the UI components. I didn't had name for any UI component, therefore I struggled there a lot and had to depend on multiple other properties to identify that element. I used Swing Explorer to spy over the UI.

I was launching the JNLP file using Netx and the tests were getting stuck whenever a Modal Dialog box opened up. therefore i used AutoIT to close those Modal Windows.

Step-by-step guide

1. Create your locator something like this: UIClassId=TableUI;ToolTipText=Click Here
    a. Keep these locators in a Hashmap, code:
    String[] locatorsArray = locator.split(SPLITTER);
    this.locatorMap = new HashMap<String, String>(locatorsArray.length);
    String[] keyValuePairArray = new String[2];
    String temp;
    for (String keyValuePair : locatorsArray) {
    keyValuePairArray = keyValuePair.split("=");
    temp = keyValuePairArray[0];
    keyValuePairArray[0] = temp.substring(0, 1).toUpperCase() + temp.substring(1);
    if (isBooleanValue(keyValuePairArray[1])) { //if value is a boolean then the method will be something like: isVisible or isShowing otherwise the methods name are: getName, getTitle
    keyValuePairArray[0] = "is" + keyValuePairArray[0];
    } else {
    keyValuePairArray[0] = "get" + keyValuePairArray[0];
    }
    this.locatorMap.put(keyValuePairArray[0], keyValuePairArray[1]);
    }
2. Use Generic type Matcher along with reflections to identify that element, like this:
    a. Below is the example to find button:
    b. final GenericTypeMatcher<JButton> genericTypeMatcher = new
    GenericTypeMatcher<javax.swing.JButton>(javax.swing.JButton.class) {
    protected boolean isMatching(javax.swing.JButton comp1) {
    boolean result = true;
    for (String condition : locatorMap.keySet()) {
    try {
    Method method = getMethod(javax.swing.JButton.class, condition);//Method in Step 3
    String resp = (String) (method.invoke(comp1) + "");
    if (!((resp).contains(locatorMap.get(condition)))) {
    return false;
    }
    } catch (Exception e) {
    e.printStackTrace();
    } }
    }
    if (!comp1.isShowing()) {
    result = false;
    }
    return result;
    }
    };
    Pause.pause(new Condition("Waiting For The Element") {
    @Override
    public boolean test() {
    Collection<JButton> list = parentFrame.robot.finder().findAll(parentFrame.target,
    genericTypeMatcher);
    return list.size() > 0;
    }
    }, MAX_WAIT_TIME);
    buttonFixture = parentFrame.button(genericTypeMatcher);
3. Use the following method for reflection:
    a. protected static Method getMethod(Class<?> clazz, String name) {
    Method m = null;
    try {
    m = clazz.getDeclaredMethod(name);
    } catch (NoSuchMethodException e) {
    if (clazz.equals(Object.class))

```
throw new RuntimeException("Method not found in any super class.");
return getMethod(clazz.getSuperclass(), name);
}
return m;
}
```

4. This is my way to do it if someone has any other way to do it, Please let me know.

> To be used when working with FEST.

Related articles

- Identifying Elements with FEST
- Get Auto Commissions Review