Kapitel 15

Testen nativer Windows Anwendungen

5.0+

15.1 Einstieg

Dieses Kapitel behandelt das automatisierte Testen von Windows Desktop Anwendungen, insbesondere von

- Klassischen Win32 Anwendungen,
- .NET Anwendungen, basierend auf Windows Presentation Foundation (WPF) oder Windows Forms,
- Universal Windows Platform (UWP) Anwendungen, die XAML Steuerelemente verwenden.

Diese Anwendungstypen unterstützen die Microsoft UI Automation oder die Microsoft Active Accessibility (MSAA) für Software-Testautomatisierung, zusammengefasst in der Windows Automation API, siehe Abschnitt 15.2(170) für weitere Informationen.

QF-Test benötigt zum Testen eine Verbindung mit dem jeweiligen Prozess. Um einen entsprechenden Vorbereitung Knoten zu erstellen, ist es am einfachsten den Schnellstart-Assistent(31) im Menü Extras zu starten und dort 'Eine native Windows Anwendung' als Anwendungstyp auszuwählen.

Falls Ihre Anwendung bereits läuft, reicht es, den Titel des (Haupt-)Fensters als regulären Ausdruck anzugeben. Für den Windows Editor (Notepad) wäre dies zum Beispiel `.*- Editor`. Wenn die Anwendung durch QF-Test gestartet werden soll, geben Sie die ausführbare Datei (.exe) Ihrer Anwendung inklusive Pfad an, vgl. Start/Anbindung einer Applikation(171).

Wenn sich QF-Test zu der Anwendung (als GUI Engine(589) `win`) verbunden hat, können Tests aufgenommen und abgespielt werden wie in Kapitel 4(37) beschrieben. Auf Grund

der Eigenschaften der Microsoft UI Automation sind allerdings die in Abschnitt 15.4(172) aufgeführten Aufnahmeregeln zu beachten.

Beispiel-Testsuiten finden Sie in der QF-Test Installation:

- `qftest-5.0.0-ea/demo/carconfigForms/winDemoForms_de.qft`
- `qftest-5.0.0-ea/demo/carconfigWpf/winDemoWPF_de.qft`
- `qftest-5.0.0-ea/demo/windows/*.qft`

Bitte beachten Sie auch die (aktuellen) Einschränkungen, Abschnitt 15.9(176), von denen zu erwarten ist, dass die meisten in zukünftigen QF-Test Releases behoben oder verbessert werden.

15.2 Technischer Hintergrund

Ein verbreitetes Framework für Windows basierte Applikationen ist die Windows Automation API, bestehend aus der Microsoft Active Accessibility und dessen Nachfolger, der Microsoft UI Automation. Dies ist das Kernstück der `win` Engine, wodurch QF-Test in der Lage ist, fast alle Arten von Windows Applicationen zu steuern.

Eine Windows Anwendung musss sogenannte `Provider` zur Verfügung stellen, um die Regeln der UI Automation einzuhalten. Dies geschieht bei der Verwendung eines Frameworks wie WPF automatisch, bei Win32 Applikationen über Proxy Provider. Wie gut eine Applikation gestestet werden kann, hängt somit von der Qualität der jeweiligen `Provider` ab, d.h. dem Framework, das zur Entwicklung verwendet wurde. Da die UI Automation zusammen mit dem Framework WPF eingeführt wurde, sollten Anwendungen, die damit entwickelt wurden, gut testbar sein. Bei Entwicklungsplattformen ohne Integration der UI Automation sieht die Sache anders aus, zum Beispiel bei Java Swing. Aber dafür gibt es ja bereits ein ziemlich gutes Testtool ...

Wenn ein Programm über UI Automation getestet werden soll, so stehen sogenannte `Automation Elements` zur Verfügung, die die eigentlichen UI Elemente der zu testenden Anwendung darstellen. Obwohl jedes `Automation Element` einen `Control Type` hat (`Button`, `MenuItem` etc.), wird seine tatsächliche Funktionalität - zum Beispiel das Setzen eines Wertes in einem Textfeld - über `Control Patterns` bestimmt, die über den jeweiligen `Provider` implementiert sind.

Zur Bedienung des UI Automation Frameworks startet QF-Test ein spezielles Java Programm, das als UI Automation Client Applikation dient. Dieses Programm kann alle UI Automation Elemente eines bestimmten Prozesses ansprechen und diese entsprechend der QF-Test Regeln bedienen (zum Beispiel zum Erstellen eines Abbildes eines Elements als Komponente(736)).

15.3 Start/Anbindung einer Applikation

Für den Test einer nativen Windows Anwendung ist es nicht notwendig diese über QF-Test zu starten. Man kann QF-Test auch mit einem bereits laufenden Prozess verbinden. Dadurch ist es auch möglich, Bereiche des Betriebssystems zu steuern, zum Beispiel die Windows Taskbar.

Die Anbindung an einen laufenden Prozess erfolgt entweder über einen regulären Ausdruck für den Fenstertitel, die Prozess-Id oder den Klassennamen des Fensters, der über die UI Automation bereitgestellt wird. Die Anbindung ist für ein `Window` im Sinne eines UI Automation `Control Types` möglich, aber auch für ein `Pane` oder `Menu` Element. Unabhängig von der gewählten Verbindungsmethode, ermittelt QF-Test die jeweilige Prozess-Id und wird genau diesen Prozess als System Under Test (SUT) behandeln.

Der Windows Client starten(613) Knoten stellt die Verbindung her. Dabei muss für die Verbindung zu einer bereits laufenden Anwendung im Attribut Fenstertitel (RegExp)(614) einer der folgenden Werte eingetragen werden (wie bereits oben erwähnt):

- ein regulärer Ausdrück für den Fenstertitel
- -pid `<Prozess-Id>`
- -class `<class name>`

Beispiele:

`.*- Editor`: zur Anbindung an einen laufenden Windows Notepad Prozess,

`-class Shell_TrayWnd`: Anbindung der Windows Taskbar verbinden.

Für die Ermittlung des Fenstertitels, der Prozess-Id oder des `Class Name` des laufenden Programms steht die Prozedur `qfs.autowin.logUIAToplevels` in der Standardbibliothek(126) `qfs.qft` zur Verfügung.

Man kann die zu testende Anwendung aber auch direkt über den Windows Client starten Knoten starten. In diesem Fall geben Sie den Dateinamen der ausführbaren Datei, inklusive Pfad, im Attribut Windows Anwendung(613) an.

Sie können auch das Attribut Windows Anwendung zusammen mit dem Fenstertitel (RegExp) Attribut angeben. Dies ist nützlich, wenn die Anwendung gestartet werden soll, falls sie nicht bereits läuft. QF-Test prüft dann zuerst, ob es sich an einen laufenden Prozess, der dem Fenstertitel (RegExp) Attribut entspricht, anbinden kann. Falls nicht, wird das angegebene Programm gestartet und über seine Prozess-Id verbunden. Es kann vorkommen, dass dieser Prozess einen weiteren Prozess startet, der das eigentliche Programm darstellt, wobei ersterer selbst aber keine (grafische) Benutzeroberfläche hat oder sich sogar beendet. In diesem Fall wird ein weiterer Versuch durchgeführt um eine Verbindung zu dem zweiten Prozess zu erlangen.

Wenn Sie in QF-Test einen `win` Client beenden (entweder mittels des Programm beenden[617] Knotens oder über das Clients Menü), wird der entspechende UI Automation Client Prozess mitsamt seiner Unterprozesse gestoppt. Das heißt, dass die zu testende Anwendung nur dann beendet wird, wenn sie aus QF-Test heraus gestartet wurde, nicht aber, wenn sie bereits vor der Verbindung mit QF-Test gelaufen ist.

Wenn sie die zu testende Anwendung beenden, wird in jedem Fall auch der UI Automation Client beendet.

Um sich mit einem heraufgestuften Prozess verbinden zu können, muss QF-Test als Adminstrator gestartet werden.

15.4 Aufnahme

Nach erfolgreicher Verbindung von QF-Test mit der zu testenden Anwendung können Sie mit der Aufnahme von Aktionen (Abschnitt 4.1[37]), Checks (Abschnitt 4.3[40]) und Komponenten (Abschnitt 4.4[42]) beginnen.

Da die Kommunikation zwischen dem QF-Test UI Automation Client und der zu testenden Anwendung über Windows (konkret, den UI Automation Core) läuft, ist der Zugriff auf die Komponenten nicht ganz so schnell wie Sie dies vielleicht von der QF-Test Java Automatisierung kennen. Außerdem werden Events im Gegensatz zu Java- oder Web-Tests (QF-Driver) asynchron verarbeitet. Das bedeutet, dass Sie nicht davon ausgehen können, dass der Dispatch Thread der Applikation geblockt wird während QF-Test ein Ereignis bearbeitet.

Dies macht Aufnahmen schwieriger, wenn die Zielkomponente in Folge der aufzunehmenden Aktion verschwindet bevor QF-Test die Komponentenidentifizierung abgeschlossen hat. Zum Beispiel bei der Aufnahme eines Combobox-Eintrags, der die Liste nach dem Klick sofort schließt, oder eines Buttons, der das Fenster, in dem er liegt, beendet.

Daher ist es sinnvoll, wenn Sie folgende Vorgehensweise beachten:

- Aktivieren Sie den Aufnahmemodus und bewegen Sie die Maus zu dem Element, für das Sie eine Aktion aufnehmen wollen.
- Da QF-Test etwas Zeit benötigt um das Element unter dem Mauszeiger zu identifizieren, wird ein rotes Panel angezeigt bis der Vorgang abgeschlossen ist. In dem kleinen 'QF-Test Element Information' Fenster sehen Sie dann, welche Komponente erkannt wurde.
- Führen Sie nun die aufzunehmende Aktion aus.

- Wenn Sie einen Mausklick aufnehmen, der einen Dialog oder ein Fenster schließt (auch Popup-Listen), halten Sie die Maustaste ein wenig länger gedrückt, so dass QF-Test die Möglichkeit hat, die nötigen Informationen auszulesen bevor das Fenster verschwindet, was häufig fast zeitgleich mit dem Loslassen der Maustaste geschieht.
- Bei der Aufnahme von Checks oder Komponenten erscheint ein Rahmen um das Element sobald der Mauszeiger darüber fährt. Bitte warten Sie mit der Aufnahme bis der Rahmen wieder verschwunden ist.
- Wenn die zu testende Anwendung zwei oder mehr Fenster hat, dürfen sich diese bei einer Check-Aufnahme nicht überlappen.

Wenn Sie bei der Aufnahme einer Aktion Probleme haben, weil diese zum Beispiel das entsprechende Fenster schließt (Klick auf OK oder Abbrechen Button), kann es hilfreich sein, einen Check auf die entspechende Komponente aufzunehmen und diesen dann in die gewüschte Aktion zu konvertieren. Manchmal kann das Drücken der Maustaste auch dazu führen, dass ein Element neu generiert wird (zum Beispiel bei der Zubehörtabelle im CarConfiguratorNet WPF Demo). Auch hier kann die Check-Aufnahme helfen. Im Check-Aufnahmemodus legt QF-Test ein (fast) unsichtbares Fenster über die zu testende Anwendung um so zu verhindern, dass Mausklicks eine Aktion in der Anwendung bewirken.

15.5 Komponenten

Ein UI Automation Element wird von QF-Test als Fenster(726) beziehungsweise Komponente(736) innerhalb des Fenster und Komponenten(747) Knotens aufgenommen. Ein manuelles Einfügen von Knoten ist natürlich auch möglich. Die (generische) QF-Test Klasse entspricht häufig dem `Type` des UI Automation Elements, zum Beispiel bei `Button`. Um den `Type` vom Klassennamen unterscheiden zu können, setzt QF-Test den Prefix `Uia.` vor den `Type`. Analog wird der Kurzname des UI Automation Frameworks als Prefix für die UI Automation `Class` des Elements verwendet. Zum Beispiel würde QF-Test bei einer Tabelle einer WPF Anwendung in Weitere Merkmale des Komponente Knotens einen Eintrag mit `classname: WPF.DataGrid` anlegen.

QF-Test bildet die Hierarchie der UI Automation Elemente unter Umständen nicht eins zu eins im Komponentenbaum ab. Dies geschieht häufig bei Dialogen (wie zum Beispiel dem Schriftart-Dialog von Notepad), die üblicherweise in der Komponentenhierarchie der UI Automation unter dem Hauptfenster aufgeführt werden. Aus der Sicht von Win32, genauso wie das auch QF-Test Anwender erwarten würden, sind diese Dialoge top-level Windows und damit in Fenster und Komponenten parallel zum Hauptfenster angeordnet. Andererseits kann ein Kontextmenü, das ganz oben in der UI Automation Hierarchie angesiedelt ist, in QF-Test innerhalb eines Fensters zu finden sein.

15.6 Wiedergabe und Patterns

Die win Engine von QF-Test zeichnet Mausklicks mit aktiviertem Als ”harten” Event wiedergeben Attribut auf. Das ist der sichere Weg, da ein ”harter” Mausklick mit hoher Wahrscheinlichkeit die gewünschte Aktion auslösen wird.

Die UI Automation unterstützt jedoch auch verschiedene ”weiche” Aktionen, die nicht auf Mausklicks basieren. Zum Beispiel kann die Aktion einer Schaltfläche mittels

```
+Auswahl: invoke [myButtonID]
```

ausgelöst werden. Die Wirkung sollte die gleiche sein wie bei

```
+Mausklick [myButtonID]
```

Bei einem Auswahl wird die Maus nicht verwendet. Stattdessen löst der UI Automation Prozess die Ausführung einer Invoke() Methode des Providers im SUT aus.

Der Auswahl Knoten unterstützt die folgenden Aktionen im Attribut Detail:

- invoke: Meist gleichwertig mit einem Mausklick.
- expand, collapse: Dies sollte eine ComboBox, ein MenuItem oder ein TreeItem aus- beziehungsweise einklappen.
- select: Dies sollte einen Listeneintrag auswählen Hierzu bitte 0 als Detail eintragen. 1 und -1 dienen dazu, die Auswahl zu erweitern oder zu verringern, wenn Mehrfachauswahl möglich ist.
- scroll:horiz%,vert%: Erlaubt Werte zwischen 0 und 100, die die Scrollposition in Prozent angeben. -1, wenn die Position (Horizontal oder vertikal) nicht verändert werden soll.

Welche Aktionen tatsächlich unterstützt werden, hängt vom Pattern des UI Automation Elements ab. Diese werden unter Weitere Merkmale einer Komponente abgespeichert oder können in einem SUT Skript wie unten beschrieben ermittelt werden.

Die genaue Bedeutung eines Patterns kann sich von Anwendung zu Anwendung unterscheiden. Wenn sowohl SelectionItem als auch Invoke Pattern unterstützt werden, sollte Invoke bevorzugt werden, da

```
+Select [list@item]
```

eventuell nur das Element markiert, nicht jedoch die entsprechende Aktion auslöst. (Ein Beispiel hierfür ist die Schriftartenauswahl bei Notepad.)

Schlimmer noch, denn die formale Unterstützung eines Pattern heißt noch lange nicht, dass deren Anwendung irgendeine Wirkung zeigt. Dies ist zum Beispiel in der Rechner-Applikation von Windows der Fall, wenn man zu einem (nicht sichtbaren) Eintrag der Modus-Auswahlliste blättern möchte (ScrollItem pattern). Um dieses Pro-

blem zu umgehen, kann in diesem Fall ein `select` abgespielt werden, unabhängig davon, ob der Eintrag gerade sichtbar ist.

Weil das ”weiche” Abspielen von Aktionen häufig im `provider` nicht implementiert ist, und somit nicht funktioniert, oder wegen des Aufrufs der COM Methode sogar zur Blockade führt, zeichnet QF-Test wie bereits oben erwähnt ”harte” Mausklicks auf. Wenn Sie das Als ”harten” Event wiedergeben Attribut deaktivieren, wird stattdessen `invoke` abgespielt - wenn das Element das entsprechende `Pattern` (formal) unterstützt.

Bei Tastaturevents kann ein Text nur dann direkt über einen Texteingabe Knoten gesetzt werden, wenn das `Value pattern` unterstützt wird. Ansonsten muss der Text über einzelne Tastaturevents eingegeben werden.

15.7 Skripting

Intern stellt die `win` Engine ein UI Automation Element mit der Klasse `WinControl` dar. Um auf ein Element in einem Groovy SUT Skript(588) Knoten zuzugreifen, führen Sie folgendes Skript mit der passenden QF-Test ID der Komponente aus:

```
def ctrl = rc.getComponent("myComponentID")
println ctrl
```

Beispiel 15.1: Zugriff auf ein `WinControl` in einem Groovy SUT Skript

Die wichtigsten Methoden der Klasse `WinControl` sind:

- `getType()`, `getClassName()`, `getFramework()`, `getName()`, `getId()`, `getHwnd()`, `getLocation()`, `getSize()`, `getLocationOnScreen()`, `getPatterns()`, `hasPattern()` um die UI Automation Eigenschaften eines Elements zu erhalten
- `getChildren()`, `getParent()`, `getChildrenOfType()`, `getAncestorOfType()`, `getElementsByClassName()` für die Elementhierarchie
- `getControl()` um das jeweilige `control` von der Mmarque Ui-automation Bibliothek zu erhalten. Ein Snapshot dieser Bibliothek ist Teil der QF-Test Auslieferungen.

15.8 Optionen

Das Verhalten der `win` Engine kann über die QF-Test Optionen beeinflusst werden. Es kann zum Beispiel vorkommen, dass Sie eine Aktion auf ein Element abspielen wollen, das sich außerhalb des sichtbaren Bereichs befindet. Um eine `invoke` Aktion abspielen zu können ohne das Element zuerst in den sichtbaren Bereich zu bringen, können Sie die Sichtbarkeitsüberprüfung, die normalerweise Bestandteil der Komponentenerkennung ist, ausschalten. Führen Sie hierzu

```
rc.setOption(Options.OPT_WIN_TEST_VISIBILITY, false)
```

in einem SUT Skript Knoten aus bevor Sie

```
+Auswahl: invoke [myComponentID]
```

abspielen. Setzen Sie anschließend die Option wieder mittels

```
rc.unsetOption(Options.OPT_WIN_TEST_VISIBILITY)
```

auf den ursprünglichen Wert zurück.

Eine weitere Möglichkeit das Verhalten von QF-Test anzupassen ist das Setzen von Parametern für den (nativen) WinDriver, der als Schnittstelle zwischen Java und der Windows UI Automation dient. Dies geschieht in einem SUT Skript Knoten mittels `rc.engine.preferences()`. Zum Beispiel hebt

```
def prefs = rc.engine.preferences()
prefs.setPref("windriver.restrict.tops.to.class", "false")
```

Beispiel 15.2: Einstellungen in einem Groovy SUT Skript setzen

die Beschränkung auf Top-Level Elemente einer bestimmten UI Automation `class` auf, wenn die Anwendung über `-class <class name>` verbunden wurde. Dadurch können dann auch Top-Level Elemente einer anderen Klasse in diesem Prozess angesprochen werden.

15.9 (Aktuelle) Einschränkungen

Die aktuelle Implementierung des Windows-Testens enthält noch eine Reihe von Einschränkungen. Wir werden versuchen, die Funktionalität laufend zu verbessern, doch mag die eine oder andere Einschränkung noch eine Weile bestehen.

Um Probleme mit der Geometrie (Abmessung eines Elements, Koordinaten für einen ”harten” Mausklick) zu vermeiden, sollte die Skalierung von Windows auf 100% für die Aufnahme und die Wiedergabe gesetzt werden. Diese Einschränkung soll in einer zukünftigen QF-Test Version beseitigt werden.

Da die Unterstützung der UI Automation vom Framework abhängt, mit dem die Anwendung entwickelt wurde, ist die Aufnahme in QF-Test eventuell nicht konsistent. Zum Beispiel kann beim Öffnen eines Dialogs ein Warten auf Komponente aufgezeichnet werden oder auch nicht.

Das Testen von Anwendungen, die aus mehreren Prozessen bestehen, ist komplex und erfordert mehrere `win` Clients.

Weitere Einschrändunken und noch nicht implementierte Funktionalitäten (Stand Jan. 2019) sind unter anderem:

- Die Kompontenaufnahme ist nur für Einzelkomponenten möglich (nicht für Kind-Elemente oder das ganze Fenster). Dies soll in einer zukünftigen QF-Test Version behoben werden.
- Die unterstützten Check-Arten sind nicht vollständig. 'Text', 'Abbild', 'Geometrie' und 'Sichtbar' sind für alle WinControls implementiert. Zusätzliche Checks können für Tabellen, Listen und andere UI Automation `types` zur Verfügung stehen. Dies soll in einer zukünftigen QF-Test Version vervollständigt werden.
- Elemente einer Titelleiste einer Windows App können nicht (einfach) angesprochen werden, weil diese in einem anderen Prozess liegen. Dies könnte in einer zukünftigen QF-Test Version behoben werden.
- MenuItems in .NET Forms Anwendungen werden nicht richtig aufgezeichnet. Man erhält nur einen Mausklick auf ein Popup-Menü mit der entsprechenden Position. Die Wiedergabe sollte jedoch funktionieren. Dies könnte in einer zukünftigen QF-Test Version behoben werden. Als Workaround können Sie ein generisches MenuItem anlegen (siehe Abschnitt 35.6(376)).
- Die Event-Weiterleitung vom Textelement einer Schaltfläche auf das Schaltflächenelement selbst erfolgt bei der Aufnahme eines Mausklicks, kann aber an anderer Stelle fehlen. Dies sollte in einer zukünftigen QF-Test Version behoben sein.
- Die Verwendung von Groovy SUT Skript Knoten ist gegenüber Jython vorzuziehen. Letzterer unterstützt aktuell noch nicht `WinControl.getControl()`. Dies wird in einer zukünftigen QF-Test Version behoben sein.
- Tiefe Hierarchien an UI Automation Elementen können zu Performanzproblemen führen. Dies sollte in einer zukünftigen QF-Test Version behoben sein.

15.10 Links

Das Windows Automation API ist beschrieben unter: https://docs.microsoft.com/en-US/windows/desktop/WinAuto/windows-automation-api-portal.

Weitere Informationen zu Mark Humphreys ui-automation Java Bibliothek finden Sie unter https://github.com/mmarquee.