SmartIDs

Komponenten ohne Aufnahme

QFS
Quality First Software

Das Ziel

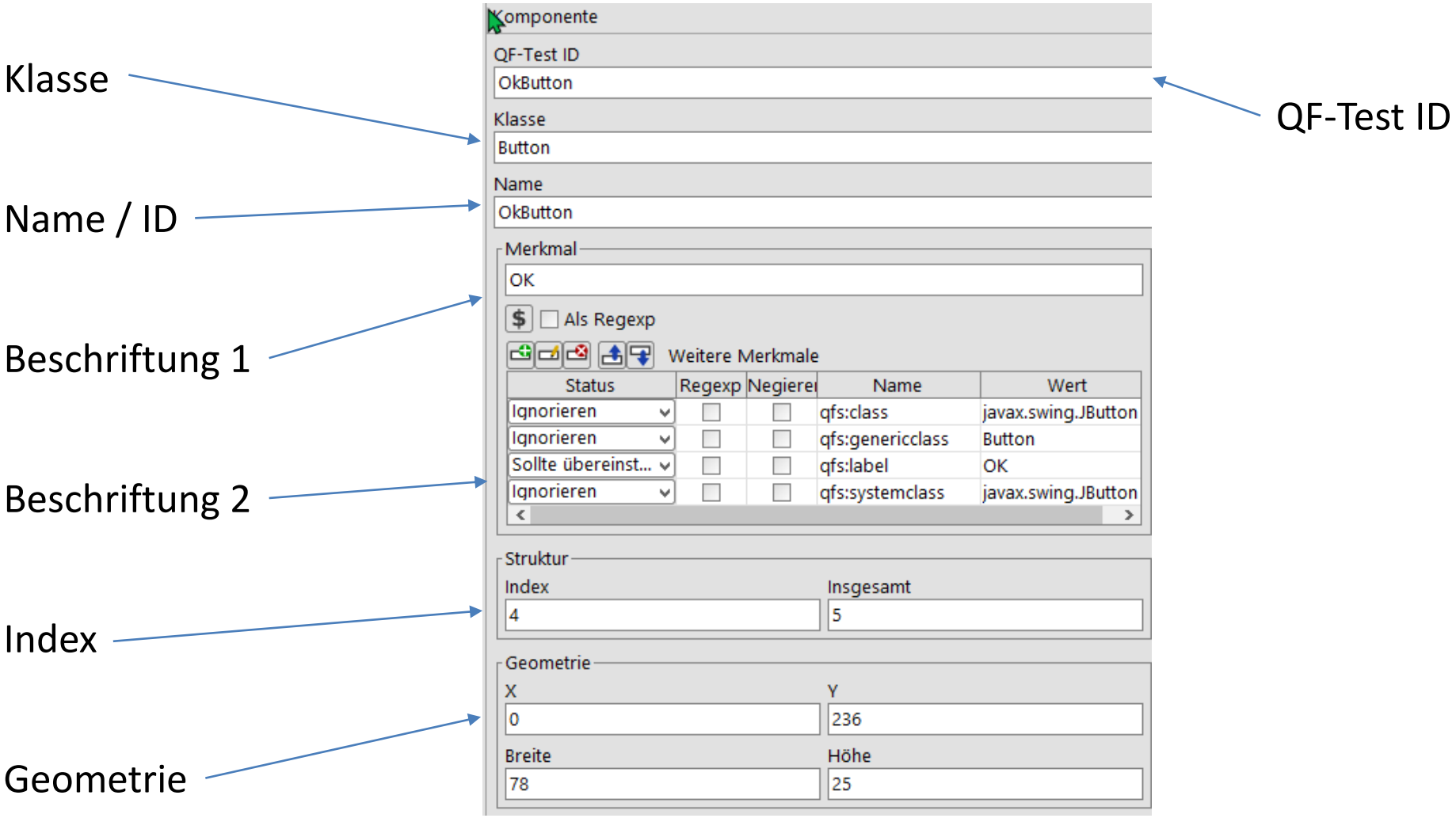
Flexible, stabile Wiedererkennung von Komponenten,
ohne dass die Wiedererkennungskriterien abgespeichert werden.

Maxime: Keep simple things simple

Klassische Komponente

QF-TEST

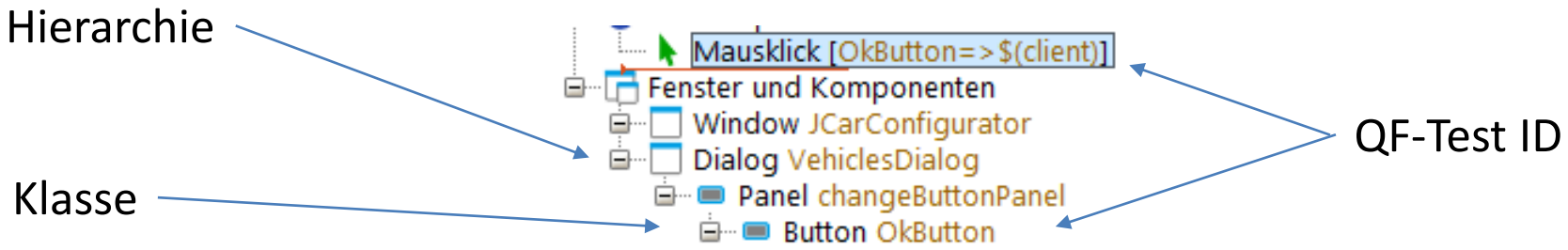
Klassisch:



Klassische Komponente

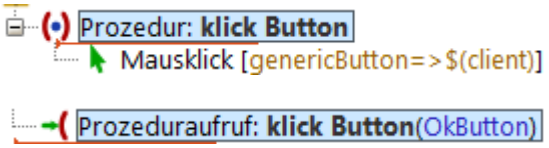
QF-TEST

Im Komponentenbaum:

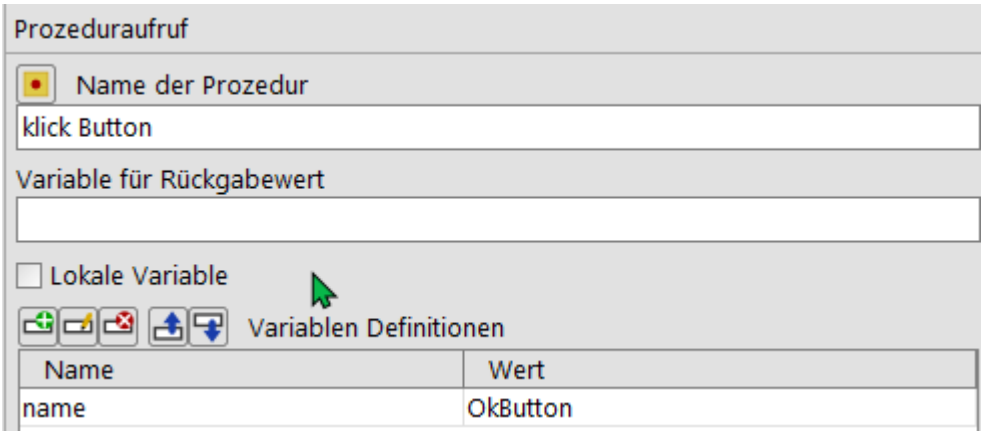


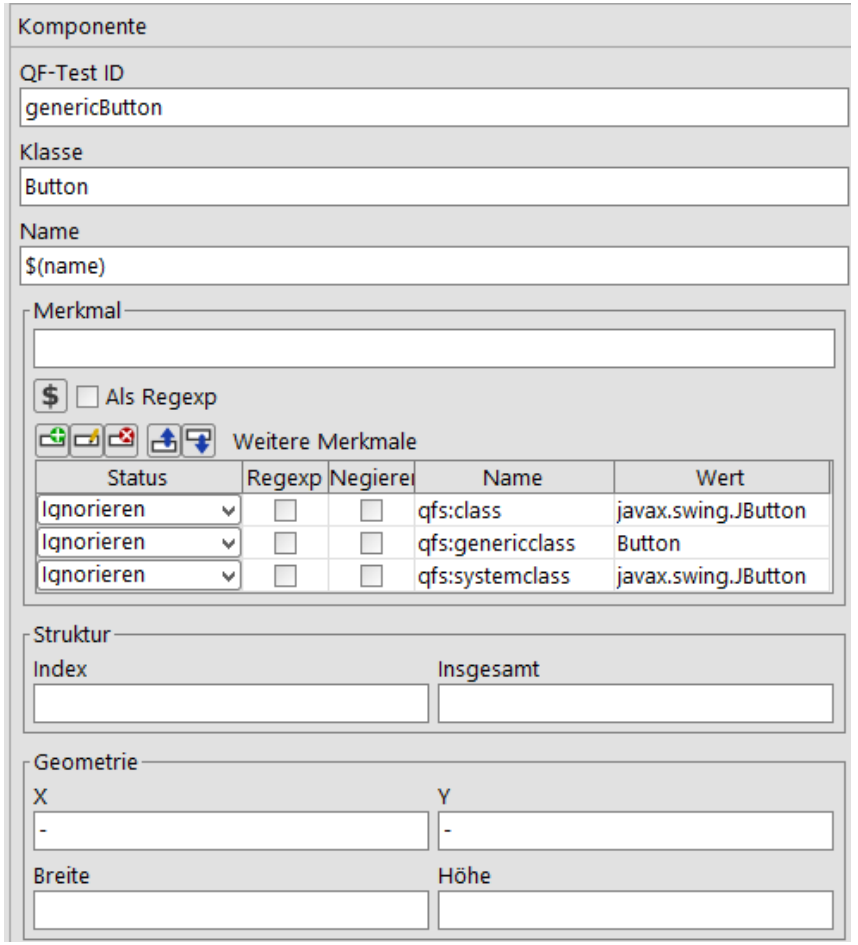
Mit generischen Komponenten:

1) Komponente mit Name



Name als Variable

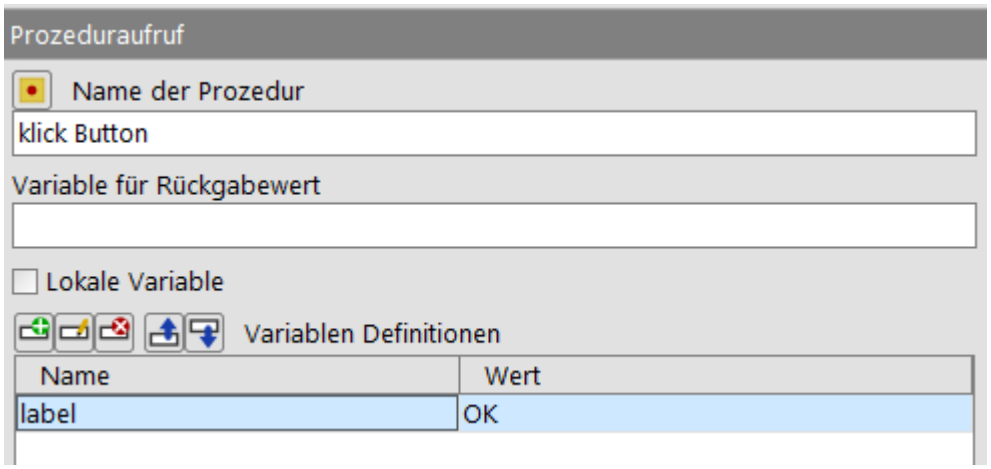


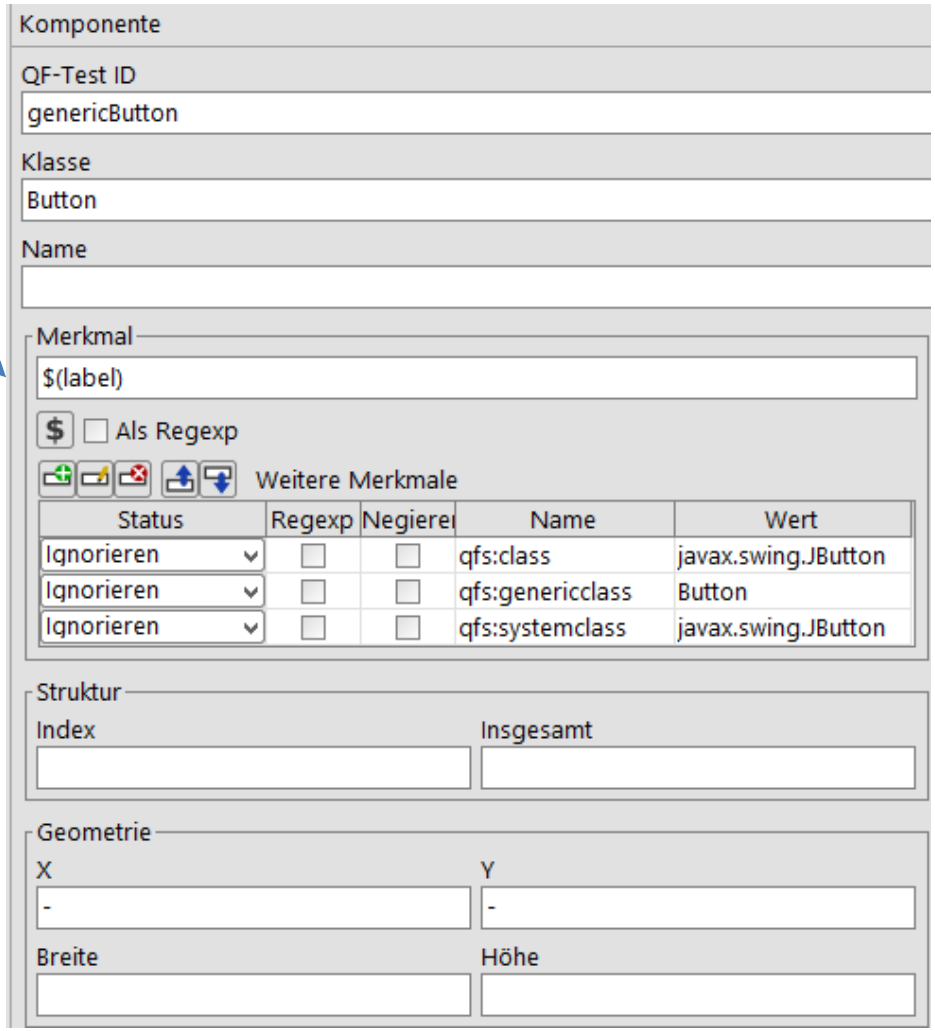


Mit generischen Komponenten:

2) Komponente mit Label

Beschriftung als Variable

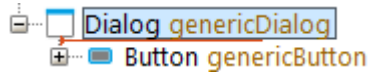




Generische Komponente

QF-TEST

Mit generischen Komponenten:



3) Komponente in modalem Dialog

Prozeduraufruf

Name der Prozedur

klick Button

Variable für Rückgabewert

Lokale Variable

Variablen Definitionen

Name	Wert
label	OK
dialogName	VehiclesDialog

Generischer Dialog

Fenster

QF-Test ID

genericDialog

Klasse

Dialog

Name

$(dialogName)

Generischer Button

Komponente

QF-Test ID

genericButton

Klasse

Button

Name

Merkmal

$(label)

Generische Komponente

QF-TEST

Vorteile:

- Schlanker, übersichtlicher Komponentenbaum
- Testgenerierung ohne Aufnahme der Komponente
- Lesbarkeit der Tests

Nachteile:

- Generische Komponente für
  - Jede Klasse
  - Art der Erkennung (Name, Label)
  - Struktur/Hierarchie (Dialog, Panel…)
- Wartung durch Suchen/Ersetzen
- Komponentenerkennungskriterien reduziert
- Unter Umständen Performanz

SmartIDs

QF-TEST

- Für eine SmartID wird aus den möglichen Wieder-erkennungsmerkmalen ein einziges oder eine bestimmte Kombination explizit ausgewählt.
- Die Erkennungskriterien werden direkt angegeben an Stelle der QF-Test ID der Komponente.
- Es wird kein Komponentenknoten verwendet.
- Verfügbare Kriterien für eine SmartID:
    - Generische Klasse
    - Name (ID)
    - Merkmal
    - Weitere Merkmale (qfs:label)
    - Index
    - Komponentenhierarchie

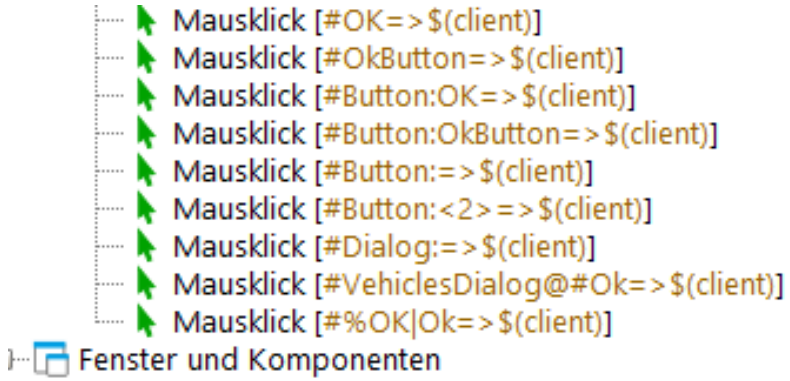
SmartIDs - Syntax

QF-TEST

Form
#Label
#Name
#Klasse:Label
#Klasse:Name
#Klasse:
#Klasse:<2>
#Parent@#Child
#%Regexp
#TabPanel:@Tab1
#Tab:Tab1
Andere Kombinationen

Allgemeine Syntax:

#[%][noscope:][ENGINE:][KLASSE:][WERT][<INDEX>]

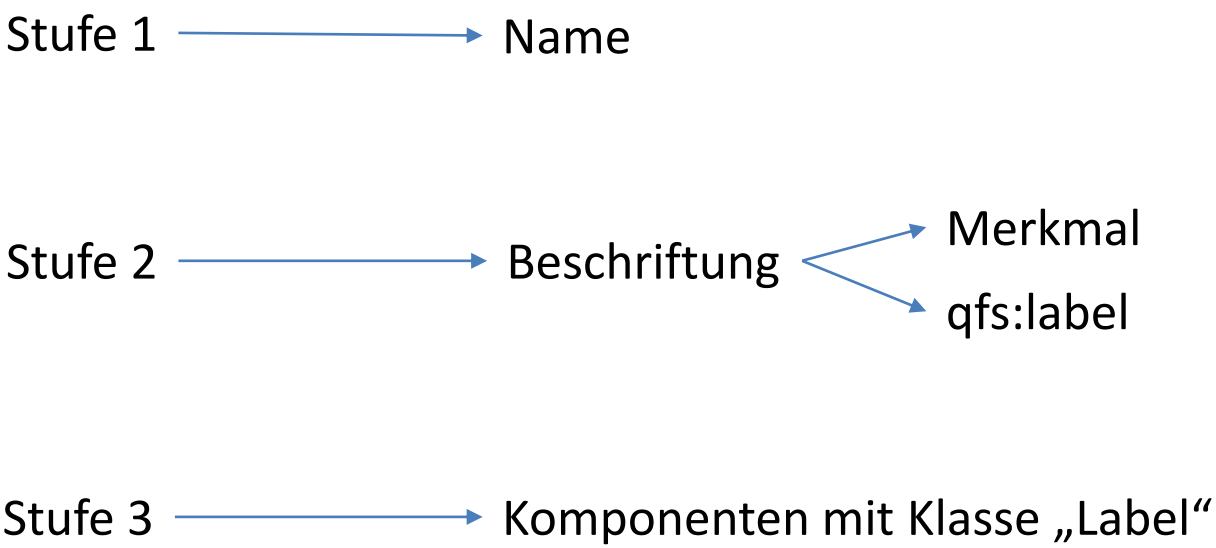


Die Wiedererkennungsmerkmale werden nicht gespeichert. Kein Komponentenbaum.

% Kennzeichnet WERT als regulären Ausdruck.

SmartIDs - Gewichtung

Wenn durch eine SmartID mehrere Komponenten erkannt werden können, gilt die Gewichtung*:



-> Gleiche Stabilität und Performanz wie klassisch: #Name

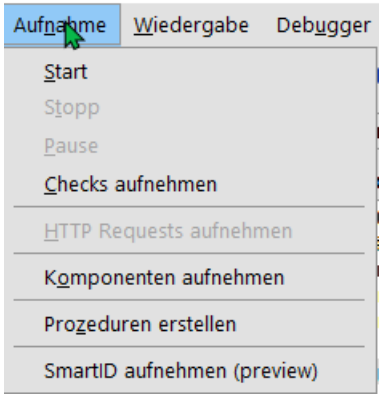
*) Kann über Option geändert werden

SmartIDs – Vor- und Nachteile

QF-TEST

- Vorteile
  - Kein Komponentenbaum
  - Testimplementierung ohne Komponentenaufnahme
  - Erleichtert der Lesbarkeit der Knoten
  - Explizite Festlegung der Erkennungskriterien
- Nachteile
  - Wartung durch Suchen/Ersetzen
  - Unter Umständen Performanz
  - Erkennungskriterien reduziert

Ab 6.0.0 als Preview vollständig unterstützt



Im Menü „Aufnahme“ erscheint ein neuer Eintrag: „SmartID aufnehmen (preview)“

Beschriftung mit Vorrang aufnehmen:

```
rc.setOption(Options.OPT_RECORD_SMARTID_PRIORITIES, "label,name")
```

SmartIDs - Anwendungsbeispiele

QF-TEST

- Ignorieren der Komponentenhierarchie
- Lesbarkeit der Tests
- Testgetriebene Entwicklung (TDD)
- Schlüsselwortbasierte Tests (KDT, BDT)
- Integration mit anderen Test-Tools
  zum Beispiel Robot-Framework
- Technologieübergreifende Tests

Scope = Gültigkeitsbereich

QF-TEST

- Container-Komponenten können einen Scope erhalten (Window, Dialog, Panel…)
- Ein Scope schränkt den Suchbereich für die nachfolgend adressierten Komponenten ein
- Beschleunigt die Komponentenerkennung
- Verbessert die Lesbarkeit des Tests

Scope - Gültigkeitsbereich

QF-TEST

- Setzen:
  - "@scope <SmartID>" als Doctag in der Bemerkung eines Knotens
  - "@scope <QF-Test ID der Komponente>" als Doctag in der Bemerkung eines Knotens
  - implizit bei Dialogen und Popups

Scopes können ineinander geschachtelt werden.

Scope - Gültigkeitsbereich

QF-TEST

- Löschen:
  - Wenn der Knoten, für den er gesetzt wurde, verlassen wird
  - Wenn der nächste Knoten keinen (@noscope) oder einen anderen Scope erhält
  - Implizit beim Schließen des Dialogs oder Popups

SmartIDs – Hinweise

Auf Eindeutigkeit achten

- Klasse zusätzlich spezifizieren

  #TextField:Vorname
- Scope verwenden

  @scope #Kundenadresse
- Index verwenden

  #Table:&0&5@#Button:<1>

Wenn eine SmartID bei der Aufnahme zu kompliziert werden würde, wird ein Komponentenknoten aufgenommen.

SmartIDs - Kundenreaktion

Die Einführung von **Scopes** und **SmartIDs** war ein entscheidender Sprung für QF-Test. Es hat die Tore für generische Lösungen weit geöffnet. Scope und SmartIDs sind sehr leicht zu erklären, zu verstehen und einzusetzen. Sie sind zudem super mächtig, leicht pflegbar, parametrisierbar, zentralisierbar und bringen viel Genauigkeit und Stabilität mit.

Ich habe drei Monate nach meiner Einführung in die Scopes und SmartIDs alle Testautomatisierungsprojekte, die ich zu dem Zeitpunkt betreut habe, auf SmartIDs umgestellt. Zudem implementierte ich eine modellbasierte Software-Lösung „automated test automation tool“, die für modellbasierte Anwendungen Testcode generiert und mit QF-Test die UI automatisiert befüllt und überprüft. Das hat die Implementierungs- und Pflegeaufwände in den Projekten erheblich reduziert. Spätestens jetzt wurden QF-Test und ich bei mgm sehr sichtbar und nicht mehr aufzuhalten.

Lilia Gargouri, mgm, im Interview mit Thomas Max, QFS