

SPLG-Grouper Testcase

Inhalt

Einführung.....	1
Erstellen Testfall aus Grouper	1
Format eines Testfalls	2
Ausführen eines Testfalls	3
Testfall über Kommandozeile spezifizieren.....	4
Spitallisten.....	4
Kommando-Referenz	6

Einführung

Die beiden Integrationsmodule Java und Dotnet können rasch und unkompliziert Testfälle erzeugen und testen. Dies ist wichtig, falls ein Kunde oder IT-Partner der AFG ein unerklärliches oder falsches Verhalten analysieren und melden will. Durch das Erzeugen eines Testcases erhöht sich die Chance, dass das Verhalten bei der AFG rasch nachvollzogen und analysiert werden kann. Für die IT-Partner reduziert sich der Aufwand für das Analysieren und Melden von möglichen Problemen.

Idealerweise wird direkt im System des Kunden eine Möglichkeit angeboten, zu einem spezifischen Fall den passenden Testfall zu erzeugen. Dieser Testfall kann dann analysiert werden und an die AFG eingeschickt werden für Analyse und ggf. Korrektur.

In folgenden werden Codebeispiele des Integrationsmoduls Java verwendet. Für das Integrationsmodul Dotnet funktioniert alles analog, ausser dass die Funktionsnamen grossgeschrieben werden.

Erstellen Testfall aus Grouper

Die Grouperklasse besitzt Funktionen, mit denen Testfälle erzeugt werden können:

```
String testcase = grouper.getTestcase(fall);
```

Dabei ist «fall» eine Instanz der Klasse «Falldaten». Der Testcase wird dann für diesen Fall und die aktuelle Konfiguration des Groupers erstellt.

Wurde der Fall bereits gruppiert und liegt eine Instanz der Klasse «Result» vor, so kann diese ebenfalls übergeben werden. In dem Fall wird der Testfall mit den erzielten Resultaten angereichert:

```
String testcase = grouper.getTestcase(fall, result);
```

Alternativ kann der Testfall in eine Datei geschrieben werden:

```
grouper.writeTestcase(fall, "C:/Temp/Testcase.txt");
```

Und mit den Resultaten:

```
grouper.writeTestcase(fall, result, "C:/Temp/Testcase.txt");
```

Wahlweise kann der Testfall auch mit Detail-Log-Informationen angereichert werden:

```
grouper.writeTestcaseWithLog(fall, result, "C:/Temp/Testcase.txt");
```

Dies funktioniert nur, wenn zuerst die Gruppierung durchgeführt wird und unmittelbar danach der obige Aufruf stattfindet (das Detail-Log wird für jeden gruppierten Fall neu angelegt).

Wir empfehlen den IT-Partnern, welche die Integrationsmodule in ihre Systeme und Produkte integrieren, eine Möglichkeit zu schaffen, um unkompliziert solche Testfälle erzeugen zu können. Dies kann beispielsweise so geschehen, dass mit einem Konfigurationsschalter die Erzeugung der Testfälle ein- oder ausgeschaltet wird, oder dass die Testfälle immer in ein Logfile geschrieben werden, oder dass im Produkt selbst über eine Tastenkombination oder ein GUI-Element auf den Testfall zugegriffen werden kann.

Bei Fragen zu einem konkreten Gruppierungsfall sind wir froh, wenn wir von den Kunden/IT-Partnern jeweils den passenden Testfall erhalten. Das liefert uns alle wichtigen Informationen, um den Sachverhalt analysieren zu können.

Format eines Testfalls

Hier ein Beispiel eines einfachen, aber kompletten Testfalles:

```
# SPLG Testcase
# Timestamp    2024-09-23 11:19:40
# Java         21.0.3 Eclipse Adoptium
# OS           Windows 10 10.0 amd64
# CPU          4 cores
# Memory       4096 MiB
# Grouper      2025.0.1
# Defversion   A_2024_4
```

```
licfile        config\license.txt
deffile        config\defs-demo.dat
release        A_2024
```

```
burnr          12345678
wohnkanton     ZH
austritt       20240510
agey           20
```

```
diagnose M169
diagnose I1090
```

```
behandlung 815111,,20240503
op 7601234567890,1,1
```

```
bewegung 20240501,20240510,1,12345678
```

```
# Output:
# SPLG:      BEW7.1
# MFZS:      BEW7.1
# MFZ0:      BEW7.1
```

```
# Operateure:  [[BEW7.1*1.0]7601234567890,[BEW7.1]]
# LACTRL:      99
# Lactrlcodes:
# Errorcode:   0
```

Ein Testfall besteht aus Textzeilen. Zeilen, welche mit # beginnen, stellen Kommentare dar und werden bei der Ausführung des Testfalles ignoriert. Sie enthalten trotzdem wichtige Informationen über das System, den Grouper oder auch die Resultate der Gruppierung des Falls.

Die eigentlichen Daten des Testfalles werden in Blöcke gruppiert.

Der erste Block zeigt auf, welche Lizenz-, Definitions- und ggf. Spitalisten-Datei verwendet wurde.

Der zweite Block enthält die Basisinformationen des Falls, wie beispielsweise die BURNr, den Wohnkanton, das Austrittsdatum, das Alter des Patienten etc.

Der dritte Block enthält die Diagnosen.

Der vierte Block enthält die Behandlungen und ggf. die Operierenden.

Der fünfte Block enthält die Patientenbewegungen.

Danach folgt ggf. noch ein auskommentierter Block, der die Resultate der Gruppierung enthält.

Optional folgt noch ein letzter auskommentierter Block, der die Detail-Log-Informationen enthält, welche aufzeigen, wie der Grouper das Regelwerk anwendet, um zum Resultat zu kommen. Diese Informationen sind detailliert und sehr technisch, bieten aber eine gute Möglichkeit zu verstehen, wieso der Grouper ein bestimmtes Resultat ausgibt.

Jeder Block ist in Zeilen strukturiert. Jede Zeile enthält ein Kommando, bestehend aus einem Schlüsselwort und einem Wert, getrennt durch Leerzeichen. Die möglichen Kommandos sind weiter unten spezifiziert.

Damit dokumentiert der Testfall sowohl das System, auf dem die Gruppierung vorgenommen wurde, die Konfiguration des Groupers, die Inputdaten des Falles sowie – sofern angegeben – die Gruppierungsergebnisse und Detail-Log-Informationen.

Ein Testfall kann als Input verwendet werden, um die Gruppierung zu testen.

Ausführen eines Testfalls

Liegt ein Testfall als Datei vor, kann er ausgeführt werden.

Für das Integrationsmodul Java sieht der Aufruf wie folgt aus:

```
java -jar splg-grouper.jar test input <dateiname>
```

Für das Integrationsmodul Dotnet sieht der Aufruf wie folgt aus:

```
testcase.exe input <dateiname>
```

Dabei stellt <dateiname> den Dateinamen der Testfalldatei dar.

Testfall über Kommandozeile spezifizieren

Ein Testfall kann auch direkt über die Kommandozeile spezifiziert werden. Als einfaches Beispiel (mit dem Java-Modul, geht analog auch mit Dotnet-Modul):

```
java -jar splg-grouper.jar test licfile config\license.txt
deffile config\defs-demo.dat slfile config\spitallisten-demo.dat
release A_2024 burnr 52599003 austritt 20240110 agey 70
behandlung 815111
```

Die gesamte Angabe erfolgt auf einer Zeile, auch wenn hier aus Platzgründen vier Zeilen verwendet werden.

Es ist auch möglich, gewisse Angaben aus einer Datei zu lesen, andere direkt auf der Kommandozeile anzugeben. Beispielsweise sind die ersten drei Kommandos «licfile», «deffile» und «slfile» normalerweise immer vorhanden. Darum können diese drei auch in einer Datei «cfg.txt» gespeichert werden:

```
licfile config\license.txt
deffile config\defs-demo.dat
slfile config\spitallisten-demo.dat
```

Nun kann der obige Testaufruf vereinfacht werden:

```
java -jar splg-grouper.jar test input cfg.txt release A_2024
burnr 52599003 austritt 20240110 agey 70 behandlung 815111
```

Es ist auch möglich, die «input» Kommandos zu verschachteln, also per «input» eine Datei zu laden, in welche wiederum per «input» weitere Dateien geladen werden...

Spitallisten

Die Spitallisten können entweder direkt aus der Standard-Spitallistendatei referenziert werden (d.h. die Spitallistendatei wird mit «slfile» spezifiziert und das «release» Kommando wählt dann die zum Release passende(n) Spitallisten aus), oder ad hoc über das «sl» Kommando.

Das «sl» Kommando erlaubt damit einfache Experimente mit verschiedenen Spitallisten und Leistungsaufträgen. Beispiel:

```
java -jar splg-grouper.jar test input cfg.txt release A_2024
burnr 12345678 austritt 20240110 agey 70 behandlung 815111
sl 12345678,,,1,ZH,2024,BP,BEW7.1,BEW7.1.1,BEW7.2,BEW7.2.1
```

Hier wurde das obige Beispiel abgewandelt, indem einerseits eine fiktive BURNr 12345678 verwendet wurde, und andererseits eine für diese BURNr passende Spitalliste mittels «sl» Kommando spezifiziert wurde. Die Ausgabe sieht wie folgt aus:

```
# SPLG Testcase
# Timestamp    2024-09-23 15:55:36
# Java         21.0.3 Eclipse Adoptium
# OS           Windows 10 10.0 amd64
# CPU          4 cores
# Memory       4096 MiB
```

```
# Grouper      2025.0.1
# Defversion   A_2024_4
# Spitalliste  A_2024_1_zh.json
```

```
# LA 12345678 2024: BP,BEW7.1,BEW7.1.1,BEW7.2,BEW7.2.1
```

```
licfile      config\license.txt
deffile      config\defs-demo.dat
slfile       config\spitallisten-demo.dat
release      A_2024
sl           12345678,,1,ZH,2024,BP,BEW7.1,BEW7.1.1,BEW7.2,BEW7.2.1
```

```
burnr       12345678
austritt     20240110
agey        70
```

```
behandlung 815111
```

```
# Output:
# SPLG:      BEW7.1
# MFZS:      BEW7.1
# MFZO:      BEW7.1
# Operateure: [[BEW7.1]]
# LACTRL:    0
# Lactrlcodes:
# Errorcode:  w31
```

Beachten Sie nach dem ersten (auskommentierten) Block die für den Fall verwendete Spitalliste. Diese Ausgabe wird sowohl für manuell spezifizierte Spitallisten (wie in unserem Beispiel) als auch für vorgegebene Spitallisten getätigt und erlaubt die effiziente Kontrolle der tatsächlich verwendeten Spitalliste. Allfällige Befristungen von SPLG werden explizit ausgewiesen. Sind keine Befristungen ausgewiesen, gelten die SPLG generell für das gesamte Datenjahr (nicht aber für andere Jahre!).

Kommando-Referenz

`input <filename>`

Liest die Datei «filename» ein und verarbeitet die darin enthaltenen Kommandos.

Beispiel: `input cfg.txt`

`output <filename>`

Schreibt die Ausgabe bei der Ausführung des Testfalls in die Datei «filename». Es können Platzhalter für «burnr», «plz», «standort» und «fallid» in «filename» verwendet werden, siehe Beispiel.

Beispiel: `output testfall-{burnr}-{fallid}.txt`

`licfile <filename>`

Spezifiziert die zu verwendende Lizenzdatei «filename». Die Lizenz muss für das gewünschte Datenjahr gültig sein.

Beispiel: `licfile config\license.txt`

`deffile <filename>`

Spezifiziert die zu verwendende Definitionsdatei «filename». Mit dem «release» Kommando kann dann eine darin enthaltene Definition aktiviert werden.

Beispiel: `deffile config\defs.dat`

`slfile <filename>`

Spezifiziert die zu verwendende Spitallistendatei «filename». Mit dem «release» Kommando kann dann eine darin enthaltene Spitalliste aktiviert werden.

Beispiel: `slfile config\spitallisten.dat`

`release <release>`

Wählt den zu verwendenden Release aus. Damit wird sowohl die Definition als auch die Spitalliste (sofern vorhanden) aktiviert. Der Release muss typischerweise passend zum Datenjahr sein. Das heisst, für Daten 2024 wird einer der folgenden Releases gewählt: «A_2024», «P_2024», «R_2024», «A_2024_legacy».

Beispiel: `release A_2024`

`sl <burnr,plz,standort,referenz,kanton,jahr,splg:valid_from:valid_to,...>`

Spezifiziert eine Spitalliste. Bei den SPLG sind «valid_from» und «valid_to» optional und werden nur verwendet, wenn eine SPLG nur für einen Teil des Jahres gültig ist. Wird eine Standort-BURNr verwendet (ab 2024 sollte das immer der Fall sein), können «plz» und «standort» leer gelassen werden. «referenz» ist «1», falls die Spitalliste eine Referenzspitalliste ist, ansonsten «0» (Referenzspitallisten werden dann verwendet, wenn für den Wohnkanton keine spezifische Spitalliste für den Kanton vorliegt).

Beispiel: `sl 12345678,,1,ZH,2024,BP,BEW7.1,BEW7.2:0106:3112`

kantonsoverride <wert>

Mit diesem Kommando kann man den Wohnkanton des Falles überschreiben. Das ist für das Auswählen der korrekten Spitalliste ggf. wichtig. «wert» ist das Kantonskürzel. Mit dem Wert «-» kann der Kantonsoverride zurückgesetzt werden.

Beispiel: kantonsoverride ZH

burnr <wert>

Setzt die BURNr des Betriebs.

Beispiel: burnr 52599003

plz <wert>

Setzt die PLZ des Betriebs.

Beispiel: plz 8400

standort <wert>

Setzt die Standortnummer des Betriebs.

Beispiel: standort 01

wohnkanton <wert>

Setzt den Wohnkanton des Patienten.

Beispiel: wohnkanton ZH

austritt <wert>

Setzt das Austrittsdatum des Patienten.

Beispiel: austritt 20240501

agey <wert>

Setzt das Alter in Jahren des Patienten.

Beispiel: agey 70

aged <wert>

Setzt das Alter in Tagen des Patienten (nur bei Neugeborenen mit «agey» gleich «0»).

Beispiel: aged 10

ssw <wert>

Setzt das Gestationsalter.

Beispiel: ssw 390

ggw <wert>

Setzt das Geburtsgewicht.

Beispiel: ggw 2800

dmb <wert>

Setzt die Dauer manueller Beatmung.

Beispiel: dmb 60

freiwilligkeit <wert>

Setzt die Freiwilligkeit (nur Psychiatrie).

Beispiel: freiwilligkeit 4

diagnose <code,seitigkeit>

Fügt eine Diagnose hinzu. Die Seitigkeit kann weggelassen werden.

Beispiel: diagnose F100

behandlung <code,seitigkeit,beginn,ambext>

Fügt eine Behandlung hinzu. Die Felder ambext, beginn und seitigkeit können weggelassen werden. Kommas am Ende können weggelassen werden.

Beispiel: behandlung 815111,,2024010110

Beispiel: behandlung 1234

Beispiel: behandlung 4321,2,2024010110,3

op <gln,funktion,zulassung>

Fügt der letzten Behandlung einen Operateur hinzu.

Beispiel: op 7601123456789,1,1

bewegung <beginn,ende,art,burnr>

Fügt eine Patientenbewegung (Episode) hinzu.

Beispiel: bewegung 20240101,20240110,1,52599003