

SPLG-Grouper REST-Schnittstelle

Einführung

Die beiden Integrationsmodule Java und Dotnet können als REST-Schnittstelle gestartet und verwendet werden. Dieses Dokument beschreibt, wie der Service gestartet und verwendet wird.

Die Verwendung wird mit Hilfe der Kommandozeilenprogramme «curl» und «jq» gezeigt. Selbstverständlich können Sie beliebige andere HTTP-Clienttools verwenden (beispielsweise Postman).

Der Service ist für lokalen Gebrauch gedacht und nicht dafür, über das Internet erreichbar zu sein. Darum wird normalerweise auf «localhost» verbunden und steht kein SSL zur Verfügung. Wenn Sie SSL benötigen, müssten Sie den Service hinter einen Reverse Proxy platzieren, wobei dann der Reverse Proxy über SSL erreichbar ist.

Starten des Service

Wenn Sie das Integrationsmodul Java verwenden, starten Sie den Service mit folgendem Befehl:

```
> java -jar splg-grouper.jar service .\license.txt .\defs-demo.dat .\spitallisten-demo.dat
```

Indem Sie die Parameter weglassen, erhalten Sie eine Hilfestellung:

```
> java -jar .\splg-grouper.jar
splg-grouper.jar <cmd> [<arg> ...]
  version
    - Displays the version of the module
  service <licfile> <defdir/file> <sldir/file> [<hostname> <port> <backlog>]
    - Starts the REST interface of the grouper
```

Wenn Sie das Integrationsmodul Dotnet verwenden, starten Sie den Service mit folgendem Befehl:

```
.\service.exe start .\license.txt .\defs-demo.dat .\spitallisten-demo.dat
```

Indem Sie die Parameter weglassen, erhalten Sie eine Hilfestellung:

```
> service.exe
service <cmd> [<arg> ...]
  version
    - Displays the version of the module
  start <licfile> <defdir> <sldir> [<hostname> <port> <backlog>]
    - Starts the REST interface of the grouper
```

Requests

In diesem Abschnitt werden die möglichen Requests dokumentiert.

Sowohl Eingabe als auch Ausgabe der Requests erfolgen grundsätzlich als JSON. Die Ausgabe ist nicht formatiert. In den Beispielen erfolgte die Formatierung mittels des «jq» Tools um die Lesbarkeit zu erhöhen.

GET /version

Liefert die Version des Service.

Beispiel:

```
> curl -s http://127.0.0.1:8080/version -X GET|jq .
{
  "status": 200,
  "text": "OK",
  "result": {
    "version": "2024.5.6"
  }
}
```

POST /group/<Release>

Gruppirt die Daten mit dem spezifizierten Release und liefert das Gruppierungsergebnis.

Beispiel:

```
> curl -s http://127.0.0.1:8080/group/A_2022
-H "Content-Type: application/json"
-X POST
-d '{"behandlungen": [{"code": "815121"}]}'|jq .
{
  "status": 200,
  "text": "OK",
  "result": {
    "splg": "BEW7.1.1",
    "quer": [],
    "mfzs": [
      "BEW7.1.1"
    ],
    "mfzo": [
      "BEW7.1.1"
    ],
    "points": [],
    "lactrl": 99,
    "lactrlcodes": [],
    "notes": "",
    "errorcode": "w31",
    "configuration": {
      "defversion": "A_2022_8",
      "spitallisten": [
        "A_2022_3_zh.json"
      ]
    }
  }
}
```

In diesem Beispiel wurde ein Fall gruppiert, von dem nur eine einzelne Behandlung definiert ist. Trotzdem wurde erfolgreich gruppiert und die SPLG «BEW7.1.1» ausgegeben.

Beachten Sie bitte, dass in der Ausgabe auch die «configuration» angegeben wird. So können Sie sehen, mit welcher Definition und welchen Spitalisten der Fall gruppiert wurde.

Falldaten-JSON

Die zu gruppierenden Falldaten werden als JSON dem Service übermittelt. Das Format der Datenstruktur ist wie folgt:

```
{
  "burnr": "12345678",
  "agey": "45",
  "aged": "0",
  "austritt": "20230227",
  ...weitere Felder gemäss abstraktem Format...
  "diagnosen": [
    {
      "code": "D100",
      "zusatz": "",
      "seitigkeit": "",
      "rang": 0
    },
    {
      "code": "D200",
      "seitigkeit": "",
      "rang": 1
    }
    ...weitere Diagnosen...
  ],
  "behandlungen": [
    {
      "code": "815121",
      "seitigkeit": "",
      "ambext": "",
      "beginn": "2023022408",
      "rang": 0,
      "opereure": [
        {
          "gln": "7601234567890",
          "funktion": "1",
          "zulassung": "1"
        }
        ...ggf. ein zweiter Operateur...
      ]
    }
    ...weitere Behandlungen...
  ],
  "bewegungen": [
    {
      "beginn": "20230224",
```

```
    "ende": "20230227",  
    "art": "1",  
    "burnr": "12345678"  
  }  
]  
}
```

Alle Felder sind grundsätzlich optional und können weggelassen werden. Damit korrekt gruppiert werden kann, müssen natürlich gewisse Angaben vorhanden sein (Gruppierungsrelevante Felder wie Diagnosen, Behandlungen, Alter, etc.). Grundsätzlich können alle Felder gemäss abstraktem Format direkt in der Hauptstruktur verwendet werden.

Bis auf die technischen Rang-Felder enthalten alle Felder grundsätzlich String-Werte. Leere Felder werden mit einem Leer-String gekennzeichnet oder gleich ganz weggelassen.