

SPLG-Grouper Integrationsmodul Java

Einführung

Dieses Dokument beschreibt, wie Sie das Integrationsmodul Java in eine Java-Applikation direkt integrieren können. Für .NET-basierte Anwendungen gibt es ein analoges Integrationsmodul Dotnet. Für andere Technologien müssen Sie entweder eine Java- oder .NET-Integrationstechnologie verwenden, oder Sie nutzen den REST-Service, welcher bei beiden Integrationsmodulen verfügbar ist (siehe Dokument «SPLG-Grouper REST-Schnittstelle»).

Das Integrationsmodul Java setzt mindestens Java 11 voraus.

Referenzierung

Stellen Sie sicher, dass «splg-grouper.jar» im Classpath Ihrer Applikation enthalten ist.

Verwendungsbeispiel

Der folgende Code ist ein kleines Beispiel, wie Sie den Grouper instanziiieren und verwenden.

```
import java.io.File;
import splg.Grouper;
import splg.Falldaten;

public class Main {
    public static void main(String[] args) throws Exception {
        // Initialize Grouper
        Grouper grouper = Grouper.create(
            new File("config/license.txt")
        );
        grouper.setDefdir(new File("config/defs-demo.dat"));
        grouper.setSldir(new File("config/spitallisten-demo.dat"));

        // Set currently used release (definition/spitalliste)
        var release = "A_2025";
        grouper.setRelease(release);

        // Get Falldaten (from file or UI or database)
        var falldaten = new Falldaten();

        // Group falldaten and get result
        var result = grouper.group(falldaten);

        // Output result (to file or UI or database or console)
        System.out.println(result.splg);

        // Optional: Create a testcase for debug purposes
        grouper.writeTestcaseWithLog("testcase.txt");
    }
}
```

Grouper

Die Grouper-Klasse wird verwendet, um Definitionen und Spitallisten zu laden und Daten zu gruppieren. Sie stellt den zentralen Punkt der Interaktion mit dem SPLG-Grouper dar.

```
public class Grouper {  
    // Erstellt die Grouperinstanz, die Lizenzdatei «licfile» muss  
    // existieren und gültig sein, damit gruppiert werden kann.  
    public static Grouper create(File licfile) throws IOException;  
  
    // Setzt das Verzeichnis, in welchem nach Definitionsdateien  
    // gesucht wird. Es kann nur ein Verzeichnis respektive eine  
    // Datei für Definitionen gesetzt werden. Erneute Aufrufe der  
    // Funktion respektive von «setDeffile» ersetzen frühere Werte.  
    public void setDefdir(File defdir);  
  
    // Setzt die («Single-File-Def») Datei, welche für Definitionen  
    // durchsucht wird. Es kann nur ein Verzeichnis respektive eine  
    // Datei für Definitionen gesetzt werden. Erneute Aufrufe der  
    // Funktion respektive von «setDeffile» ersetzen frühere Werte.  
    public void setDeffile(File deffile);  
  
    // Setzt das Verzeichnis, in welchem nach Spitallisten gesucht  
    // wird. Es können mehrere Verzeichnisse/Dateien durch  
    // wiederholten Aufruf der Funktion gesetzt werden.  
    public void setSldir(File sldir);  
  
    // Setzt die («Single-File-SL») Datei, welche für Spitallisten  
    // durchsucht wird. Es können mehrere Dateien/Verzeichnisse  
    // durch wiederholten Aufruf der Funktion gesetzt werden.  
    public void setSlfile(File slfile);  
  
    // Fügt eine Spitalliste manuell hinzu. Die Spitalliste gilt für  
    // den spezifizierten Release «release» und betrifft den durch  
    // «burnr» und optional «plz» und «location» identifizierten  
    // Betriebsstandort (falls nur «burnr» geliefert wird, werden  
    // für «plz» und «location» leere Strings übergeben). Die Liste  
    // der Leistungsaufträge «las» bestimmt die dem Betriebsstandort  
    // erteilten Leistungsaufträge.  
    public void addSpitalliste(  
        String release,  
        String burnr,  
        String plz,  
        String location,  
        boolean reference,  
        String canton,  
        List<LA> las
```

```
);

// Fügt eine Spitalliste manuell hinzu. Die Spitalliste gilt für
// den spezifizierten Release «release» und betrifft den durch
// «burnr» und optional «plz» und «location» identifizierten
// Betriebsstandort (falls nur «burnr» geliefert wird, werden
// für «plz» und «location» leere Strings übergeben). Die Liste
// der SPLG «splgs» bestimmt die dem Betriebsstandort erteilten
// Leistungsaufträge. Achtung, da hier nur die SPLG spezifiziert
// werden, können keine Befristungen angegeben werden. Dafür
// nutzen Sie die «addSpitalliste»-Method, welche eine Liste von
// LA-Objekten akzeptiert (siehe oben).
public void addSpitalliste(
    String release,
    String burnr,
    String plz,
    String location,
    boolean reference,
    String canton,
    String... splgs
);

// Setzt einen Kantons-Override. Der Kantonsoverride
// muss ein Kantonskürzel sein (z. B. «ZH»). Dieses
// wird ab nun für alle Fälle beim Leistungscontrolling
// als Wohnkanton verwendet (bei der Auswahl der zu
// verwendenden Spitalliste). Ein gesetzter Override
// kann durch erneuten Aufruf mit leerem String zurück-
// gesetzt werden.
public void setKantonOverride(String kantonOverride);

// Aktiviert einen Release (z. B. «A_2024» oder «R_2024»)
// Ab diesem Zeitpunkt werden alle folgenden Fälle mit
// diesem Release gruppiert. Es kann jederzeit ein neuer
// oder bereits früher gewählter Release gesetzt werden.
// Die Definitionen und Spitallisten werden gecacht.
public void setRelease(String release) throws IOException;

// Löscht gecachte Definitionen und Spitallisten.
public void clearCache();

// Löscht konfigurierte sowie gecachte Definitionen.
public void clearDefinitions();
```

```
// Löscht konfigurierte sowie gecachte Spitallisten.
public void clearSpitallisten();

// Gruppiert einen Fall
public Result group(Falldaten fall);

// Für Test- und Analysezwecke können detaillierte
// Informationen zum letzten gruppierten Fall abgefragt
// werden.
public List<String> getDetailsLog();

// Für Testzwecke können Falldaten und Result als «Testcase»
// formatiert zurückgegeben respektive in eine Datei geschrieben
// werden. Siehe «SPLG-Grouper Testcase» Dokument.

public String getTestcase(
    Falldaten fall
);

public String getTestcase(
    Falldaten fall,
    Result result
);

public String getTestcaseWithLog(
    Falldaten fall,
    Result result
);

public void writeTestcase(
    Falldaten fall,
    String destfile
) throws IOException;

public void writeTestcase(
    Falldaten fall,
    Result result,
    String destfile
) throws IOException;

public void writeTestcaseWithLog(
    Falldaten fall,
    Result result,
    String destfile
) throws IOException;
```

```
}  
  
public class LA {  
    // Erstellt einen unbefristeten Leistungsauftrag für «splg»  
    public LA(String splg);  
  
    // Erstellt einen auf das Jahr «year» befristeten  
    // Leistungsauftrag für «splg»  
    public LA(String splg, int year);  
  
    // Erstellt einen auf den Bereich «validFrom» bis «validTo»  
    // befristeten Leistungsauftrag für «splg»  
    public LA(String splg, Date validFrom, Date validTo);  
}
```

Falldaten

```
public class Falldaten {

    // controlling-relevant
    public String burnr = "";
    public String plz = "";
    public String standort = "";
    public String wohnkanton = "";
    public String statistikfall = "";
    public String behandlungsart = "";
    public String tarifsystem = "";

    // fallidentifikation
    public String fallid = "";

    // grouper-relevant
    public String agey = "";
    public String aged = "";
    public String ssw = "";
    public String ggw = "";
    public String dmb = "";
    public String freiwilligkeit = "";
    public String austritt = ""; // Austrittsdatum, Format YYYYMMDD

    // grouper-relevant
    public List<FalldatenDiagnose> diagnosen = new ArrayList<>();
    public List<FalldatenBehandlung> behandlungen = new ArrayList<>();

    // controlling-relevant

    public List<FalldatenPatientenbewegung> bewegungen =
        new ArrayList<>();

    // output-relevant
    public String ave = "";
    public String weg = "";
    public String sn = "";
    public String skz = "";
    public String ea = "";
    public String ad = "";
```

```
    public String ana = "";
    public String ed = "";
    public String mk = "";
    public String ei = "";
    public String gew = "";
    public String hktr = "";

    // output-relevant SwissDRG/TARPSY/STReha
    public String drg = "";
    public String pcg = "";
    public String rcg = "";
    public String mdc = "";
    public String cw = "";
    public String pccl = "";
    public String ecwt = "";

    // zusatz (durchreiche)
    public String zusatz = "";
    public String erhebungsjahr = "";
    public String standortkanton = "";
}

public class FalldatenDiagnose {
    public int rang;
    public String code;
    public String seitigkeit = "";
    public String zusatz = "";
}

public class FalldatenBehandlung {
    public int rang;
    public String code;
    public String seitigkeit = "";
    public String ambExt = "";
    public String beginn = "";
    public List<FalldatenOperateur> operateure = null;
}

public class FalldatenOperateur {
    public String gln = "";
    // 1 = Erstoperateur, 2 = Zweitoperateur
    public String funktion = "";
    // 1 = Auf Liste GD, 0 = Nicht auf Liste GD
```

```
        public String zulassung = "";
    }

    public class FalldatenPatientenbewegung {
        public String beginn = "";
        public String ende = "";
        public String art = "";
        public String burnr = "";
    }
```

Resultat

```
public class Result {
    public String fallid;

    public String splg;
    public List<String> lgs;
    public List<String> quer;
    public List<String> mfzs;
    public List<String> mfzo;

    public List<MFZOPoints> points;

    public int slst;
    public int lactrl;
    public Set<String> lactrlcodes;

    public Set<ErrorCode> errorcode;

    public String notes = "";
    public String zusatz = "";

    public String defversion;
    public List<String> spitallisten;

    public String getErrorCode();

    public String getMFZOString();
}

public class MFZOPoints {
    public String gln;
    public String lg;
    public double points;
}
```



```
public class ErrorCode implements Comparable<ErrorCode> {  
    public String getCode();  
    public boolean isOK();  
    public boolean isWarning();  
    public boolean isError();  
}
```

REST-Service

Das Integrationsmodul Java implementiert im «splg-grouper.jar» auch einen einfachen REST-Service, mit dem Sie technologieunabhängig gruppieren können.

Sie starten den Service auf der Kommandozeile wie folgt:

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

Dadurch wird per Default unter Port 8080 auf localhost ein HTTP-Server gestartet, über welchen gruppiert werden kann. Weitere Informationen dazu finden Sie im Dokument «SPLG-Grouper REST-Schnittstelle».

Testcase

Das Integrationsmodul Java implementiert im «splg-grouper.jar» auch die Möglichkeit auf einfache Art und Weise Testfälle zu erstellen und zu analysieren. Weitere Informationen dazu finden Sie im Dokument «SPLG-Grouper Testcase».