### クエリコマンドパーサの実装

本節では、[SampleQueryCommandParser.java](https://github.com/logpresso/logpresso-app-examples/blob/main/logpresso-sample-app/src/main/java/com/logpresso/sonar/sample/query/SampleQueryCommandParser.java) および [SampleSubnetGroupsCommandParser.java](https://github.com/logpresso/logpresso-app-examples/blob/main/logpresso-sample-app/src/main/java/com/logpresso/sonar/sample/query/SampleSubnetGroupsCommandParser.java) のコードについて説明します。

#### クエリコマンドパーサの概要

ログプレッソ・ソナーのクエリは、パイプで区切られたクエリコマンドの組み合わせで構成されます。クエリエンジンは、クエリ文字列をパイプ記号で分割し、空白で区切られた最初のトークンをコマンド名として認識します。クエリエンジンはコマンド名をもとに QueryCommandParser オブジェクトを検索し、コマンドのパースを依頼します。

クエリコマンドパーサは QueryCommandParser インターフェースを実装します。

public interface QueryCommandParser { String getCommandName(); QueryCommand parse(QueryContext context, String commandString); Map<String, QueryErrorMessage> getErrorMessages(); QueryCommandHelp getCommandHelp(); QueryParserService getQueryParserService(); void setQueryParserService(QueryParserService queryParserService);}

各メソッドの機能は以下の通りです。

* getCommandName(): ログプレッソ・ソナー プラットフォーム上で一意となるクエリコマンド名を返します。コマンド名はクエリエディタで自動的にシンタックスハイライトされます。
* parse(): クエリコマンドの構文をパースし、QueryCommand オブジェクトを返します。
* getErrorMessages(): クエリパース中に発生するエラーコードに対するローカライズ済みメッセージを提供します。
* getCommandHelp(): コマンドの説明、オプション、出力フィールドの詳細を提供します。本メソッドにはデフォルト実装が用意されています。
* getQueryParserService(): クエリパーササービスオブジェクトを返します。パーサ内でサブクエリなど再帰的なパースが必要な場合に利用します。
* setQueryParserService(): クエリパーササービスオブジェクトを設定します。ユニットテスト以外で本メソッドを直接呼び出すことは推奨されません。

#### 接続プロファイル対応クエリコマンドパーサ

アプリで拡張するクエリコマンドの多くは外部システムと連携するため、接続プロファイルを利用します。接続プロファイルを利用するクエリコマンドパーサは、ConnectProfileQueryCommandParser クラスを継承することで簡単に実装できます。SampleQueryCommandParser は、アプリで実装する複数のコマンドパーサが継承するための基本実装を提供します。

public abstract class SampleQueryCommandParser extends ConnectProfileQueryCommandParser { protected static final String ERR\_SERVICE\_UNAVAILABLE = "204000"; protected static final String ERR\_PROFILE\_REQUIRED = "204001"; protected static final String ERR\_NAME\_REQUIRED = "204002"; public SampleQueryCommandParser() { super("sample", ERR\_SERVICE\_UNAVAILABLE, ERR\_PROFILE\_REQUIRED); }}

コンストラクタのパラメータは以下の通りです。

* profileType: コマンドが利用する接続プロファイル種別の識別子です。ConnectProfileFactory の getType() の戻り値と一致する必要があります。
* emptyProfileErrorCode: 現在のセッションで利用可能な接続プロファイルが一つも存在しない場合に発生させるエラーコードです。
* missingProfileErrorCode: コマンドの profile パラメータが必須で、かつ profile 値が指定されていない場合に発生させるエラーコードです。

protected List<String> getSupportedOptions() { return new ArrayList<String>(getCommandHelp().getOptions().keySet());}

クエリコマンドは COMMAND-NAME KEY1=VALUE1 KEY2=VALUE2 の形式で定義されます。getSupportedOptions() は利用可能なパラメータの一覧を返します。この実装では、ヘルプに定義されたすべてのパラメータを返すようになっています。

protected abstract QueryCommand parse(QueryContext context, SampleParams params);

SampleQueryCommandParser を継承するコマンドパーサは、既にパース済みのパラメータが SampleParams として渡されます。各コマンドパーサは、必須パラメータの検証など追加のバリデーションを実施します。

protected ConnectProfileParams parseParams(QueryContext context, Map<String, String> opts) { SampleParams params = new SampleParams(); params.setName(opts.get("name")); return params;}protected QueryCommand parse(QueryContext context, ConnectProfileParams params, String commandString) { return parse(context, (SampleParams) params);}

* parseParams(): すべての拡張クエリコマンドのパラメータを一括でパースします。個別コマンドごとにパラメータをパースすることも可能ですが、共通パラメータのパースを一箇所にまとめることで管理が容易になります。
* parse(): ConnectProfileQueryCommandParser クラスは、コマンド構文からキー=値形式のパラメータをパースしてマップ化し、parseParams() を呼び出します。その戻り値である ConnectProfileParams オブジェクトを再度 parse() メソッドに渡します。既にパース済みのパラメータを利用するだけで済みます。

public Map<String, QueryErrorMessage> getErrorMessages() { Map<String, QueryErrorMessage> errors = new HashMap<>(); errors.put(ERR\_SERVICE\_UNAVAILABLE, newMsg("No available sample profile found.", "使用可能なサンプルプロファイルがありません。")); errors.put(ERR\_PROFILE\_REQUIRED, newMsg("Specify valid sample profile.", "サンプルプロファイル名を入力してください。")); errors.put(ERR\_NAME\_REQUIRED, newMsg("Specify name option in the sample-create-subnet-group command.", "sample-create-subnet-group コマンドで name オプションを指定してください。")); return errors;}

getErrorMessages() はエラーコードとローカライズ済みメッセージのペアを返します。コマンドパーサは QueryParseException 例外をスローする際にエラーコードを指定します。ログプレッソ・ソナーのクエリエンジンは、エラーコードをローカライズメッセージに変換する際にこのメソッドの戻り値を参照します。

#### クエリコマンドパーサの実装

SampleSubnetGroupsCommandParser クラスは、前述の SampleQueryCommandParser を継承しています。

public class SampleSubnetGroupsCommandParser extends SampleQueryCommandParser { public String getCommandName() { return "sample-subnet-groups"; } protected QueryCommand parse(QueryContext context, SampleParams params) { return new SampleSubnetGroupsCommand(params); }}

このパーサは sample-subnet-groups というコマンド名を定義しており、追加パラメータがないため、parse() では特別な処理を行わずに SampleSubnetGroupsCommand オブジェクトを生成して返します。

本パーサで注目すべきは、コンストラクタで設定するメタデータです。

setDescription(Locale.ENGLISH, "Get subnet groups from the Logpresso server.");setOutput("guid", ValueType.STRING, Locale.ENGLISH, "GUID", "");

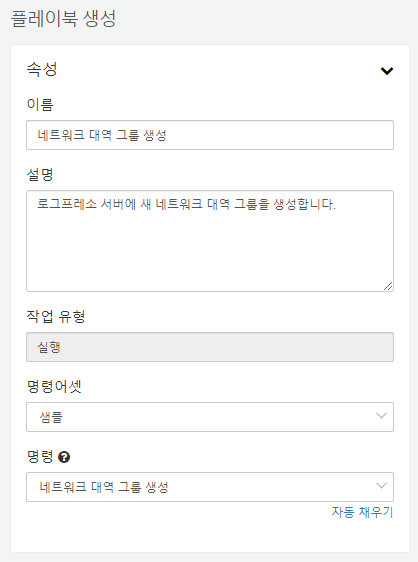
* setDescription(): クエリヘルプに表示するロケールごとのコマンド説明を設定します。
* setOutput(): クエリヘルプに表示する出力フィールドの名称、型、説明を設定します。

SampleCreateSubnetGroupCommandParser クラスでは、さらに別のメタデータも設定しています。

setDisplayGroup(Locale.KOREAN, "샘플");setDisplayName(Locale.KOREAN, "네트워크 대역 그룹 생성");setDescription(Locale.KOREAN, "로그프레소 서버에 새 네트워크 대역 그룹을 생성합니다.");setOption("name", REQUIRED, Locale.KOREAN, "이름", "새 네트워크 대역 그룹 이름");

* setDisplayGroup(): クエリコマンドのグループ。プレイブックのコマンドセット項目にマッピングされます。
* setDisplayName(): クエリコマンドの表示名。プレイブックのコマンド項目にマッピングされます。
* setOption(): コマンドパラメータを定義します。Ctrl+Space 入力時のパラメータヘルプや自動補完に利用されます。

例えば、プレイブックデザイナーで実行アクションを作成する際に「샘플」コマンドセットを選択すると、下図のように「네트워크 대역 그룹 생성」項目が表示されます。



クエリコマンドパース中の例外は、以下のように発生させます。create-subnet-group コマンドは name パラメータが必須ですが、これが指定されていない場合は QueryParseException をスローし、パースを失敗させます。

protected QueryCommand parse(QueryContext context, SampleParams params) { if (params.getName() == null) throw new QueryParseException(ERR\_NAME\_REQUIRED, -1); return new SampleCreateSubnetGroupCommand(params);}

次節では、このように実装したクエリコマンドパーサの登録方法について解説します。