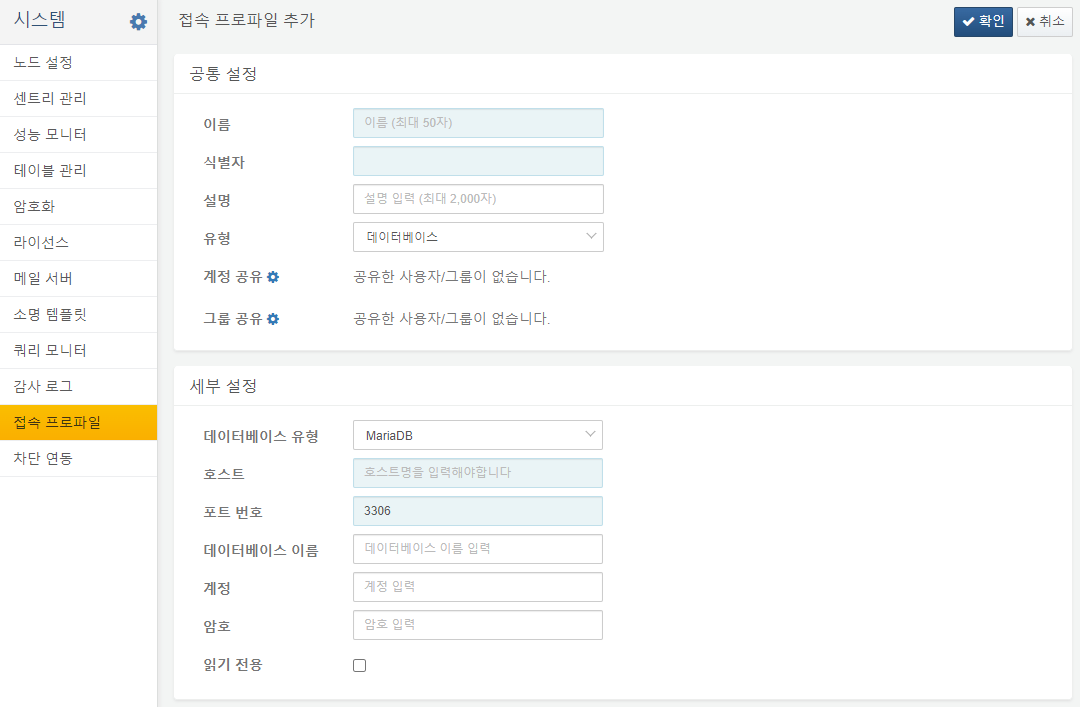
### 接続プロファイルの実装

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

#### 接続プロファイルの概要

ログプレッソ・ソナーのクエリコマンドやロガーは、外部システムと通信する場合が多くあります。例えば、[dbquery](https://docs.logpresso.comnull) コマンドは外部データベースでSQLを実行した結果を取得します。外部データベースに接続するには、JDBC接続文字列、ユーザー、パスワードが必須です。そのため、dbquery クエリコマンドは接続プロファイルを PROFILE パラメータとして受け取ります。dbquery クエリコマンドを実行するには、以下のようにデータベース接続プロファイルを事前に設定しておく必要があります。



なお、接続プロファイルが設定されていても、誰でもデータベースにSQLクエリを実行できるようにしてはいけません。ユーザーまたはユーザーグループに接続プロファイルの利用権限を付与することで、該当ユーザーのみが接続プロファイルを利用して外部システムと通信できるように制御します。

データベースに限らず、ほとんどの外部システムは認証を要求します。本アプリのサンプルでは、ログプレッソ・ソナーのREST APIを呼び出しますが、REST APIを利用するには Authorization ヘッダーにAPIキーを指定する必要があります。

外部システムが要求する認証方式によって設定項目は異なるため、アプリは専用のConnectProfileFactoryを実装し、ログプレッソ・ソナーのプラットフォーム機能を拡張する必要があります。

#### ConnectProfileFactoryプラグイン

サンプルアプリでは、SampleConnectProfileFactoryを実装し、ログプレッソ・ソナーのプラットフォームに登録します。

@Component(name = "sample-connect-profile-factory")@Instantiatepublic class SampleConnectProfileFactory extends AbstractConnectProfileFactory { @Requires private ConnectProfileFactoryRegistry connectProfileFactoryRegistry; @Validate public void start() { connectProfileFactoryRegistry.addFactory(this); } @Invalidate public void stop() { if (connectProfileFactoryRegistry != null) connectProfileFactoryRegistry.removeFactory(this); }}

上記のコードは、AbstractConnectProfileFactory を継承した SampleConnectProfileFactory コンポーネントを定義しています。コンポーネントの起動時に ConnectProfileFactoryRegistry OSGiサービスの addFactory() を呼び出して自身を登録し、コンポーネントの停止時には removeFactory() を呼び出して自身を削除します。

@Component、@Instantiate、@Requires、@Validate、@Invalidate はすべてiPOJOアノテーションです。各アノテーションの説明は以下の通りです。

* @Component: このクラスがiPOJOコンポーネントであることを宣言します。iPOJOコンポーネントには一意の名前を指定する必要があります。
* @Instantiate: iPOJOコンポーネントインスタンスの生成を指示します。@Instantiateが宣言されていない場合、アプリバンドルを起動してもコンポーネントインスタンスは生成されません。
* @Requires: サービスインターフェースへの依存関係を宣言します。明示的な代入文はありませんが、コンポーネントの起動時にiPOJOフレームワークが自動的にConnectProfileFactoryRegistryインターフェースを提供するOSGiサービスオブジェクトを注入します。これをinjected POJO（Plain Old Java Object）と呼び、ここからiPOJOというフレームワーク名が生まれました。
* @Validate: コンポーネントの起動時に呼び出されるコールバックを宣言します。@Requiresで宣言されたすべての依存関係が満たされて初めてコンポーネントが起動します。
* @Invalidate: コンポーネントの停止時に呼び出されるコールバックを宣言します。@Requiresで宣言された依存関係のいずれかが満たされなくなると、コンポーネントは停止します。OSGiバンドルを停止または削除すると、サービスが削除され、それに依存していたコンポーネントも停止します。

#### ConnectProfileFactoryの実装

**識別子**

getType() は、ConnectProfileFactoryRegistry に登録された各 ConnectProfileFactory オブジェクトを識別するためのIDを返します。一般的に小文字のアルファベットとハイフンで定義します。

public String getType() { return "sample";}

**表示名**

getDisplayName() は、ユーザーインターフェースに表示する接続プロファイルタイプの名称を返します。ロケールに応じて適切な文字列を返す必要があります。

public String getDisplayName(Locale locale) { if (locale != null && locale.equals(Locale.KOREAN)) return "샘플"; return "Sample";}

**説明**

getDescription() は、ユーザーインターフェースに表示する接続プロファイルタイプの説明を返します。ロケールに応じて適切な文字列を返す必要があります。

public String getDescription(Locale locale) { if (locale != null && locale.equals(Locale.KOREAN)) return "샘플 접속 설정을 관리합니다."; return "Manage connection properties of sample app.";}

**保護設定項目リスト**

getProtectedConfigKeys() は、ユーザーインターフェースに表示してはならない設定項目の識別子リストを返します。例えば、パスワードやAPIキーは管理者であっても画面上で参照できないようにする必要があります。保護された設定項目は再設定のみ可能で、参照はできません。また、ログプレッソ・ソナーのプラットフォーム内でも暗号化して保存されます。

public Set<String> getProtectedConfigKeys() { return Set.of("api\_key");}

**設定仕様リスト**

getConfigOptions() は、接続プロファイルの設定フォームを定義します。接続プロファイルの設定仕様ですが、LoggerConfigOption 型を使用します。これはロガーの設定フォームを定義する際にも同じ型を用いて動的に設定フォームを定義するためです。

public List<LoggerConfigOption> getConfigOptions() { LoggerConfigOption endpoint = new MutableStringConfigType("endpoint", t("Endpoint", "エンドポイント"), t("REST API URL", "REST APIアドレス"), true); LoggerConfigOption apiKey = new MutableStringConfigType("api\_key", t("API key", "APIキー"), t("GUID format", "GUID形式"), true); return Arrays.asList(endpoint, apiKey);}

利用可能な型は以下の3種類です。

* MutableStringConfigType: 変更可能な文字列設定項目
* MutableIntegerConfigType: 変更可能な32ビット整数設定項目
* MutableBooleanConfigType: 変更可能なブール設定項目

リストで返した設定項目は、その順序で画面に表示されます。

**設定サマリー**

接続プロファイル一覧で主要な設定内容を素早く確認できるよう、重要な設定のみを要約して返します。ロケールに応じて適切な文字列を返す必要があります。

public String describeConfigs(ConnectProfile profile, Locale locale) { String endpoint = profile.getString("endpoint"); if (locale != null && locale.equals(Locale.KOREAN)) return "エンドポイント: " + endpoint; return "Endpoint: " + endpoint;}