BL-06 データベースアクセス機能

■ 概要

◆ 機能概要

- TERASOLUNA Batch Framework for Java ver 3.x で使用するデータベースアクセス機能は、TERASOLUNA Server Framework for Java ver 2.x で使用していたデータベースアクセス機能と同一のものを利用して、データベースアクセスを行う。
- 本項目では、TERASOLUNA Batch Framework for Java ver 3.x でデータベースアクセス機能を使用する場合の TERASOLUNA Server Framework for Java ver 2.x との違いのみを説明するものとし、データベースアクセス機能の詳細な説明は別資料の「CB-01 データベースアクセス機能」の機能説明書を参照すること。

◆ コーディングポイント

- 本説明書でのコーディングポイントは、別資料の「CB-01　データベースアクセス機能」のコーディングポイントと異なる以下の項目についてのみ説明を行う。
  - SqlMapClientFactoryBean の Bean 定義
  - DAO の Bean 定義
  - QueryDAOiBatisImpl を使用した一覧データ取得例
  - UpdateDAOiBatisImpl を使用したデータ登録例
  - QueryRowHandleDAOiBatisImpl を使用したデータ取得例データソースの Bean 定義

  その他の項目については、「CB-01　データベースアクセス機能」を参照すること。

- データソースの Bean 定義
  - データソースの設定は Bean 定義ファイル beansDef/dataSource.xml に定義する。
  - Bean 定義例(beansDef/dataSource.xml)

```
<!-- DBCP のデータソースを設定する。　-->
<context:property-placeholder location="SqlMapConfig/jdbc.properties" />
<bean id="dataSource" destroy-method="close"
      class="org.apache.commons.dbcp.BasicDataSource">
      <property name="driverClassName" value="${jdbc.driver}" />
      <property name="url" value="${jdbc.url}" />
      <property name="username" value="${jdbc.username}" />
      <property name="password" value="${jdbc.password}" />
      <property name="maxActive" value="10" />
      <property name="maxIdle" value="1" />
      <property name="maxWait" value="5000" />
</bean>
```

設定値はプロパティファイルに切り離し、プレースホルダを利用して設定する。

  - Bean 定義で利用されるプロパティファイル例(SqlMapConfig/jdbc.properties)

```
#
# ジョブ管理テーブル DB 接続先
#
jdbc.driver=org.postgresql.Driver
jdbc.url=jdbc:postgresql://127.0.0.1:5432/postgres
jdbc.username=postgres
jdbc.password=postgres
```

- SqlMapClientFactoryBean の Bean 定義
  Spring で iBatis を使用する場合、SqlMapClientFactoryBean を使用して iBatis 設定ファイルの Bean 定義を DAO に設定する必要がある。
  SqlMapClientFactoryBean は、iBatis のデータアクセス時に利用されるメインのクラス「SqlMapClient」を管理する役割をもつ。
  iBatis 設定ファイルの Bean 定義はアプリケーション内で一つとする。
  - iBatis 設定ファイル
    "configLocation"プロパティに、iBatis 設定ファイルのコンテキストルートからのパスを指定する。
  - 単一データベースの場合は"dataSource"プロパティに、使用するデータソースの Bean 定義を設定する。
  - Bean 定義例(beansDef / dataSource.xml)

```
<!-- システム共通 SqlMapConfig 定義 -->
<bean id="sqlMapClient" class="org.springframework.orm.ibatis.SqlMapClientFactoryBean">
        <property name="configLocation" value="SqlMapConfig/SqlMapConfig.xml" />
        <property name="dataSource" ref="dataSource" />
</bean>
```

  - 複数のデータベースの場合は"dataSource"プロパティは指定せずに、"configLocation"プロパティのみ設定する。
  - Bean 定義例(beansDef / dataSource.xml)

```
<!-- システム共通 SqlMapConfig 定義 -->
<bean id="sqlMapClient" class="org.springframework.orm.ibatis.SqlMapClientFactoryBean">
        <property name="configLocation" value="SqlMapConfig/SqlMapConfig.xml" />
</bean>
```

- DAO の Bean 定義
  DAO の実装クラスは、beansDef/dataSource.xml に定義する
  Bean 定義時に DAO 実装クラスの"sqlMapClient"プロパティに iBatis 設定ファイルの Bean 定義を設定する必要がある。
  - Bean 定義例(beansDef / dataSource.xml)

```
<!-- システム共通 SqlMapConfig 定義 -->
<bean id="sqlMapClient" class="org.springframework.orm.ibatis.SqlMapClientFactoryBean">
        <property name="configLocation" value="SqlMapConfig/SqlMapConfig.xml" />
        <property name="dataSource" ref="dataSource" />
</bean>

<!-- 照会系の DAO 定義 -->
<bean id="queryDAO" class="jp.terasoluna.fw.dao.ibatis.QueryDAOiBatisImpl">
        <property name="sqlMapClient" ref="sqlMapClient" />
</bean>
```

  - 複数データベースの場合は"sqlMapClient"プロパティの設定だけでなく、"dataSource"プロパティに、DAO 実装クラスで使用するデータソースを指定する必要がある。
  - Bean 定義例(beansDef/dataSource.xml)

```
<!-- 照会系の DAO 定義 -->
<bean id="queryDAO" class="jp.terasoluna.fw.dao.ibatis.QueryDAOiBatisImpl">
        <property name="sqlMapClient" ref="sqlMapClient" />
        <property name="dataSource" ref=" dataSource " />
</bean>
```

- QueryDAOiBatisImpl を使用した一覧データ取得例
  TERASOLUNA Batch Framework for Java ver 3.x では、DB からのデータの取得の為に「AL-041 入力データ取得機能」を提供しており、QueryDAOiBatisImpl を使用したデータの取得を推奨していない。
  DB からのデータの取得を行う場合は、「AL-041 入力データ取得機能」の内容を参考にし、実装すること。

- UpdateDAOiBatisImpl を使用したデータ登録例
  UpdateDAOiBatisImpl を使用して、データベースに情報を登録する場合の設定およびコーディング例を以下に記述する。

① DAO 実装クラスを以下のように Bean 定義ファイルに定義する。

➢ Bean 定義例(beansDef/dataSource.xml)

```
<!-- 更新系の DAO 定義 -->
<bean id="updateDAO" class="jp.terasoluna.fw.dao.ibatis.UpdateDAOiBatisImpl">
        <property name="sqlMapClient" ref="sqlMapClient" />
</bean>
```

② ビジネスロジックを作成する。
Bean 定義ファイルにて設定された DAO の updateDAO.execute(String sqlID, Object bindParams)メソッドを使用する。
メソッドの引数に、「発行する SQLID」と「SQL に関連付けられるオブジェクト」を設定する必要がある。

➢ ビジネスロジック実装例

```
@Autowired
protected UpdateDAO updateDAO = null;

    public int execute(BLogicParam arg0) {
        ......
        updateDAO.execute("insertUser", bean);
        ......
    }
```

設定された DAO を使用して、
データベースにデータを登録する。

- QueryRowHandleDAOiBatisImpl を使用したデータ取得例
  「QueryDAOiBatisImpl を使用した一覧データ取得例」の項目でも説明したように、DB からのデータの取得には「AL-041 入力データ取得機能」を推奨している。
  「AL-041 入力データ取得機能」を使用した場合に、要件を満たせないような場合のみ、以下を参考にして QueryRowHandleDAOiBatisImpl を使用した DB からの取得を実装すること。
  (Bean 定義ファイルの定義方法は、UpdateDAOiBatisImpl と同様なため省略する)
  - DataRowHandler の実装

```
import jp.terasoluna.fw.dao.event.DataRowHandler;

public class SampleRowHandler implements DataRowHandler {
    public void handleRow(Object param) {
        if (param instanceof HogeData) {
            HogeData hogeData = (HogeData)param;
            // 一件のデータを処理するコードを記述
        }
    }
}
```

一件毎に handleRow メソッドが呼ばれ、引数に一件分のデータが格納されたオブジェクトが渡される。

一件のデータを元に更新処理を行うのであれば、あらかじめ DataRowHandler に UpdateDAO を渡しておく。
ダウンロードであれば ServletOutputStream などを渡しておくとよい。

  - ビジネスロジック実装例

```
@Autowired
protected QueryRowHandleDAO queryRowHandleDAO = null;

    public int execute(BLogicParam arg0) {
        Parameter param = new Parameter();
        SampleRowHandler dataRowHandler = new SampleRowHandler();

        queryRowHandleDAO.executeWithRowHandler(
                "selectDataSql", param, dataRowHandler);

        // 終了コードの返却
        return 0;
    }
...(以下略)
```

実際に一件ずつ処理を行う DataRowHandler インスタンスを渡す。

※ TERASOLUNA Batch Framework for Java ver 3.x においてデータベースアクセス機能を使用する場合の注意点。
先の実装例にも掲載した通り、「CB-01 データベースアクセス機能」と異なり、ビジネスロジックの Bean 定義を行う必要はない。
TERASOLUNA Batch Framework for Java ver 3.x では、アノテーション(@Autowired や@Component)を利用してビジネスロジックと DAO の DI を行う。

◆ 拡張ポイント

なし。

■ 構成クラス

	クラス名	概要
1	jp.terasoluna.fw.dao.QueryDAO	参照系 SQL を実行するための DAO インタフェース
2	jp.terasoluna.fw.dao.UpdateDAO	更新系 SQL を実行するための DAO インタフェース
3	jp.terasoluna.fw.dao.StoredProcedureDAO	StoredProcedure を実行するための DAO インタフェース
4	jp.terasoluna.fw.dao.QueryRowHandleDAO	参照系 SQL を実行し一件ずつ処理するための DAO インタフェース
5	jp.terasoluna.fw.dao.ibatis.QueryDAOiBatisImpl	QueryDAO インタフェースの iBATIS 用実装クラス
6	jp.terasoluna.fw.dao.ibatis.UpdateDAOiBatisImpl	UpdateDAO インタフェースの iBATIS 用実装クラス
7	jp.terasoluna.fw.dao.ibatis.StoredProcedureDAOiBatisImpl	StoredProcedureDAO インタフェースの iBATIS 用実装クラス
8	jp.terasoluna.fw.dao.ibatis.QueryRowHandleDAOiBatisImpl	QueryRowHandleDAO インタフェースの iBATIS 用実装クラス

■ 関連機能

- なし

■ 使用例

- 機能網羅サンプル(terasoluna-batch-functionsample)
- チュートリアル(terasoluna-batch-tutorial)