
富士通 NetCOBOL for .NET

Retrieve Classes for .NET を使用するサンプルプログラム

2021 年 6 月

株式会社エージーテック

富士通 NetCOBOL for .NET から Btrieve Classes for .NET を使用するサンプルについて説明します。

Btrieve Classes for .NET の本サンプルでは、**DDF** クラスおよび **Record** クラスを使用しています。

■ サンプルの概要

このサンプルでは、`demodata` の `Person.mkd` を開き、読み込みおよび更新を行います。

「OPEN」ボタンのクリック処理で DDF クラス、Record クラス の生成およびファイルのオープンを行います。

「Get First」 ボタンのクリック処理で先頭のレコードを読み込み、データを表示します。

「Next」ボタンのクリック処理で2番目以降のレコードを読み込み、データを表示します。

「Update」 ボタンのクリック処理でデータの更新を行います。

「Close」 ボタンでファイルのクローズを行います。

■ サンプルプログラムの動作環境

サンプルプログラムは、次の環境で動作します。

- Windows 10
- NetCOBOL Standard Edition 開発パッケージ for .NET V8.0
- Microsoft Visual Studio 2017
- Btrieve Classes for .NET v8.0
- Actian Zen v14 (14.20.014)
- .NET Framework 4.5

■処理の詳細

1. OPEN 処理

この処理では、DDF クラス、Record クラスを生成しファイルのオープンを行います。

* DDF を指定して BCN DDF クラスを生成

```
INVOKE BtLib-DDF "NEW" USING BY VALUE "btrv:///demodata" RETURNING BtLib1.
```

```
INVOKE BtLib1 "GetRecord" USING BY VALUE "Person" RETURNING BtLib2.
```

```
INVOKE BtLib2 "Open" RETURNING RC.
```

* 読み込むインデックスに"PersonID"を指定

```
SET PROP-INDEX OF BtLib2 TO "PersonID".
```

まず、DDF クラスで辞書ファイル (.DDF) を開きます。

クラスの生成には、INVOKE 文を使用します。

GetRecord メソッドに"Person" を指定し、Record クラスを生成します。

メソッドの実行には、INVOKE 文を使用します。

Record クラスの OPEN メソッドで Person.mkd ファイルをオープンします。

読み込みを行う際のインデックスを、Record クラスの Index プロパティに設定します。

プロパティへ値を設定する際には、SET 文を使用します。

SET 文では、直接値を指定できないため、

```
PROPERTY PROP-INDEX AS "Index"
```

のように定義しておきます。

2. Get First 処理

この処理では、先頭のレコードを読み込み、データを表示します。

```
* 先頭レコードの読み込み
    SET B-GET-FIRST TO GetFirst OF BtLib-OP.
    INVOKE BtLib2 "Read" USING BY VALUE B-GET-FIRST RETURNING RC.
* 戻り値を判定
    if RC = 0 then
* First_Name フィールドの内容を textbox1 に代入
        INVOKE BtLib2 "get_Item" USING N"First_Name" RETURNING tmpObj
        INVOKE tmpObj "ToString" RETURNING STRBUF2
        SET PROP-TEXT OF textBox1 TO STRBUF2
* Last_Name フィールドの内容を textbox2 に代入
        INVOKE BtLib2 "get_Item" USING N"Last_Name" RETURNING tmpObj
        INVOKE tmpObj "ToString" RETURNING STRBUF2
        SET PROP-TEXT OF textBox2 TO STRBUF2
* City フィールドの内容を textbox3 に代入
        INVOKE BtLib2 "get_Item" USING N"City" RETURNING tmpObj
        INVOKE tmpObj "ToString" RETURNING STRBUF2
        SET PROP-TEXT OF textBox3 TO STRBUF2
    else
* 戻り値を label3 に表示
        move RC to RCStr
        string "RC = " RCStr delimited by size INTO msgBUF
        SET PROP-TEXT OF label3 to msgBUF
    end-if
```

Record クラスの Read メソッドを実行し、先頭レコードを読み込みます。このとき、引数にオペレーションコード (Get First) を指定します。

戻り値が 0 であれば、Textbox に読み込んだ内容を表示し、0 以外であれば、Label にエラーを表示します。

読み込んだ内容を取り出すには、Record クラスの get_Item メソッドを使用し、引数にフィールド名を指定します。*

値は、オブジェクト型の変数で受け取ります。

Textbox に値を設定する前に、ToString メソッドで文字列に変換する必要があります。

Textbox への値の設定は、set 文を使い text プロパティに代入します。

* Record クラスの説明には、get_Item メソッドは記載されていませんが、Record クラスが System.Collections 名前空間の Hashtable クラスから派生していることから、Item オブジェクトが自動生成されています。

3. Next 処理

この処理では、2 番目以降のレコードを読み込み、データを表示します。

```
* 次のレコード読み込み
    SET B-GET-NEXT TO GetNext OF BtLib-OP.
    INVOKE BtLib2 "Read" USING BY VALUE B-GET-NEXT RETURNING RC.
* 戻り値を判定
    if RC = 0 then
* First_Name フィールドの内容を textbox1 に代入
        INVOKE BtLib2 "get_Item" USING N"First_Name" RETURNING tmpObj
        INVOKE tmpObj "ToString" RETURNING STRBUF2
        SET PROP-TEXT OF textBox1 TO STRBUF2
* Last_Name フィールドの内容を textbox2 に代入
        INVOKE BtLib2 "get_Item" USING N"Last_Name" RETURNING tmpObj
        INVOKE tmpObj "ToString" RETURNING STRBUF2
        SET PROP-TEXT OF textBox2 TO STRBUF2
* City フィールドの内容を textbox3 に代入
        INVOKE BtLib2 "get_Item" USING N"City" RETURNING tmpObj
        INVOKE tmpObj "ToString" RETURNING STRBUF2
        SET PROP-TEXT OF textBox3 TO STRBUF2
    else
* 戻り値を label3 に表示
        move RC to RCStr
        string "RC = " RCStr delimited by size INTO msgBUF
        SET PROP-TEXT OF label3 to msgBUF
    end-if
```

Record クラスの Read メソッドを実行し次のレコードを読み込みます。このとき、引数にオペレーションコード (Get Next) を指定します。

戻り値が 0 であれば、Textbox に読み込んだ内容を表示し、0 以外であれば、Label にエラーを表示します。

読み込んだ内容を取り出すには、Record クラスの get_Item (*1) メソッドを使用し、引数にフィールド名を指定します。

値は、オブジェクト型の変数で受け取ります。

Textbox に値を設定する前に、ToString メソッドで文字列に変換する必要があります。

Textbox への値の設定は、set 文を使い text プロパティに代入します。

*1 Record クラスの説明には、get_Item メソッドは記載されていませんが、Record クラスが System.Collections 名前空間の Hashtable クラスから派生していることから、Item オブジェクトが自動生成されています。

4. Update 処理

この処理では、Textbox で変更したデータの更新を行います。

```
* First_Name フィールドの内容を textbox1 に代入
    SET STRBUF2 to PROP-TEXT OF textBox1.
    INVOKE BtLib2 "Set_Item" USING N"First_Name" STRBUF2.
* Last_Name フィールドの内容を textbox2 に代入
    SET STRBUF2 to PROP-TEXT OF textBox2.
    INVOKE BtLib2 "Set_Item" USING N"Last_Name" STRBUF2.
* City フィールドの内容を textbox3 に代入
    SET STRBUF2 to PROP-TEXT OF textBox3.
    INVOKE BtLib2 "Set_Item" USING N"City" STRBUF2.
* 次のレコードを更新
    SET B-OP TO PROP-update OF BtLib-OP .
    INVOKE BtLib2 "Write" USING BY VALUE B-OP RETURNING RC.
* 戻り値を判定
    if RC Not = 0 then
* 戻り値を label3 に表示
        move RC to RCStr
        string "RC = " RCStr delimited by size INTO msgBUF
        SET PROP-TEXT OF label3 to msgBUF
    end-if
```

SET 文で Textbox の text プロパティから文字列変数へ値を設定します。

Record クラスの Set_Item (*2) メソッドを 1 つ目の引数にフィールド名、2 つ目の引数に書き込む値を指定して呼び出します。

Record クラスの Write メソッドを引数にオペレーションコード (Update) を指定して呼び出します。

戻り値が 0 でなければ、Label にエラーを出力します。

*2 Record クラスの説明には、set_Item メソッドは記載されていませんが、Record クラスが System.Collections 名前空間の Hashtable クラスから派生していることから、Item オブジェクトが自動生成されています。

5. Close 処理

この処理では、ファイルのクローズを行います。

INVOKE BtLib2 "Close".

Record クラスの Close メソッドを呼び出すことで、ファイルのクローズを行います。

■サンプルプロジェクトのビルド方法

1. 任意のフォルダーを作成し、BCNsample.zip を解凍してください。
2. Visual Studio でプロジェクトを開きます。
3. ターゲットを x86 に変更します。
 - a. [ビルド] -> [構成マネージャー] の順にクリックし、「アクティブ ソリューション プラットフォーム」で「<新規作成>」をクリックします。
 - b. 「新しいプラットフォームを入力または選択してください」で「x86」を選択し、OK をクリックします。

【ご注意】 Btrieve Classes for .NET は、アンマネージコードで作成されているため、ターゲットが Any CPU では、実行できません。

4. ビルドして実行します。

■64bit アプリケーションを作成する場合の注意点

64bit アプリケーションを作成する場合、64bit の Btlib.dll を使用する必要があります。

64bit の Btlib.dll は、次の zip ファイルに圧縮されています。

解凍して、bin フォルダーにコピーします。

(32bit の Btlib.dll は他のフォルダー等にバックアップしてください。)

Program Files (x86)\TechKnowledge\Btrieve Classes for .NET 8.0\bin64\BtLib.zip

また、ターゲットを x64 に変更する必要があります。

- a. [ビルド] -> [構成マネージャー] の順にクリックし、「アクティブ ソリューション プラットフォーム」で「<新規作成>」をクリックします。
- b. 「新しいプラットフォームを入力または選択してください」で「x64」を選択し、OK をクリックします。

※本書は情報提供のみを目的としており、本書の内容、提供のサンプルの使用において、明示的あるいは非明示的ないかなる保証もいたしません。

※著作権上の権利に限定されることなく、本書の一部または全部を無断で使用、複製することはできません。

※本書の内容は、2021 年 6 月現在のものです。また、予告なく変更する場合があります。

※NetCOBOL は富士通株式会社の登録商標です。

※すべての Actian のブランド名は、Actian Corporation の米国およびその他の国における登録商標または商標です。

※本書に記載されている、会社名、製品名等は、各社の登録商標または商標です。

(C) Copyright 2021 AG-TECH Corp. All rights reserved.

株式会社エージーテック

〒101-0054

東京都千代田区神田錦町 1-21-1 ヒューリック神田橋ビル 3F