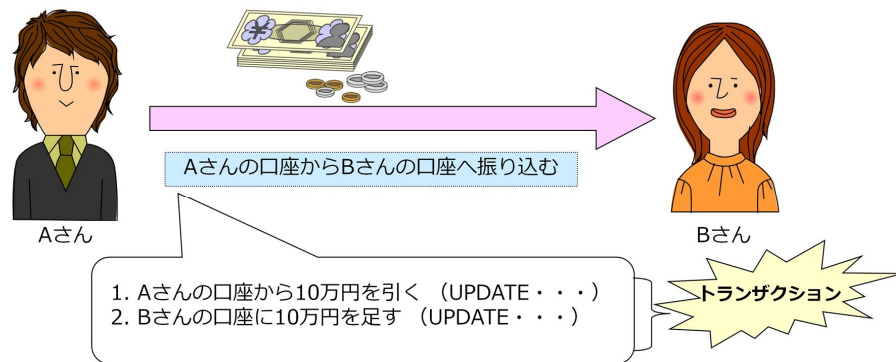


リフレクションレビュー

- ①ADO.NETにおけるトランザクション処理の方法
- ②パラメータクエリを実装する方法

①ADO.NETにおけるトランザクション処理の方法

①ADO.NETにおけるトランザクション処理の方法 トランザクションとは(復習)



トランザクション : 処理の基本単位のこと。トランザクション内で処理が中断された場合は、トランザクション開始直前まで状態を巻き戻すことができる。

コミット : トランザクション内の処理が **問題なく終わった場合**、処理を確定させること。

ロールバック : トランザクション内の処理で **問題があった場合**、データを開始直前の状態に巻き戻すこと。

①ADO.NETにおけるトランザクション処理の方法 ADO.NETでの利用

ADO.NETでは、トランザクションを **暗黙的なトランザクション** と **明示的なトランザクション** の2種類の方法で扱える。

①ADO.NETにおけるトランザクション処理の方法 ADO.NETでの利用

ADO.NETでは、トランザクションを **暗黙的なトランザクション** と **明示的なトランザクション** の2種類の方法で扱える。

暗黙的なトランザクション：

トランザクションの終了宣言(コミット・ロールバック)を、**自動的に処理させる**トランザクション。
(自動的に処理させるという部分が、「暗黙的」と呼ばれる理由だと思います。)

①ADO.NETにおけるトランザクション処理の方法 ADO.NETでの利用

ADO.NETでは、トランザクションを **暗黙的なトランザクション** と **明示的なトランザクション** の2種類の方法で扱える。

暗黙的なトランザクション：

トランザクションの終了宣言(コミット・ロールバック)を、**自動的に処理させる**トランザクション。
(自動的に処理させるという部分が、「暗黙的」と呼ばれる理由だと思います。)

明示的なトランザクション：

トランザクションの終了宣言(コミット・ロールバック)を、**コード内に記述する**トランザクション。
(コミット・ロールバックを実際に宣言するという部分が、「明示的」と呼ばれる理由だと思います。)

①ADO.NETにおけるトランザクション処理の方法 具体的な処理

昨日扱われた、暗黙的なトランザクションを例にします。

【参考】暗黙的なトランザクションの処理

トランザクションの範囲を定めるために必要なクラスを読み込む。



```
1. Imports System.Data.SqlClient
2. Imports System.Transactions    ※ファイルの先頭に宣言
3.
4. Try
5.
6.     Using scope As TransactionScope = New TransactionScope ()
7.
8.         'DBへの接続処理
9.         '各テーブルへのコマンド実行処理
10.        scope.Complete()
11.        'DBとの切断処理
12.    End Using
13.
14.        '成功時の処理
15.
16. Catch se As SqlException
17.
18.        '失敗時の処理
19. Finally
20.
21.        '例外の有無にかかわらず最後に行う処理 (DBとの切断処理など)
22. End Try
```

①ADO.NETにおけるトランザクション処理の方法 具体的な処理

昨日扱われた、暗黙的なトランザクションを例にします。

【参考】暗黙的なトランザクションの処理

コミットするかロールバックするかは
例外処理によって判断させる。



```
1. Imports System.Data.SqlClient
2. Imports System.Transactions    ※ファイルの先頭に宣言
3.
4. Try
5.
6.     Using scope As TransactionScope = New TransactionScope ()
7.
8.         'DBへの接続処理
9.         '各テーブルへのコマンド実行処理
10.        scope.Complete()
11.        'DBとの切断処理
12.    End Using
13.
14.        '成功時の処理
15.
16. Catch se As SqlException
17.
18.        '失敗時の処理
19. Finally
20.
21.        '例外の有無にかかわらず最後に行う処理（DBとの切断処理など）
22. End Try
```


①ADO.NETにおけるトランザクション処理の方法 具体的な処理

昨日扱われた、暗黙的なトランザクションを例にします。

【参考】暗黙的なトランザクションの処理

トランザクションの開始
(ここでTransactionsクラスを利用)



```
1. Imports System.Data.SqlClient
2. Imports System.Transactions    ※ファイルの先頭に宣言
3.
4. Try
5.
6.     Using scope As TransactionScope = New TransactionScope ()
7.
8.         'DBへの接続処理
9.         '各テーブルへのコマンド実行処理
10.        scope.Complete()
11.        'DBとの切断処理
12.    End Using
13.
14.        '成功時の処理
15.
16. Catch se As SqlException
17.
18.        '失敗時の処理
19. Finally
20.
21.        '例外の有無にかかわらず最後に行う処理 (DBとの切断処理など)
22. End Try
```

①ADO.NETにおけるトランザクション処理の方法 具体的な処理

昨日扱われた、暗黙的なトランザクションを例にします。

【参考】暗黙的なトランザクションの処理

例外が起こらないなら、
Completeメソッドによりコミットされる
(これが働かなければ、ロールバックされることとなる)



```
1. Imports System.Data.SqlClient
2. Imports System.Transactions    ※ファイルの先頭に宣言
3.
4. Try
5.
6.     Using scope As TransactionScope = New TransactionScope ()
7.
8.         'DBへの接続処理
9.         '各テーブルへのコマンド実行処理
10.        scope.Complete()
11.        'DBとの切断処理
12.    End Using
13.
14.        '成功時の処理
15.
16. Catch se As SqlException
17.
18.        '失敗時の処理
19. Finally
20.
21.        '例外の有無にかかわらず最後に行う処理 (DBとの切断処理など)
22. End Try
```


①ADO.NETにおけるトランザクション処理の方法 具体的な処理

昨日扱われた、暗黙的なトランザクションを例にします。

【参考】暗黙的なトランザクションの処理

```
1. Imports System.Data.SqlClient
2. Imports System.Transactions    ※ファイルの先頭に宣言
3.
4. Try
5.
6.     Using scope As TransactionScope = New TransactionScope ()
7.
8.         'DBへの接続処理
9.         '各テーブルへのコマンド実行処理
10.        scope.Complete()
11.        'DBとの切断処理
12.    End Using
13.
14.        '成功時の処理
15.
16. Catch se As SqlException
17.
18.        '失敗時の処理
19. Finally
20.
21.        '例外の有無にかかわらず最後に行う処理（DBとの切断処理など）
22. End Try
```

この時点で、
Completeメソッドが実行されたかどうかに応じて、コミットかロールバックがなされている。



②パラメータクエリを実装する方法

②パラメータクエリを実装する方法 パラメータクエリとは

パラメータクエリ: プレースホルダーを使って表現されたクエリ

(参考) プレースホルダー: 実際の内容を後から挿入するために、とりあえず仮に確保した場所のこと。IT用語辞典 e-WORDSより)

クエリ: データベース管理システムに対する問合せ(処理要求)のこと。ITトレンド IT用語集より)

②パラメータクエリを実装する方法 パラメータークエリとは

パラメータークエリ: プレースホルダーを使って表現されたクエリ

(参考) プレースホルダー: 実際の内容を後から挿入するために、とりあえず仮に確保した場所のこと。IT用語辞典 e-WORDSより)
クエリ: データベース管理システムに対する問合せ(処理要求)のこと。ITトレンド IT用語集より)

普通のクエリ:

```
SELECT * FROM Employee WHERE EmpName = 小林賢太郎 And DeptCode = SH0001
```

=「Employeeテーブルから、EmpNameが**小林賢太郎**で、DeptCodeが**SH0001**の人の情報を表示してください」

②パラメータクエリを実装する方法 パラメータクエリとは

パラメータクエリ: プレースホルダーを使って表現されたクエリ

(参考)プレースホルダー: 実際の内容を後から挿入するために、とりあえず仮に確保した場所のこと。IT用語辞典 e-WORDSより)

クエリ: データベース管理システムに対する問合せ(処理要求)のこと。ITトレンド IT用語集より)

普通のクエリ:

```
SELECT * FROM Employee WHERE EmpName = 小林賢太郎 And DeptCode = SH0001
```

=「Employeeテーブルから、EmpNameが小林賢太郎で、DeptCodeがSH0001の人の情報を表示してください」

パラメータクエリ:

プレースホルダー

プレースホルダー

```
SELECT * FROM Employee WHERE EmpName = @empName And DeptCode = @deptCode
```

=「Employeeテーブルから、EmpNameが@empNameで、DeptCodeが@deptCodeの人の情報を表示してください」

②パラメータクエリを実装する方法 パラメータクエリの実装方法

前提①: VB.NET上でSQLの管理・実行を行うためには **SqlCommand**クラスの機能が必要

②パラメータクエリを実装する方法 パラメータクエリの実装方法

前提①: VB.NET上でSQLの管理・実行を行うためには **SqlCommand**クラスの機能が必要

前提②: SqlCommandクラスは、次のようなメンバーをもつ。

- ・**Connection**プロパティ: どのデータベースを使うかが入るSqlConnectionオブジェクトとして)
- ・**CommandText**プロパティ: クエリが入る(Connectionプロパティの値に対して実行するための)
- ・**Parameters**プロパティ: SqlParameterCollection (Parameterオブジェクトの配列)を取得する。
(補足)Parameterオブジェクトとは、プレースホルダー名とその値の組のこと。

②パラメータクエリを実装する方法 パラメータクエリの実装方法

前提①: VB.NET上でSQLの管理・実行を行うためには **SqlCommand**クラスの機能が必要

前提②: SqlCommandクラスは、次のようなメンバーをもつ。

- ・**Connection**プロパティ: どのデータベースを使うかが入るSqlConnectionオブジェクトとして)
- ・**CommandText**プロパティ: クエリが入る(Connectionプロパティの値に対して実行するための)
- ・**Parameters**プロパティ: SqlParameterCollection (Parameterオブジェクトの配列)を取得する。
(補足)Parameterオブジェクトとは、プレースホルダー名とその値の組のこと。

前提③: **New SqlParameter**(“プレースホルダー名”, 値) とすることで、
SqlParameterCollectionに新たなParameterオブジェクトを生成できる。

②パラメータクエリを実装する方法 パラメータクエリの実装方法

前提①: VB.NET上でSQLの管理・実行を行うためには **SqlCommand**クラスの機能が必要

前提②: SqlCommandクラスは、次のようなメンバーをもつ。

- **Connection**プロパティ: どのデータベースを使うかが入るSqlConnectionオブジェクトとして)
- **CommandText**プロパティ: クエリが入る(Connectionプロパティの値に対して実行するための)
- **Parameters**プロパティ: SqlParameterCollection (Parameterオブジェクトの配列)を取得する。
(補足)Parameterオブジェクトとは、プレースホルダー名とその値の組のこと。

前提③: **New SqlParameter("プレースホルダー名", 値)** とすることで、SqlParameterCollectionに新たなParameterオブジェクトを生成できる。

```
2.
3. '実行時に合わせて変更する部分に、@パラメーター名のプレースホルダーを設定
4. Dim DBCmd As New SqlCommand()
5. DBCmd.Connection = DBConnection
6. DBCmd.CommandText = "SELECT * FROM Employee " & _
7.                      "WHERE EmpName = @empName And " & _
8.                      "DeptCode = @deptCode"
9. 'プレースホルダーに対応するパラメーターオブジェクトを生成
10. DBCmd.Parameters.Add _
11.    (New SqlParameter("@empName", ShopLogin.UserName))
12. DBCmd.Parameters.Add _
13.    (New SqlParameter("@deptCode", ShopLogin.Password))
14.
15. Dim reader As SqlDataReader
16. reader = DBCmd.ExecuteReader()
```

1

2 および 3

4

実装例

②パラメータクエリを実装する方法 パラメータクエリの実装方法

前提①: VB.NET上でSQLの管理・実行を行うためには **SqlCommand**クラスの機能が必要

前提②: SqlCommandクラスは、次のようなメンバーをもつ。

- ・**Connection**プロパティ: どのデータベースを使うかが入るSqlConnectionオブジェクトとして)
- ・**CommandText**プロパティ: クエリが入る(Connectionプロパティの値に対して実行するための)
- ・**Parameters**プロパティ: SqlParameterCollection (Parameterオブジェクトの配列)を取得する。
(補足)Parameterオブジェクトとは、プレースホルダー名とその値の組のこと。

前提③: **New SqlParameter("プレースホルダー名", 値)** とすることで、SqlParameterCollectionに新たなParameterオブジェクトを生成できる。

SqlCommandクラスのインスタンス
「**DBCmd**」生成

```
2.
3. '実行時に合わせて変更する部分に、@パラメーター名のプレースホルダーを設定
4. Dim DBCmd As New SqlCommand()
5. DBCmd.Connection = DBConnection
6. DBCmd.CommandText = "SELECT * FROM Employee " & _
7.                      "WHERE EmpName = @empName And " & _
8.                      "DeptCode = @deptCode"
9. 'プレースホルダーに対応するパラメーターオブジェクトを生成
10. DBCmd.Parameters.Add _
11.    (New SqlParameter("@empName", ShopLogin.UserName))
12. DBCmd.Parameters.Add _
13.    (New SqlParameter("@deptCode", ShopLogin.Password))
14.
15. Dim reader As SqlDataReader
16. reader = DBCmd.ExecuteReader()
```

1

4

2 および 3

実装例

②パラメータクエリを実装する方法 パラメータクエリの実装方法

前提①: VB.NET上でSQLの管理・実行を行うためには **SqlCommand**クラスの機能が必要

前提②: SqlCommandクラスは、次のようなメンバーをもつ。

- ・**Connection**プロパティ: どのデータベースを使うかが入るSqlConnectionオブジェクトとして)
- ・**CommandText**プロパティ: クエリが入る(Connectionプロパティの値に対して実行するための)
- ・**Parameters**プロパティ: SqlParameterCollection (Parameterオブジェクトの配列)を取得する。
(補足)Parameterオブジェクトとは、プレースホルダー名とその値の組のこと。

前提③: **New SqlParameter("プレースホルダー名", 値)** とすることで、SqlParameterCollectionに新たなParameterオブジェクトを生成できる。

DBCmdが扱うデータベースの指定
(**Connection**プロパティ)

```
1.
2.
3. '実行時に合わせて変更する部分に、@パラメーター名のプレースホルダーを設定
4. Dim DBCmd As New SqlCommand()
5. DBCmd.Connection = DBConnection
6. DBCmd.CommandText = "SELECT * FROM Employee " & _
7.                      "WHERE EmpName = @empName And " & _
8.                      "DeptCode = @deptCode"
9. 'プレースホルダーに対応するパラメーターオブジェクトを生成
10. DBCmd.Parameters.Add _
11.    (New SqlParameter("@empName", ShopLogin.UserName))
12. DBCmd.Parameters.Add _
13.    (New SqlParameter("@deptCode", ShopLogin.Password))
14.
15. Dim reader As SqlDataReader
16. reader = DBCmd.ExecuteReader()
```

1

2 および 3

4

実装例

②パラメータクエリを実装する方法 パラメータクエリの実装方法

前提①: VB.NET上でSQLの管理・実行を行うためには **SqlCommand**クラスの機能が必要

前提②: SqlCommandクラスは、次のようなメンバーをもつ。

- ・**Connection**プロパティ: どのデータベースを使うかが入るSqlConnectionオブジェクトとして)
- ・**CommandText**プロパティ: クエリが入る(Connectionプロパティの値に対して実行するための)
- ・**Parameters**プロパティ: SqlParameterCollection (Parameterオブジェクトの配列)を取得する。
(補足)Parameterオブジェクトとは、プレースホルダー名とその値の組のこと。

前提③: **New SqlParameter("プレースホルダー名", 値)** とすることで、SqlParameterCollectionに新たなParameterオブジェクトを生成できる。

実行したいクエリを**CommandText**に格納

```
1.
2.
3. '実行時に合わせて変更する部分に、@パラメーター名のプレースホルダーを設定
4. Dim DBCmd As New SqlCommand()
5. DBCmd.Connection = DBConnection
6. DBCmd.CommandText = "SELECT * FROM Employee " & _
7.                      "WHERE EmpName = @empName And " & _
8.                      "DeptCode = @deptCode"
9. 'プレースホルダーに対応するパラメーターオブジェクトを生成
10. DBCmd.Parameters.Add _
11.    (New SqlParameter("@empName", ShopLogin.UserName))
12. DBCmd.Parameters.Add _
13.    (New SqlParameter("@deptCode", ShopLogin.Password))
14.
15. Dim reader As SqlDataReader
16. reader = DBCmd.ExecuteReader()
```

1

2 および 3

4

実装例

②パラメータクエリを実装する方法 パラメータクエリの実装方法

前提①: VB.NET上でSQLの管理・実行を行うためには **SqlCommand**クラスの機能が必要

前提②: SqlCommandクラスは、次のようなメンバーをもつ。

- ・**Connection**プロパティ: どのデータベースを使うかが入るSqlConnectionオブジェクトとして)
- ・**CommandText**プロパティ: クエリが入る(Connectionプロパティの値に対して実行するための)
- ・**Parameters**プロパティ: SqlParameterCollection (Parameterオブジェクトの配列)を取得する。
(補足)Parameterオブジェクトとは、プレースホルダー名とその値の組のこと。

前提③: **New SqlParameter("プレースホルダー名", 値)** とすることで、SqlParameterCollectionに新たなParameterオブジェクトを生成できる。

実行したいクエリを**CommandText**に格納

プレースホルダー有り。
(@empName, @deptCode)

```
1.
2.
3. '実行時に合わせて変更する部分に、@パラメーター名のプレースホルダーを設定
4. Dim DBCmd As New SqlCommand()
5. DBCmd.Connection = DBConnection
6. DBCmd.CommandText = "SELECT * FROM Employee " &
7.                      "WHERE EmpName = @empName And " & _
8.                      "DeptCode = @deptCode"
9. 'プレースホルダーに対応するパラメーターオブジェクトを生成
10. DBCmd.Parameters.Add _
11.    (New SqlParameter("@empName", ShopLogin.UserName))
12. DBCmd.Parameters.Add _
13.    (New SqlParameter("@deptCode", ShopLogin.Password))
14.
15. Dim reader As SqlDataReader
16. reader = DBCmd.ExecuteReader()
```

1

4

2 および 3

実装例

つまり、ここでパラメータクエリが設定された！

②パラメータクエリを実装する方法 パラメータクエリの実装方法

前提①: VB.NET上でSQLの管理・実行を行うためには **SqlCommand**クラスの機能が必要

前提②: SqlCommandクラスは、次のようなメンバーをもつ。

- **Connection**プロパティ: どのデータベースを使うかが入るSqlConnectionオブジェクトとして)
- **CommandText**プロパティ: クエリが入る (Connectionプロパティの値に対して実行するための)
- **Parameters**プロパティ: SqlParameterCollection (Parameterオブジェクトの配列)を取得する。
(補足)Parameterオブジェクトとは、プレースホルダー名とその値の組のこと。

前提③: **New SqlParameter("プレースホルダー名", 値)** とすることで、SqlParameterCollectionに新たなParameterオブジェクトを生成できる。

プレースホルダーの具体的な値を設定

```
1.
2.
3. '実行時に合わせて変更する部分に、@パラメーター名のプレースホルダーを設定
4. Dim DBCmd As New SqlCommand()
5. DBCmd.Connection = DBConnection
6. DBCmd.CommandText = "SELECT * FROM Employee " & _
7.                      "WHERE EmpName = @empName And " & _
8.                      "DeptCode = @deptCode"
9. 'プレースホルダーに対応するパラメーターオブジェクトを生成
10. DBCmd.Parameters.Add _
11.    (New SqlParameter("@empName", ShopLogin.UserName))
12. DBCmd.Parameters.Add _
13.    (New SqlParameter("@deptCode", ShopLogin.Password))
14.
15. Dim reader As SqlDataReader
16. reader = DBCmd.ExecuteReader()
```

実装例

②パラメータクエリを実装する方法 パラメータクエリの実装方法

前提①: VB.NET上でSQLの管理・実行を行うためには **SqlCommand**クラスの機能が必要

前提②: SqlCommandクラスは、次のようなメンバーをもつ。

- ・**Connection**プロパティ: どのデータベースを使うかが入るSqlConnectionオブジェクトとして)
- ・**CommandText**プロパティ: クエリが入る(Connectionプロパティの値に対して実行するための)
- ・**Parameters**プロパティ: SqlParameterCollection (Parameterオブジェクトの配列)を取得する。
(補足)Parameterオブジェクトとは、プレースホルダー名とその値の組のこと。

前提③: **New SqlParameter("プレースホルダー名", 値)** とすることで、SqlParameterCollectionに新たなParameterオブジェクトを生成できる。

続く処理が行われる。

これでパラメータクエリは実装できた。

```
1.
2.
3. '実行時に合わせて変更する部分に、@パラメーター名のプレースホルダーを設定
4. Dim DBCmd As New SqlCommand()
5. DBCmd.Connection = DBConnection
6. DBCmd.CommandText = "SELECT * FROM Employee " & _
7.                      "WHERE EmpName = @empName And " & _
8.                      "DeptCode = @deptCode"
9. 'プレースホルダーに対応するパラメーターオブジェクトを生成
10. DBCmd.Parameters.Add _
11.    (New SqlParameter("@empName", ShopLogin.UserName))
12. DBCmd.Parameters.Add _
13.    (New SqlParameter("@deptCode", ShopLogin.Password))
14.
15. Dim reader As SqlDataReader
16. reader = DBCmd.ExecuteReader()
```

1

2 および 3

4

実装例

リフレクションレビューおわり

質問

質問

質問1 次のうち、トランザクションの説明として正しいものはどれですか？（一つだけあります）

- ①何がトランザクションになるかは、コンピューターが判断する。
- ②何がトランザクションになるかは、人間が判断する。
- ③もしトランザクションの中で問題が起こると、トランザクション中で扱われたデータに異常が生じる。

質問2 以下のコードだけで判断できる、間違っている行は何行目ですか？（一つだけあります）

- ☐ 101. DBCmd.CommandText = "SELECT * FROM Table1 WHERE Name = @Name"
- 102. @Name = "小林賢太郎"
- 103. DBCmd.ExecuteNonQuery()
- ☐

質問

質問1 次のうち、トランザクションの説明として正しいものはどれですか？（一つだけあります）

- ①何がトランザクションになるかは、コンピューターが判断する。×
→人間が判断する。
- ②何がトランザクションになるかは、人間が判断する。○
→正解。
- ③もしトランザクションの中で問題が起こると、トランザクション中で扱われたデータに異常が生じる。×
→それを防ぐためにトランザクションを指定する。

質問2 以下のコードだけで判断できる、間違っている行は何行目ですか？（一つだけあります）

```
☐
101. DBCmd.CommandText = "SELECT * FROM Table1 WHERE Name = @Name"
102. @Name = "小林賢太郎"
103. DBCmd.ExecuteNonQuery()
☐
```

質問

質問1 次のうち、トランザクションの説明として正しいものはどれですか？（一つだけあります）

- ①何がトランザクションになるかは、コンピューターが判断する。×
→人間が判断する。
- ②何がトランザクションになるかは、人間が判断する。○
→正解。
- ③もしトランザクションの中で問題が起こると、トランザクション中で扱われたデータに異常が生じる。×
→それを防ぐためにトランザクションを指定する。

質問2 以下のコードだけで判断できる、間違っている行は何行目ですか？（一つだけあります）

```
□  
101. DBCmd.CommandText = "SELECT * FROM Table1 WHERE Name = @Name"  
102. @Name = "小林賢太郎"  
103. DBCmd.ExecuteNonQuery()  
□
```

←「@Name」はVB.NET上の変数ではないので間違い
DBCmd.Parameters.Add("@Name", "小林賢太郎") が正解

質問

質問1 次のうち、トランザクションの説明として正しいものはどれですか？（一つだけあります）

- ①何がトランザクションになるかは、コンピューターが判断する。×
→人間が判断する。
- ②何がトランザクションになるかは、人間が判断する。○
→正解。
- ③もしトランザクションの中で問題が起こると、トランザクション中で扱われたデータに異常が生じる。×
→それを防ぐためにトランザクションを指定する。

質問2 以下のコードだけで判断できる、間違っている行は何行目ですか？（一つだけあります）

```
□  
101. DBCmd.CommandText = "SELECT * FROM Table1 WHERE Name = @Name"  
102. @Name = "小林賢太郎"  
103. DBCmd.ExecuteNonQuery()  
□
```

←「@Name」はVB.NET上の変数ではないので間違い
DBCmd.Parameters.Add("@Name", "小林賢太郎") が正解

全て終わりです。ありがとうございました。