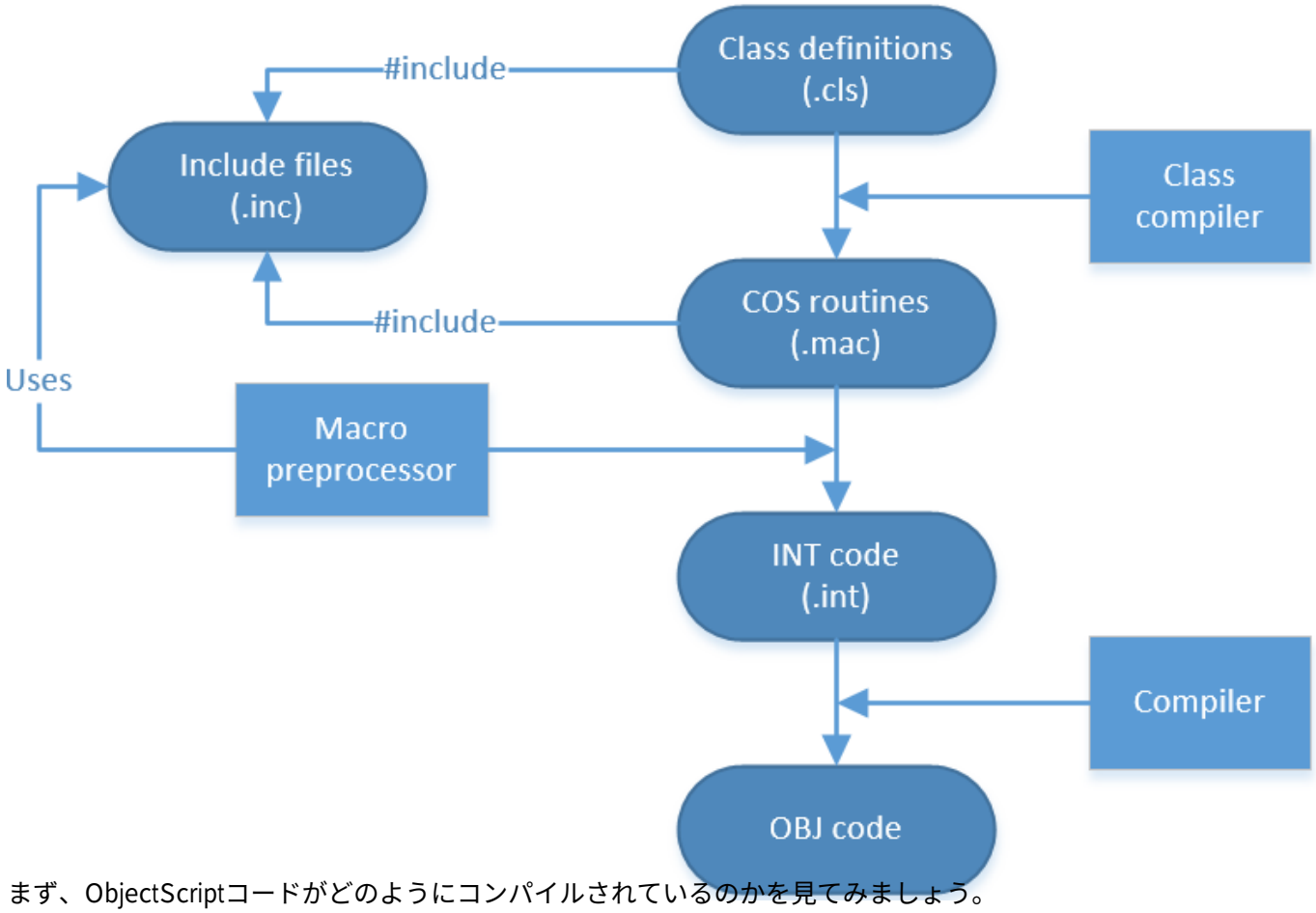
記事
Toshihiko Minamoto · 2021年6月23日 10m read

InterSystems Cachéのマクロ

この記事では、InterSystems Cachéにおけるマクロについて説明します。
マクロは、コンパイル中に一連の命令に置き換えられるシンボリック名です。 マクロは、渡されたパラメーターとアクティブ化したシナリオに応じて、呼び出されるたびに一連の命令セットに「展開」されます。
これは、静的コードの場合もあれば、ObjectScriptを実行して得られる結果である場合もあります。
それでは、アプリケーションでマクロをどのように使用できるのかを見てみましょう。

コンパイル



まず、ObjectScriptコードがどのようにコンパイルされているのかを見てみましょう。

クラスコンパイラはクラス定義を使用してMACコードを生成します。
場合によっては、コンパイラはクラスを元に追加クラスを生成します。
これらのクラスはStudioで閲覧できますが、変更してはいけません。
この動作は、たとえば、WebサービスやWebクライアントのクラスを生成する際に発生します。
クラスコンパイラはランタイム時にCaché が使用するクラス記述子も生成します。
プリプロセッサ（マクロプロセッサまたはMPPとも呼ばれます）が、INCファイルを使用してマクロを置き換えます。 さらに、ObjectScriptルーチンにある埋め込みSQLも処理します。
これらの変更はすべてメモリで発生するため、ユーザーのコードに変化はありません。

その後、コンパイラはObjectScriptルーチンのINTコードを作成します。
このレイヤーは中間コードとして知られるレイヤーです。
このレベルでのデータへのすべてのアクセスは、グローバルを介して提供されます。
INTコードはコンパクトで、人間が読み取ることができます。
Studioで閲覧するには、Ctrl＋Shift＋Vを押してください。
INTコードを使用して、OBJコードが生成されます。
OBJコードは、Caché仮想マシンが使用するコードです。 OBJコードが生成されるとCLS/MAC/INTコードは不要になるため、不要となったそれらのコードは削除することができます（ソースコードを含めずに製品を出荷する場合など）。

1. クラスが永続クラスである場合、SQLコンパイラは対応するSQLテーブルを作成します。

マクロ

前に述べたように、マクロは、プリプロセッサによって命令セットに置き換えられるシンボリック名です。
マクロは#Defineコマンドにマクロ名（おそらく引数のリストを含む）とその値を続けて定義します。

```
#Define Macro[(Args)] [Value]
```

マクロはどこに定義されるのでしょうか。
マクロの定義は、コード内か、マクロのみを含む独立したINCファイルで行われます。
必要なファイルは、クラス定義の最初にInclude MacroFileNameコマンドを使用してクラスに含められます。これがマクロをクラスに含めるための推奨される主な方法です。
この方法で含められるマクロは、クラスのどの部分にでも使用できます。 #Include MacroFileNameコマンドを使ってマクロを含むINCファイルをMACルーチンや特定のクラスメソッドのコードに含めることができます。

マクロをコンパイル時に使用する場合、またはクラスにIncludeGenerator
キーワードを使用する場合は、メソッドジェネレーターの本文に#Includeを使用する必要があることに注意してください。

Studioの自動補完でマクロを使用できるようにするには、前の行に///を追加します。

```
///
#Define Macro[(Args)] [Value]
```

例

例1

では、例をいくつか見てみましょう。標準的な「Hello World」メッセージから始めます。
COSコードは次のようになります。

```
Write "Hello, World!"
```

HWという、次の行を書き込むマクロを作成します。

```
#define HW Write "Hello, World!"
```

後は、$$$HW（マクロを呼び出す$$$と、その後にマクロ名を指定）を記述するのみです。

```
ClassMethod Test()
{
     #define HW Write "Hello, World!"
     $$$HW
}
```

これは、コンパイル中に次のINTコードに変換されます。

```
zTest1() public {
     Write "Hello, World!" }
```

このメソッドが呼び出されると、ターミナルに次のテキストが表示されます。

```
Hello, World!
```

例2

次の例では、変数を使用し見ましょう。

```
ClassMethod Test2()
{
     #define WriteLn(%str,%cnt) For ##Unique(new)=1:1:%cnt { ##Continue
         Write %str,! ##Continue
     }

     $$$WriteLn("Hello, World!",5)
}
```

上記のコードでは、%str文字列が%cnt回書き込まれます。 変数名は%で始まる必要があります。 ##Unique(new)
コマンドで、生成されたコードに新しい一意の変数
を作成し、##Continueによって、次の行にマクロの定義が続くことを示します。
このコードは、次のINTコードに変換されます。

```
zTest2() public {
     For %mmmu1=1:1:5 {
         Write "Hello, World!",!
     } }
```

ターミナルには次のように表示されます。

```
Hello, World!
Hello, World!
Hello, World!
Hello, World!
Hello, World!
```

例3

より複雑な例に進みましょう。 ForEach
演算子は、グローバルを反復処理する上で非常に役立ちます。それでは作成してみましょう。

```
ClassMethod Test3()
```

```
{
    #define ForEach(%key,%gn) Set ##Unique(new)=$name(%gn) ##Continue
    Set %key="" ##Continue
    For { ##Continue
        Set %key=$o(@##Unique(old)@(%key)) ##Continue
        Quit:%key=""

    #define EndFor    }

       Set ^test(1)=111
       Set ^test(2)=222
       Set ^test(3)=333

       $$$ForEach(key,^test)
           Write "key: ",key,!
           Write "value: ",^test(key),!
       $$$EndFor
}
```

INTコードでは次のようになります。

```
zTest3() public {
       Set ^test(1)=111
       Set ^test(2)=222
       Set ^test(3)=333
       Set %mmmu1=$name(^test)
       Set key=""
       For {
           Set key=$o(@%mmmu1@(key))
           Quit:key=""
           Write "key: ",key,!
           Write "value: ",^test(key),!
       } }
```

これらのマクロでは何が起こっているのでしょうか。

1. グローバルの名前を新しい%mmmu1変数に書き込みます（$name関数）。
2. キーは、初期の空の文字列値です。
3. 反復サイクルが開始します。
4. 間接演算子と$order関数を使って、キーに次の値が割り当てられます。
5. キーが""値を取っているかどうかを、事後条件
   を使ってチェックします。取っている場合は反復が完了し、サイクルが終了します。
6. 任意のユーザーコードが実行されます。この場合、キーと値が出力されます。
7. サイクルが終了します。

このメソッドが呼び出されると、ターミナルには次のように表示されます。

```
key: 1
value: 111
key: 2
value: 222
key: 3
value: 333
```

%Collection.AbstractIterator

クラスから継承したリストと配列を使用している場合は、同様のイテレーターを記述できます。

例4

マクロにはさらに、コンパイル段階で任意のObjectScriptコードを実行し、マクロの代わりにその結果に置き換えるという別の機能があります。 コンパイル時間を示すマクロを作成してみましょう。

```
ClassMethod Test4()
{
      #Define CompTS ##Expression("""Compiled: " _ $ZDATETIME($HOROLOG) _ """,!")
      Write $$$CompTS
}
```

これは、次のINTコードに変換されます。

```
zTest4() public {
      Write "Compiled: 18.10.2016 15:28:45",! }
```

このメソッドが呼び出されると、ターミナルには次の行が表示されます。

```
Compiled: 18.10.2015 15:28:45
```

##Expressionは、コードを実行して結果を置き換えます。
入力には、ObjectScript言語の次の要素を使用できます。

- 文字列: "abc"
- ルーチン: $$Label^Routine
- クラスメソッド: ##class(App.Test).GetString()
- COS関数: $name(var)
- 上記の要素の任意の組み合わせ

例5

コンパイル時に、ディレクティブの後に続く式の値に応じてソースコードを選択するには、プリプロセッサディレクティブの#If、#ElseIf、#Else、#EndIfを使用します。 たとえば、次のメソッドがあるとします。

```
ClassMethod Test5()
{
    #If $SYSTEM.Version.GetNumber()="2016.2.0" && $SYSTEM.Version.GetBuildNumber()="7
36"
        Write "You are using the latest released version of Caché"
    #ElseIf $SYSTEM.Version.GetNumber()="2017.1.0"
        Write "You are using the latest beta version of Caché"
    #Else
        Write "Please consider an upgrade"
    #EndIf
}
```

Cachéバージョン2016.2.0.736では、このメソッドは次のINTコードにコンパイルされます。

```
zTest5() public {
    Write "You are using the latest released version of Caché"
}
```

ターミナルには次のように表示されます。

```
You are using the latest released version of Caché
```

ベータポータルからダウンロードしたCachéを使用している場合、コンパイルされたINTコードは異なります。

```
zTest5() public {
    Write "You are using the latest beta version of Caché"
}
```

ターミナルには次のように表示されます。

```
You are using the latest beta version of Caché
```

古いバージョンのCachéは、次のようにプログラムの更新を提案するINTコードをコンパイルします。

```
zTest5() public {
    Write "Please consider an upgrade"
}
```

ターミナルには次のように表示されます。

```
Please consider an upgrade
```

この機能は、クライアントアプリケーションで古いバージョンと新しいバージョンとの互換性を保証したい場合に、Cachéの新機能が使用される可能性があるときなどに役立ちます。
プリプロセッサディレクティブの#IfDefと#IfNDef
は、順にマクロの存在と不在を検証することで、同じ目的を果たすことができます。

まとめ

マクロは、コンパイル段階で、頻繁に使用される構造を単純化することでコードを読みやすくして一部のアプリケーションのビジネスロジックを実装しやすくするため、ランタイム時の負荷を軽減することができます。

次の内容

次の記事では、アプリケーションにおけるマクロのより実用的な使用例について説明します。ロギングシステムです。

リンク

- コンパイルについて
- プリプロセッサディレクティブ一覧
- システムマクロ一覧

- この記事の例を使ったクラス
- 第2部: ロギング

#ObjectScript #コンパイラ #ターミナル #ヒントとコツ #ベストプラクティス #初心者 #Caché

ソースURL:https://jp.community.intersystems.com/post/intersystems-cach%C3%A9%E3%81%AE%E3%83%9E%E3%82%AF%E3%83%AD