Code Template Framework Guide

by SparxSystems Japan

Enterprise Architect 日本語版

コードテンプレートフレームワーク 機能ガイド

基礎編

(2018/05/16 最終更新)

1.はじめに

Enterprise Architect には、コードテンプレートフレームワーク(以下 CTF と表記します)と呼ばれる機能が搭載されています。このドキュメントでは、この CTF の基本的な内容について説明します。

この CTF に関する説明は、以下の 4 つに分割して行います。

- 基礎編(本ドキュメント)
  CTF の概念の説明・サンプルを通した機能の確認
- 応用編
  既存のテンプレートの修正
  (ステレオタイプを指定したテンプレートの追加)
- 発展編
  Enterprise Architect の対応していない独自のプログラム言語のソースコード生成
- 振る舞い図からのコード生成編
  ステートマシン図など、振る舞い図からのソースコード生成時に役に立つ情報

2.CTFとは

CTF とは、UML モデルのクラス図からソースコードを生成するための仕組みです。この CTF を利用することで、ユーザーがそれぞれ希望する形のソースコードを生成することができます。

もし、ソースコードの生成結果をカスタマイズできない場合、以下のような問題が発生します。

- 開発する部署で、ヘッダやコメントに関する独自のルールがある場合に、生成後に編集する必要がある
- コーディングルールがある場合に、そのルールに適合する形の出力結果に変更できない
- 開発する部署独自の解釈(あるステレオタイプは、xxx とみなしてソースコードに反映する、等)ができない

これらの問題を解決するために、CTF が導入されました。

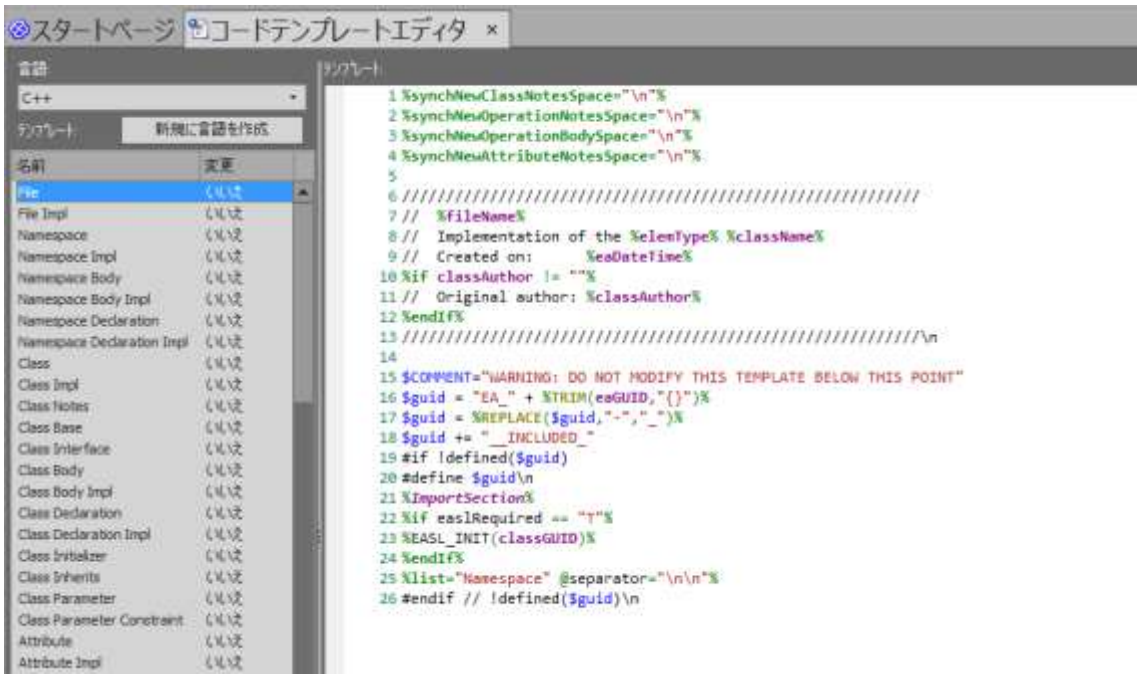
3.CTFを構成する要素

CTF は以下のものから構成されています。

- コードテンプレートエディタ
- デフォルトテンプレート
- 編集済み(追加)テンプレート
- マクロ
- 変数

3.1. コードテンプレートエディタ

コードテンプレートエディタは、「コード」リボン内の「設定」パネルにある「ソースコード」ボタンを押すと表示されるメニューから、「コード生成テンプレート」を選択すると呼び出すことができます。この項目を選択すると、以下のようにタブが追加され、コードテンプレートエディタが利用可能な状態になります。



左側のテンプレート一覧には、Enterprise Architect で定義されているテンプレートが含まれ、この一覧から選択すると右側の編集領域に選択したテンプレートの内容が表示され

ます。

3.2. 既定のテンプレート

Enterprise Architect では、Enterprise Architect がサポートしているプログラム言語(C++, C#, C, Java, Visual Basic, VB.NET, Delphi, PHP, ActionScript, Python。さらに、ユニファイド版およびアルティメット版では、SystemC, VHDL, Verilog, Ada)のソースコードを生成するために、既にテンプレートを持っています。また、それ以外にも Enterprise Architect の機能に関係するテンプレートもありますが、ここでは詳細を割愛します。

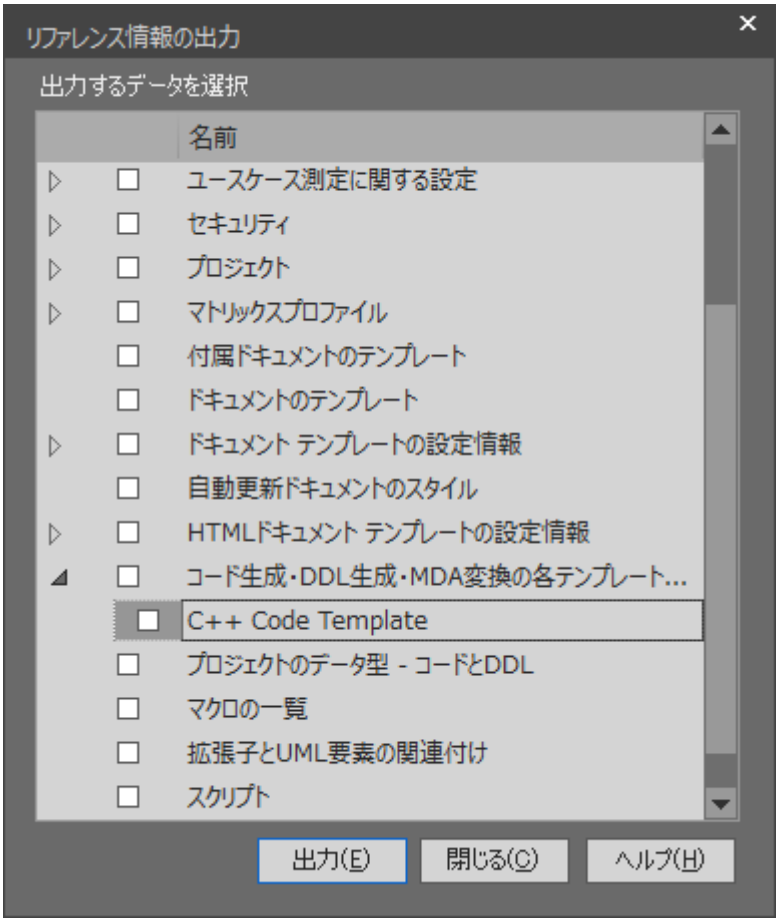
このように、Enterprise Architect が内部的に保持しているテンプレートを、既定のテンプレートと呼びます。通常はこの既定のテンプレートが利用されるため、この CTF を利用してテンプレートを作成しなくてもソースコードが生成できます。CTF をカスタマイズする場合には、この既定のテンプレートの内容が参考になります。

3.3. 編集済みテンプレート

ユーザーが希望するような形にテンプレートを編集すると、その結果(編集済みテンプレート)は現在開いているプロジェクトに保存されます。つまり、この編集済みテンプレートは、別のプロジェクトではそのまま利用できません。

作成したテンプレートを別のプロジェクトで利用する場合には、一度 XML ファイルとして作成した情報を出力する必要があります。「プロジェクト」リボン内の「ツール」パネルにある「転送」ボタンを押すと表示されるメニューから「リファレンス情報の出力」を実行すると、作成したテンプレートを出力することができます(この編集済みテンプレートがない場合には、一覧にはテンプレートは含まれません)。

下の例ですと、「コード生成 DDL 生成・MDA 変換の各テンプレート・CSV 入出力定義」のグループ内に、C++の編集済みテンプレートが「C++ Code Template」という名前で一覧に表示されています。



ここで出力した XML ファイルは「プロジェクト」リボン内の「ツール」パネルにある「転送」ボタンを押すと表示されるメニューから「リファレンス情報の読み込み」を実行することで、別のプロジェクトで読み込んで利用することができます。また、MDG テクノロジーファイルに含めることで、他の設定と合わせて 1 つの XML ファイルとして配布することもできます。

3.4. マクロ

テンプレートを定義するためには、例えばクラスの属性情報を読み出して、それがある値になっていれば関連の情報を出力する、といったようなモデル情報との連携や条件分岐などを記述する必要があります。こういった処理のすべてを受け持つのが、マクロです。

マクロには、いくつかの種類があります。

- テンプレート置換マクロ
- フィールド置換マクロ

- タグ付き値置換マクロ
- 制御マクロ
- 機能マクロ

ここでは、これらの詳細については説明しません。ヘルプファイルをご覧ください。

3.5. 変数

ソースコードを生成するためには、出力結果の文字列を操作したり、あるいはマクロの処理結果を保存しておいて後で利用したりする、といったような処理も必要になる場合があります。こういった場合に利用できるのが、変数です。

```
$hoge = "fuga"
```

この変数については、多くのプログラム言語の変数と概念は同じですので、これらの詳細については説明しません。必要に応じてヘルプファイルをご覧ください。

なお、CTF では全ての内容はいわゆる「文字列型」になります。数値型はありません。

4.テンプレートの簡単な編集

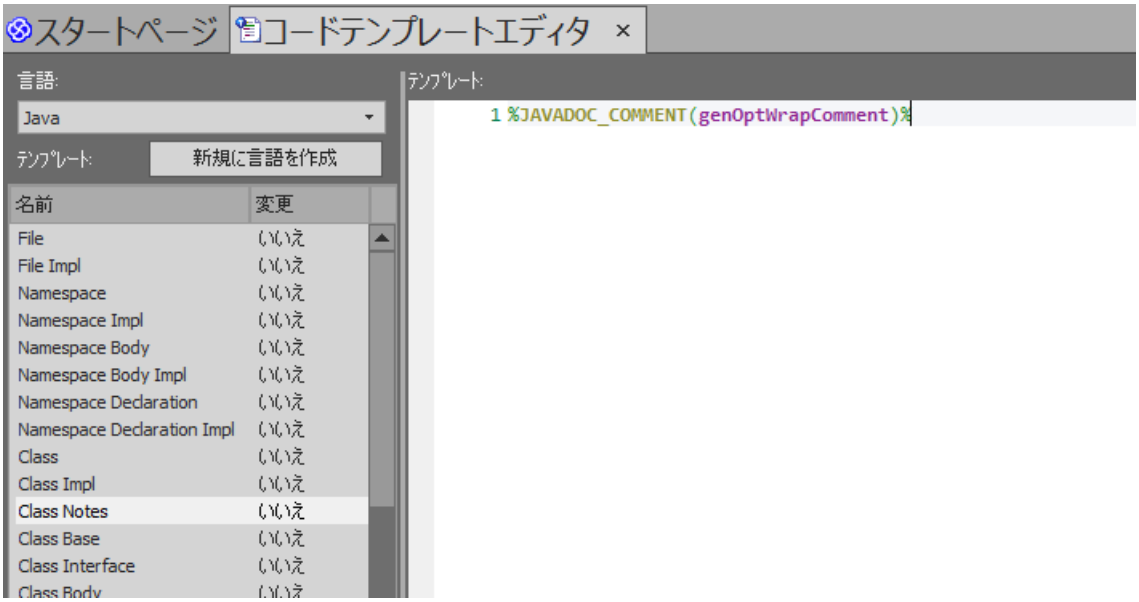
今まで CTF の概要を簡単に説明しましたが、CTF 自身や CTF の要素について説明するよりも、まずは実際に例を通して確認するほうがわかりやすいと思いますので、実際に CTF を利用してソースコードの生成を変えてみたいと思います。

具体的には、次のようなシナリオを考えます。

「VisualBasic.NET の出力で、コメントは JavaDoc 形式にする」

それでは、早速コードテンプレートエディタを起動します。今回は Java 言語の既定のテンプレートをコピーする形で実現するのが早いので、最初に Java のテンプレートの内容を確認します。

言語のコンボボックスから Java を、テンプレート一覧から Class Notes を選択します。この状態が下の画像です。



すると、右側のエディタ部分には、ただ 1 行、次のような行があります。

%JAVADOC_COMMENT(genOptWrapComment)%

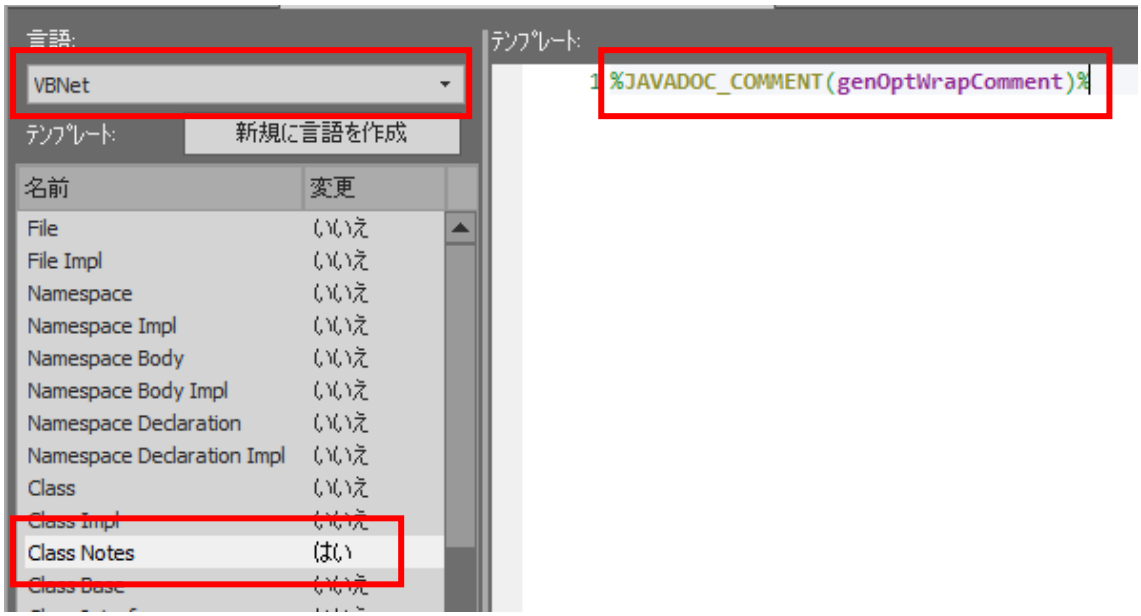
これが、先ほど説明したマクロです。マクロには引数を設定できるものがあり、今回の場合には genOptWrapComment が指定されています。この genOptWrapComment というのは、折り返しの文字数であり、Enterprise Architect 内部で定義済みのフィールド置換マクロです。このような定義済みのフィールド置換マクロは、ヘルプファイルの「フィールド置換マクロ」のページをご覧ください。

この 1 行をコピーしてください。今回は 1 行しか存在しないので、他の多くのエディタと同じように、Ctrl+A キーで全てを選択して、Ctrl+C キーでコピーすることができます。

その後、言語を今度は対象の VB.NET(画面での表示では VBNet)に変更し、同じく Class Notes テンプレートを選択します。すると、エディタ部分には

%XML_COMMENT(genOptWrapComment)%

と書かれています。これも、マクロです。名前のとおり、XML 形式のコメントを生成する機能マクロです。この部分に、先ほどコピーした内容を上書きします。キー入力であれば、Ctrl+A キーで全てを選択して、Ctrl+V キーで貼り付けです。貼り付けた後、エディタの下のほうにある「保存」ボタンを押した状態が、下の図の状態です。



このダイアログの赤枠で囲んだ部分が、この説明の中で操作した内容です。テンプレート一覧の中に、1 箇所「はい」と表示された部分があります。この部分は、テンプレートを変更したかどうかを示します。今回、Class Notes テンプレートを変更しましたので、この部分のみが「はい」になっていると思います。

これで、ソースコードを生成してみましょう。もちろん、出力する対象のクラスのプログラム言語を VB.NET に変更する必要があります。

すると、以下のように出力結果が変わるのがわかると思います。

・編集前

```
''' <summary>
''' クラスのコメントです。
''' </summary>
Public Class Block
```

(以下省略)

・編集後

```
/**
 * クラスのコメントです。
```

```
 * @author Geoffrey Sparks
 * @version 1.0
 * @created 10-9-2003 10:22:07
 */
Public Class Block
```

(以下省略)

今回は、クラスのコメント部分のみを変更しましたが、同様にして属性や操作のコメント部分も置換することで、JavaDoc 形式でコメントを出力することができるようになります。

5.テンプレートの内容の解釈について

実際にクラス要素に対してソースコード生成が実行されると、そのクラス要素のプログラム言語の指定に応じて、対応するテンプレートが参照され、ソースコードが生成されます。

この際に、文法間違いなど、もしテンプレートの内容に問題がある場合には、問題となる内容はそのまま文字列としてソースファイル内に出力されます。例えば、以下のような例を考えます。

```
%if $a <> ""%
output something
%endIf%
```

この内容の問題点は、不一致を示す記号として、「<>」を利用していることです。CTF では「！=」が不一致を示す記号になります。ですので、この内容には間違いが含まれます。

この内容のまま出力した場合には、以下のような結果になります。

```
%if   <> ""%
output something
```

これは、以下のような処理が行われています。

1. If 文の中に含まれる「<>」が解釈できないため、この行全体を通常の文字列として扱うことになる

2. 通常の文字列として扱う場合、$a は変数なので、変数の内容(この例では空文字列)に置換して出力する
   →この結果、1 行目の出力結果となる
3. 2 行目は CTF の処理が含まれない(%記号や$変数が存在しない)ので、そのまま文字列として出力する
4. 3 行目は文法上正しいが、対応する%if がこれより前に存在しない(=1 行目は文法間違いがあったので、if は無視されている)ので、無視する
   →何も出力されない

ですので、いわゆる「エラーメッセージ」のようなものは表示されないことに注意して下さい。上記の様に Enterprise Architect が解釈できなかった内容はすべて、文字列としてソースファイルに出力されます。

6.最後に

CTF について簡単に説明しましたが、いかがでしたでしょうか？覚えることは多少ありますが、慣れてくれば希望通りの出力結果を生成することができるようになると思います。ヘルプファイルではマクロの説明も書かれていますので、ぜひ参考にしてください。

○改版履歴

2007/01/09 表現を見直し、わかりやすい文章に変更
2008/03/06 説明を一部追加
2009/03/24 バージョン 7.5 のリリースに伴い、内容を更新。
2009/08/31 ドキュメントのタイトルを変更。
2011/05/18 バージョン 9.0 のリリースに伴い、内容を更新。
2012/12/14 バージョン 10.0 のリリースに伴い、内容を更新。
2013/01/24 第 5 章を追加。いくつかの内容の補足を追加。
2014/04/22 バージョン 11.0 のリリースに伴い、内容を更新。
2015/02/12 バージョン 12.0 のリリースに伴い、内容を更新。
2015/12/01 バージョン 12.1 のリリースに伴い、内容を更新。
2016/10/07 バージョン 13.0 のリリースに伴い、内容を更新。
2018/05/16 バージョン 14.0 のリリースに伴い、内容を更新。