Jarosław Kuchta

Model Gen

Opis techniczny

data opracowania: 30.05.2023

# Wprowadzenie

ModelGen to narzędzie do generowania kodu modelu dokumentu na podstawie biblioteki DocumentFormat.OpenXml. Generuje kod klas (typów) elementów modelu w katalogu DocumentModel oraz kod konwerterów klas modelu z/do formatu OpenXML w katalogu DocumentModel.OpenXml.

Narzędzie występuje w postaci rozwiązania C# z dwoma projektami:

* biblioteką ModelGen realizującą proces generowania kodu,
* programem uruchomieniowym ModelGenRun.

## Program uruchomieniowy

Program ModelGenRun zawiera dwie procedury:

* GenerateModelTypes – do generowania typów modelu,
* GenerateTypeConverters – do generowania konwerterów typów.

Obie procedury mają podobną strukturę:

1. Tworzą podstawową ścieżkę plików wynikowych na podstawie lokalizacji rozwiązania.
2. Tworzą odpowiedni kreator dla klas wynikowych:
   * ModelCreator – kreator modelu,
   * ConverterCreator – kreator konwerterów klas modelu na elementy formatu OpenXml.
3. Uruchamiają kreator dla podanego typu z biblioteki DocumentFormat.OpenXml, który to typ będzie służył za typ główny (korzeń w drzewie analizy).

Aby wygenerować pełny model dokumentu tekstowego należy podać typ DocumentFormat.OpenXml.Packaging.WordprocessingDocument.

## Proces generowania kodu

Proces generowania kodu, zarówno typów modelu, jak i konwerterów typów, składa się z sześciu kroków:

1. Skanowanie typów – typy publiczne z biblioteki DocumentFormat.OpenXml są przeglądane i rejestrowane rekurencyjnie metodą refleksji typów począwszy od podanego typu głównego. Dla każdej klasy przeglądane i rejestrowane są typy właściwości publicznych. Dodatkowo dołączane są typy odczytywane ze schematu typów biblioteki, gdzie mogą występować typy elementów składowych nieuwzględnione w deklaracji klas biblioteki. Rejestrowane są również typy wyliczeniowe i typy strukturalne.
2. Zmiana nazw typów – zarejestrowane typy mają zmieniane nazwy wynikowe tak, aby dopasować nazwy typów generowanych do konkretnego kreatora.
3. Konwertowanie typów – ustalane jest odwzorowanie części typów na typy podstawowe – typy systemowe C# lub typy bazowe modelu zdefiniowane uprzednio.
4. Sprawdzanie użycia typów – tylko typy wykorzystywane w innych typach będą generowane. Niektóre typy zostaną pominięte ze względu na konwersję innych typów na typy podstawowe.
5. Walidowanie typów i przestrzeni nazw – typy przeznaczone do generowania są sprawdzane pod względem unikatowości nazw w przestrzeniach nazw. Dokonywane są niezbędne korekty.
6. Generowanie typów wynikowych – konkretny kreator generuje kod odpowiednich typów (interfejsów, klas, struktur, typów wyliczeniowych) umieszczany w plikach o odpowiednich nazwach w katalogach tworzonych w ścieżce plików wynikowych na podstawie przestrzeni nazw.

Pierwsze pięć kroków jest wspólnych dla obu kreatorów, ostatni jest realizowany w każdym kreatorze osobno.

# Konstrukcja oprogramowania

W bibliotece ModelGen zdefiniowano dwie główne klasy kreatorów kodu, które analizują typy zdefiniowane w bibliotece DocumentFormat.OpenXml i generują kod klas wynikowych. Są to

* ModelCreator – kreator klas (i innych typów) modelu,
* ConverterCreator – kreator klas konwertujących klasy modelu na elementy formatu OpenXml.

## Konwerter bazowy

Oba kreatory są zdefiniowane na bazie abstrakcyjnej klasy BaseConverter, która zawiera metodę RunOn(Type) organizującą proces analizy i generowania klas modelu oraz metody implementujące kolejne kroki tego procesu. Są to (w kolejności wywoływania):

* 1. ScanType(Type) – skanująca i rejestrująca typy
  2. RenameTypes – zmieniająca nazwy typów,
  3. AddTypeConversions – ustalająca konwersje niektórych typów na typy proste i typy bazowe.
  4. CheckTypeUsage – sprawdzająca użycie typów i zaznaczająca te typy, które będą generowane,
  5. ValidateTypes – sprawdzająca poprawność typów do generowania i dokonująca korekty w razie potrzeby,
  6. GenerateCode – abstrakcyjna metoda generująca kod. Ta metoda jest implementowana inaczej w każdej z klas potomnych.

Każda z tych metod zwraca wynik typu TimeSpan reprezentująca czas wykonania procedury.

Pomiędzy kolejnymi krokami procesu mogą być wywoływane metody monitorujące proces. Sterują tym dwa opcjonalne parametry metody RunOn:

* monitorDisplaySelector (typu wyliczeniowego MDS),
* originTargetSelector (typu wyliczeniowego OTS).

Flagi MDS wybierają krok, po którym wyświetlane są wyniki analizy:

* ScannedNamespaces – po pierwszym kroku wyświetlane jest podsumowanie zeskanowanych przestrzeni nazw,
* TypeRenames – po drugim kroku wyświetlane są relacje zmiany nazwy,
* TypeConversions – po trzecim kroku wyświetlane są relacje konwersji typów,
* TypeUsage – po czwartym kroku wyświetlane są szczegóły przestrzeni nazw przeznaczonych do generowania.
* ValidatedTypes – po piątym kroku wyświetlane są szczegóły sprawdzonych i skorygowanych przestrzeni nazw.

Przy wyświetlaniu przestrzeni nazw możliwy jest wybór przestrzeni oryginalnych, docelowych lub systemowych poprzez parametr typu OTS. Brak podania parametru powoduje wybór typu:

* Origin – oryginalnych przestrzeni nazw rozpoczynających się od „DocumentFormat” (po pierwszym kroku),
* Target – docelowych przestrzeni nazw rozpoczynających się od „DocumentModel” (po czwartym i piątym kroku).

Możliwy jest jeszcze wybór OTS:

* System – systemowych przestrzeni nazw rozpoczynających się od „System”.

## Monitorowanie procesu

Do monitorowania procesu jest wykorzystywana klasa ModelDisplay. Klasa ta działa w oparciu o Writer (typu IndentedTextWriter), który domyślnie jest ustawiony na konsolę