nAxiom

Pętle podprocesów asynchronicznych w nAxiom

Wersja nAxiom: 1.13.2.0

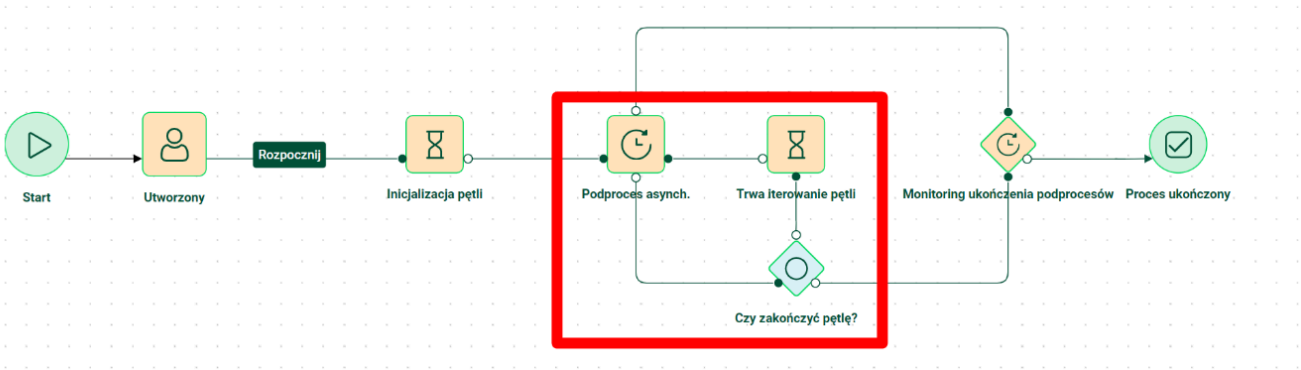
Spis treści

1. Kluczowe informacje

- Blok podprocesu można umieścić w pętli, to jest poprowadzić wyjście procesu głównego przez blok decyzyjny i ewentualnie inne bloki z powrotem na wejście do bloku podprocesu.
- Taka pętla może być iterowana automatycznie, na podstawie warunku w bloku decyzyjnym, lub ręcznie, przez użycie bloku zadania ręcznego z dwoma wyjściami: do podprocesu i do synchronizatora (koniec pętli).
- Przed wejściem do bloku podprocesu synchronicznego (także w pętli) zaleca się użycie bloku oczekiwania 0 s.

2. Budowa pętli

Schemat wywoływania podprocesu asynchronicznego w pętli iterowanej automatycznie przedstawia poniższy diagram.



Przykład wywołania podprocesu w pętli

Pętla składa się z następujących elementów:

- blok podprocesu asynchronicznego: wyjście procesu głównego poprowadzone do bloku decyzyjnego, a wyjście statusu podprocesu do synchronizatora,
- blok decyzyjny: decyduje o kontynuacji pętli (przejście do bloku oczekiwania) lub jej zakończeniu (przejście do synchronizatora),
- blok oczekiwania 0 s: jego użycie na wejściu do bloku podprocesu asynchronicznego wyłącza blokadę ekranu (ikona operacji w toku), zamyka poprzednią transakcję i otwiera nową (każdy dokument podprocesu jest powoływany w osobnej transakcji) oraz pozwala zaprezentować odpowiedni status w procesie głównym (wymagane ręczne odświeżenie formularza).

UWAGA

Należy pamiętać, że wyjście procesu głównego z bloku podprocesu asynchronicznego może prowadzić tylko do bloku decyzyjnego lub do bloku synchronizatora. Aby wywołać podproces asynchroniczny w pętli iterowanej ręcznie, czyli z zastosowaniem zadania ręcznego, konieczne jest użycie bloku decyzyjnego, jako *proxy*.

3. Transakcyjność przetwarzania w pętli

Użycie bloku oczekiwania w pętli przed powrotem do bloku podprocesu asynchronicznego powoduje, że każda iteracja pętli, czyli każde utworzenie dokumentu podprocesu, będzie wykonywane w osobnej transakcji. Jeśli podczas tworzenia dokumentu podprocesu wystąpi błąd, wykonywanie pętli zostanie przerwane, ale nie wpłynie to na dokumenty utworzone do momentu wystąpienia błędu.

W przypadku pętli bez bloku oczekiwania, błąd spowoduje przerwanie pętli i wycofanie transakcji, to znaczy, że wszystkie dokumenty utworzone do momentu wystąpienia błędu zostaną usunięte.

Kiedy użytkownik klika przycisk zmiany statusu, otwierana jest transakcja, która kończy się z chwilą osiągnięcia następnego statusu. W czasie wykonywania transakcji wyświetlana jest ikona operacji w toku, która blokuje okno przeglądarki. Ponieważ wywołanie podprocesu asynchronicznego, a szczególnie w pętli, może trwać pewien czas, taka blokada obniża komfort obsługi aplikacji. Użycie bloku oczekiwania powoduje zamknięcie transakcji zainicjowanej przez

użytkownika i wznowienie jej przez aplikację *taskservice* po upływie ustawionego czasu. W tym przypadku nAxiom nie blokuje ekranu, więc użytkownik może wykonywać inne czynności, podczas gdy inicjowanie podprocesów odbywa się w tle.

Dodatkowo status bloku oczekiwania można wykorzystać do informowania użytkownika o stanie przetwarzania pętli podprocesów. Należy jednak pamiętać, że wyświetlenie statusu bloku oczekiwania wymaga ręcznego odświeżenia formularza.

4. Przykładowe rozwiązania projektowe

Do obsługi przetwarzania podprocesów w pętli może być przydatna tabela pomocnicza, taka jak opisana w artykule Projektowanie podprocesów w nAxiom. W tym przypadku będzie ona wcześniej wypełniana danymi dla dokumentów, które dopiero zostaną utworzone. W tabeli zapisywany będzie identyfikator procesu głównego i unikatowy identyfikator każdego podprocesu oraz flaga, która będzie zmieniana po każdej iteracji.

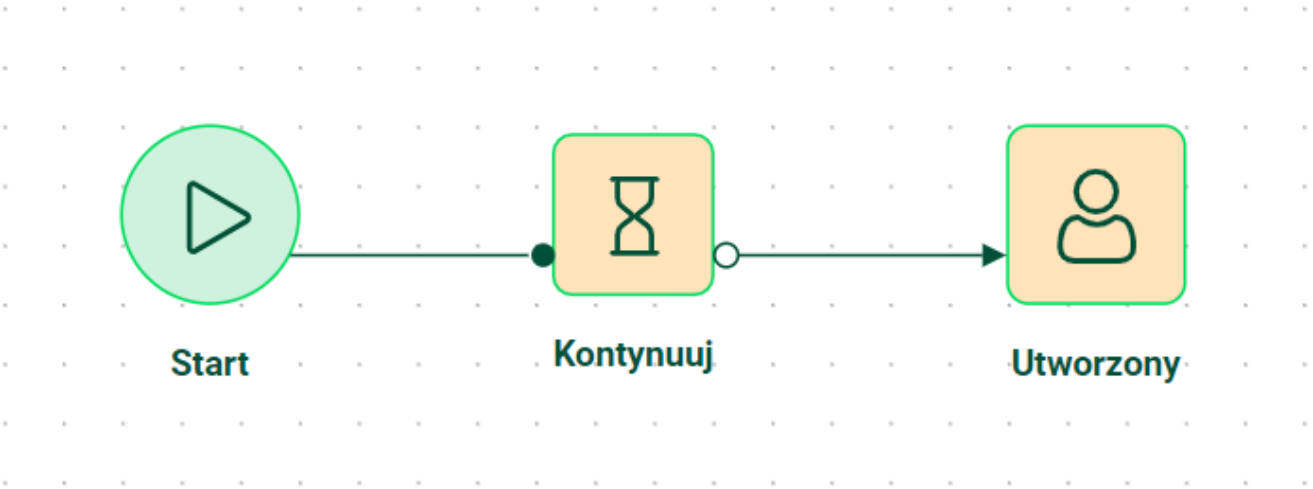
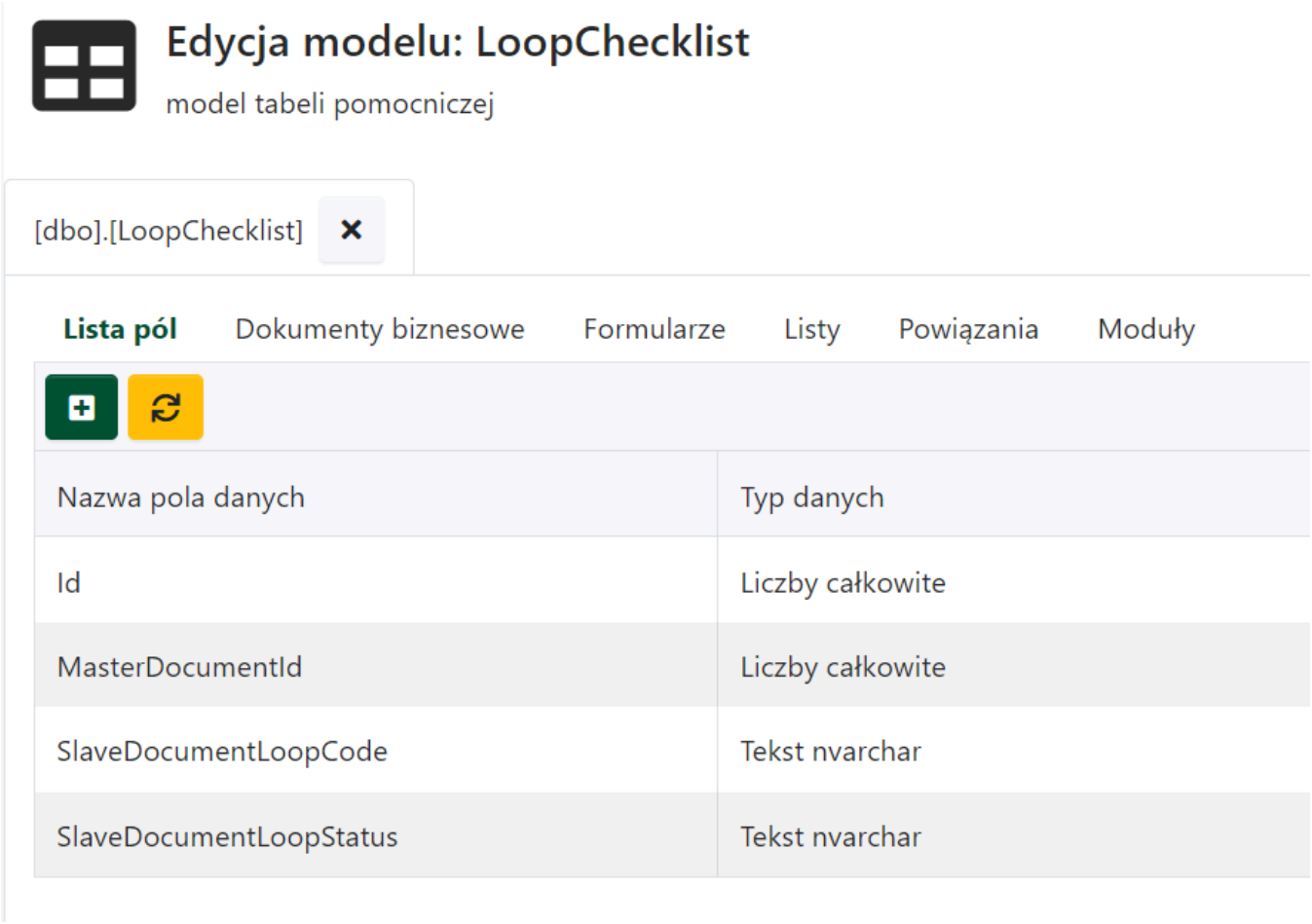


Tabela pomocnicza używana do sterowania pętlą

Poprzez mapowanie zmiennych z procesu głównego do podprocesu przekazany zostanie identyfikator procesu głównego (*MasterDocumentId*) oraz pobrany z tabeli pomocniczej identyfikator dokumentu w podprocesie (*LoopCode*).



Mapowanie zmiennych do podprocesu - obsługa pętli

Proces główny musi zostać skonstruowany tak, aby przed rozpoczęciem zapisać w tabeli pomocniczej dane, które będą potrzebne do sterowania pętlą. Liczba podprocesów jest określana na podstawie wartości podanej w polu formularza, z którego użytkownik inicjuje pętlę podprocesów (*{@LoopSize}*). Dla każdego obiegu pętli generowany jest identyfikator podprocesu oraz ustawiana jest flaga *waiting*.

```
DECLARE @i INT = 1;
DECLARE @maxIterations INT = {@LoopSize};

WHILE @i <= @maxIterations
BEGIN
  INSERT INTO dbo.[LoopChecklist] (
    [MasterDocumentId]
```

```
   ,[SlaveDocumentLoopCode]
   ,[SlaveDocumentLoopStatus]
   )
  VALUES (
    {@Id}
   ,concat({@Code},'_',@i)
   ,'waiting'
   );
SET @i += 1;
END;
```

Rekordy tej tabeli będą aktualizowane w każdym zainicjowanym podprocesie akcją SQL, która zmieni flagę na *created*, kiedy dokument w podprocesie zostanie utworzony.

```
UPDATE [dbo].[LoopChecklist]
SET SlaveDocumentLoopStatus = 'created'
WHERE MasterDocumentId = {@MasterDocumentId}
  AND SlaveDocumentLoopCode = {@LoopCode}
```

W kolejnym kroku proces główny przejdzie do bloku decyzyjnego, który sprawdzi w tabeli pomocniczej, czy dla danego identyfikatora procesu głównego istnieją jeszcze rekordy z flagą *waiting* i odpowiednio wybierze kierunek przejścia.

```
DECLARE @counter int

SET @counter = (
  SELECT count(Id) from [dbo].[LoopChecklist]
  WHERE MasterDocumentId = {@Id}
    AND SlaveDocumentLoopStatus = 'waiting'
  )

IF @counter > 0
  SELECT 'NextDocument'
ELSE
  SELECT 'EndLoop'
```

Tabelę pomocniczą można również wykorzystać w konstrukcji warunku synchronizacji.

```
DECLARE @createdDocuments int
DECLARE @thisProcessDocuments int

SET @createdDocuments = (
  SELECT count(Id) from [dbo].[LoopChecklist]
  WHERE MasterDocumentId = {@Id}
    AND SlaveDocumentLoopStatus = 'created'
  )
SET @thisProcessDocuments = (
  SELECT count(Id) FROM [dbo].[LoopChecklist]
  WHERE MasterDocumentId = {@Id}
  )
IF @createdDocuments = @thisProcessDocuments
  SELECT 1
ELSE
  SELECT 0
```

Z synchronizatora prowadzi tylko jedno przejście, więc do wskazania go wystarczy prosta instrukcja.

```
SELECT `FinishProcess`
```