WEBCON

BPS 2020.1

Zmiany w SDK i API

1. Wprowadzenie

Spis zmian w SDK i API w stosunku do wersji 2019.1. Wypisane są zmiany, które łamią kompatybilność kodu lub zmieniają zachowanie istniejącego kodu.

2. Zmiany w paczkach dodatków

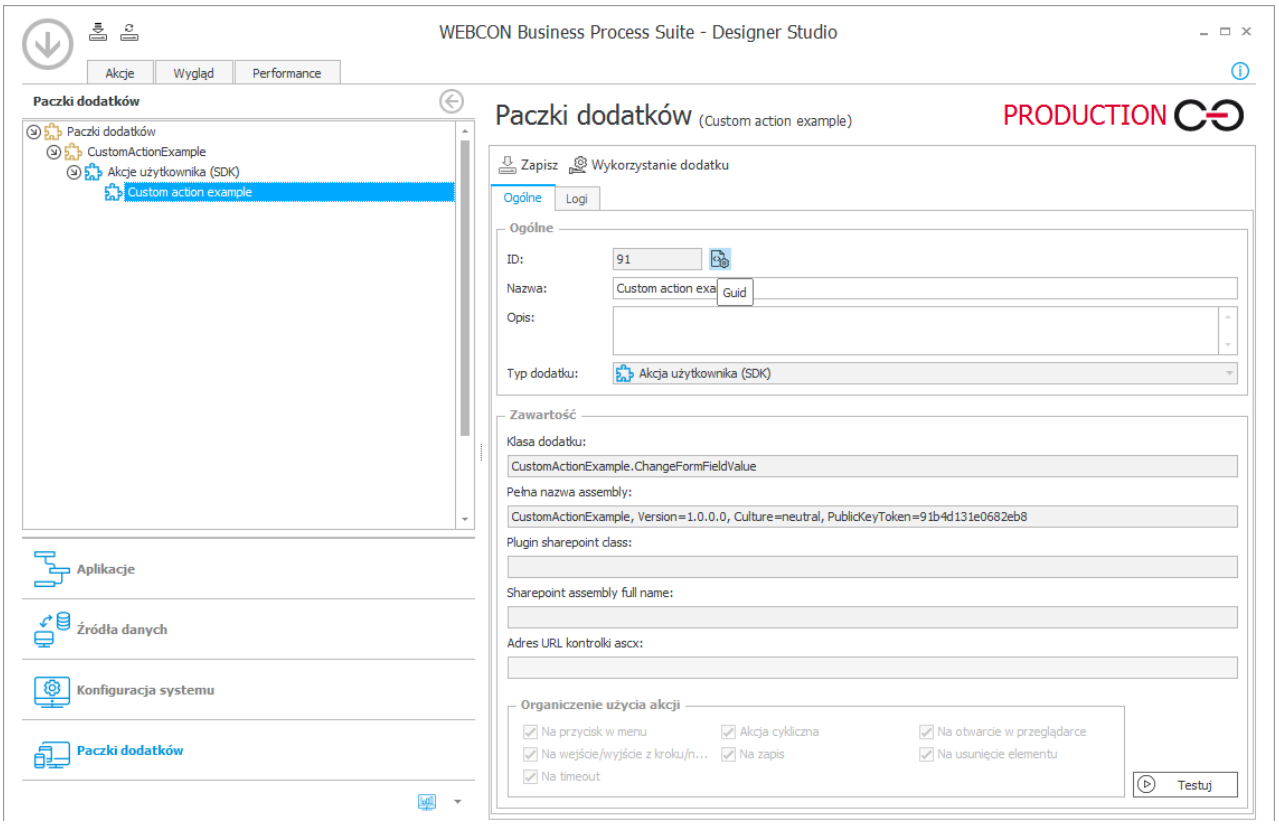
Zmiany paczek dodatków dotyczą migracji z wersji niższej niż 2019.1.4 i wynikają z prowadzenia w manifeście paczki dodatków pola: Guid.

W zależności od wersji z której migrujemy WEBCON BPS możemy mieć do czynienia z dwoma scenariuszami.

2.1. Migracja z wersji niższej niż 2019.1

Pierwszy z nich ma miejsce podczas migracji dodatków z wersji BPS, w których pojęcie paczek dodatków nie istniało np. z BPS 2017. Zaraz po udanej aktualizacji wersji, automatycznie tworzone są puste paczki dodatków. Dla każdej z nich konieczne będzie przejście przez proces migracji każdego dodatku. Proces ten jest opisany w dokumencie migracyjnym dla BPS 2019.1.

Dla każdego zarejestrowany dodatku, konieczne będzie dodanie w manifeście paczki dodatków wpisu z jego danymi, dodatkowo zawierający identyfikator GUID znajdujący się w bazie danych. Najprostszym sposobem na zdobycie informacji o nim jest wykorzystanie WEBCON BPS Designer Studio tak, jak zostało to pokazane na obrazku poniżej:

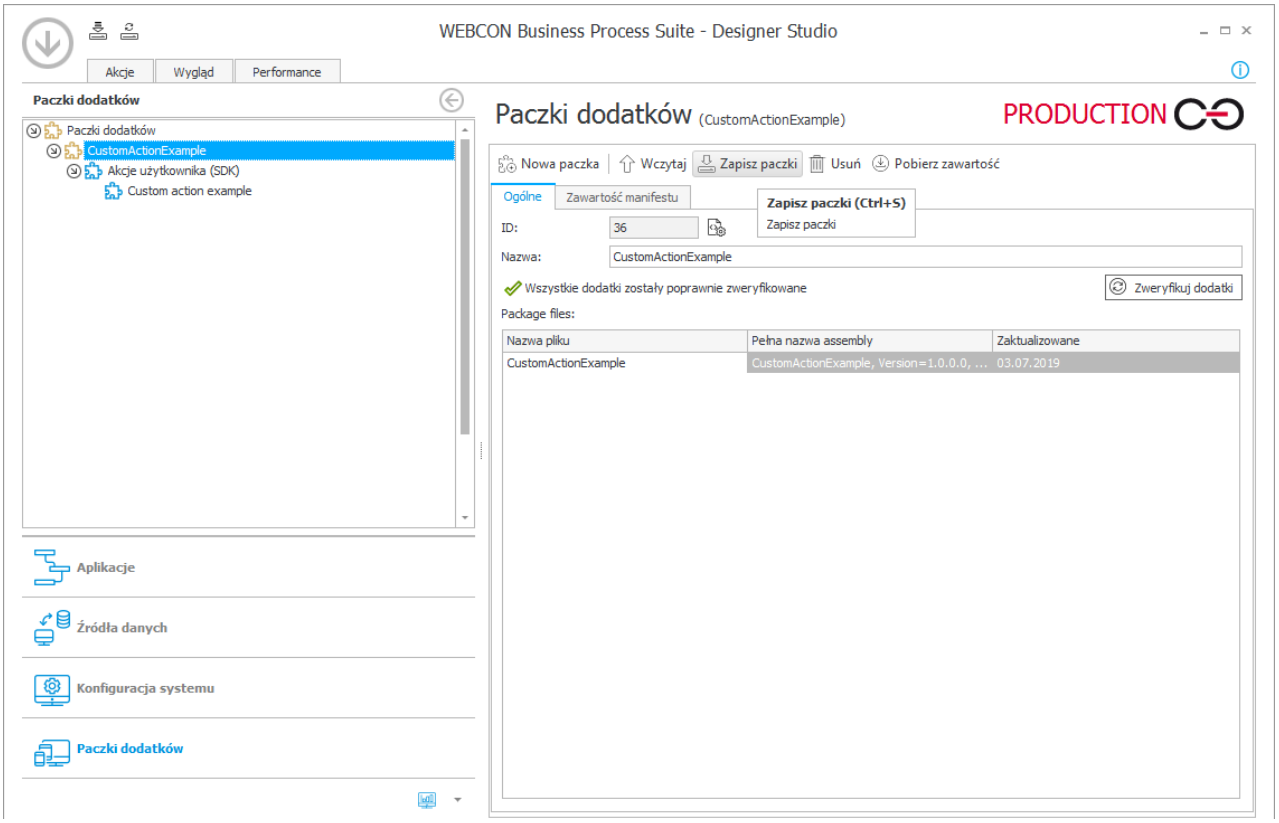


Inaczej jest, gdy podczas migracji tworzymy nowe, nie rejestrowane wcześniej dodatki. W tym przypadku dla każdego wpisu w manifeście paczki, konieczne będzie wygenerowanie unikalnego guida.

Jednym ze sposobów jest wykorzystanie do tego celu narzędzia Windows PowerShell i wykonanie statycznej metody [guid]::NewGuid(). Następnie wystarczy skopiować wygenerowaną wartość, umieścić ją we właściwym miejscu w węźle odpowiadającym migrowanemu dodatkowi w manifeście i gotowe.

2.2. Migracja z wersji od 2019.1 do 2019.4

Drugi typ migracji ma miejsce, gdy wykonujemy ten proces dla dodatków z wersji BPS 2019.1 lub wyższej. Dla zmigrowanych i zaktualizowanych wcześniej paczek dodatków uruchamiany jest skrypt migracyjny, który automatycznie aktualizuje istniejące paczki z dodatkami, aby były one kompatybilne z najnowszą wersją. Dzięki tej operacji jedyne co trzeba zrobić po instalacji najnowszej wersji jest zapisanie paczek z dodatkami wykorzystując funkcjonalność zapisu w WEBCON Business Process Suite – Designer Studio, wyciągnięcie z nich plików manifestu i skopiowanie go do powiązanego projektu w Visual Studio. Po wykonaniu operacji dodatki będą gotowe do użycia.



2.3. Uwspólnianie identyfikatorów dodatków na kilku bazach WEBCON BPS

W przypadku dodatków zarejestrowanych na kilku powiązanych bazach WEBCON BPS (DEV, TEST, PROD) będą istnieć manifesty z różnymi identyfikatorami tego samego dodatku np.

Dla bazy TEST:


```
[
 {
  "Guid": "10bd08ad-a9a3-4004-91a9-94167fe3ee9e",
  "Name": "Custom action example",
  "Assembly": "CustomActionSample",
  "Class": " CustomActionSample.ChangeFormFieldValue",
  "Type": "CustomAction"
 }
]
```


Dla bazy PROD:


```
[
 {
  "Guid": "760b455f-8a99-4469-95b9-dba0ded9e667",
  "Name": "Custom action example",
  "Assembly": "CustomActionSample",
  "Class": " CustomActionSample.ChangeFormFieldValue",
  "Type": "CustomAction"
 }
]
```


Proces podmiany guidów na bazie TEST na wartości znajdujące się w bazie PROD składa się z dwóch kroków.

- Modyfikacji manifestu bazy PROD i dodanie dla każdego z migrowanych dodatków pola AlternativeGuid uzupełnionego wartością Guid tego dodatku z bazy TEST.

Poniżej docelowy manifest:


```
[
 {
  "Guid": "760b455f-8a99-4469-95b9-dba0ded9e667",
  "AlternativeGuid ": "10bd08ad-a9a3-4004-91a9-94167fe3ee9e",
  "Name": "Custom action example",
  "Assembly": "CustomActionSample",
  "Class": " CustomActionSample.ChangeFormFieldValue",
```

```
  "Type": "CustomAction"
 }
]
```


- Tak przygotowany manifest wraz z paczką dodatków powinien zostać zaktualizowany na bazie TEST. Możemy tego dokonać poprzez wczytanie nowej paczki ze zmienionym manifestem w WEBCON BPS Designer Studio.

Zarejestrowane dodatki zostaną dopasowane do odpowiednich wpisów w manifeście na podstawie wartości w polu AlternativeGuid, a następnie wartość ta zostanie przepisana do pola Guid. Od tej pory dodatki na obydwu bazach będą miały takie same identyfikatory, a wartość AlternativeGuid będzie już ignorowana.


```
[
 {
  "Guid": "760b455f-8a99-4469-95b9-dba0ded9e667",
  "Name": "Custom action example",
  "Assembly": "CustomActionSample",
  "Class": " CustomActionSample.ChangeFormFieldValue",
  "Type": "CustomAction"
 }
]
```


W przypadku większej liczby baz np. dodatkowej bazy DEV proces uwspólniania guidów dodatków wykonujemy analogicznie.

3. Zmiany w SDK

3.1. DataSourcesHelper – pobieranie danych pola i kolumny listy pozycji

Z klasy określającej parametry pobierania danych z atrybutu, oraz kolumny listy pozycji (`GetDataTableFromFieldDataSourceParams`, `GetDataTableFromItemsListColumnDataSourceParams`), została usunięta właściwość: `GetDataTableRunningContextParams`.

Została ona zastąpiona właściwością `Context`.

Można do niej przekazać kontekst w którym uruchomiany jest dany dodatek, lub utworzyć nowy kontekst procesu, dokumentu, lub wiersza listy pozycji.

- Kod pobierający dane z atrybutu, było:

```
var fieldDataSourceParams = new GetDataTableFromFieldDataSourceParams();
fieldDataSourceParams.FieldId = formFieldID;

fieldDataSourceParams.GetDataTableRunningContextParams = new
GetDataTableRunningContextParams(args.Context); //current context
//fieldDataSourceParams.GetDataTableRunningContextParams = new
GetDataTableRunningContextParams(documentInstance); //context of a specific document

var dt = DataSourcesHelper.GetDataTableFromFieldDataSource(fieldDataSourceParams);
```

Jest:

```
var fieldDataSourceParams = new GetDataTableFromFieldDataSourceParams();
fieldDataSourceParams.FieldId = formFieldID;

fieldDataSourceParams.Context = args.Context; //current context
//fieldDataSourceParams.Context = new ProcessContext(processID); //context of the specific
process
//fieldDataSourceParams.Context = new ProcessContext(documentInstance); //context of the
specific document

var dt = DataSourcesHelper.GetDataTableFromFieldDataSource(fieldDataSourceParams);
```

- Kod pobierający dane z kolumny listy pozycji, było:

```
var itemsListDataSourceParams = new GetDataTableFromItemsListColumnDataSourceParams();
itemsListDataSourceParams.ItemsListColumnID = itemsListColumnID;

itemsListDataSourceParams.GetDataTableRunningContextParams = new
GetDataTableRunningContextParams(args.Context); //current context
//itemsListDataSourceParams.GetDataTableRunningContextParams = new
GetDataTableRunningContextParams(documentInstance); //context of a specific document

itemsListDataSourceParams.ItemsListRow = itemsListRow;
var dt = DataSourcesHelper.GetDataTableFromSubelementDataSource(itemsListDataSourceParams);
```

Jest:

```
var itemsListDataSourceParams = new GetDataTableFromItemsListColumnDataSourceParams();
itemsListDataSourceParams.ItemsListColumnID = itemsListColumnID;

itemsListDataSourceParams.Context = args.Context; //current context; if it contains a items list row you don't need to provide it separately
/itemsListDataSourceParams.Context = new ProcessContext(documentInstance, itemsListRow);
//context of the specific document and items list row

var dt = DataSourcesHelper.GetDataTableFromSubelementDataSource(itemsListDataSourceParams);
```

3.2. Kontrolka użytkownika - zmiana nazwy metody

W kontrolce użytkownika, część interfejsowa SharePoint bez własnego modelu danych (klasa `CustomFormFieldSPControl<TConfig>`) zmieniona została nazwa metody z

```
void SetValue(SetControlParams<CustomFormFieldContext> args)
```

na

```
void SetControlValue(SetControlParams<CustomFormFieldContext> args)
```

3.3. Część logiczna - zmiana nazwy klasy

Zmieniona została nazwa klasy z `SPConnection` na `SharePointConnection` (namespace `WebCon.WorkFlow.SDK.Tools.Data.Model`)

3.4. Część logiczna - zmiany przestrzeni nazw

Zmienione zostały namespace:

Nazwa poprzednia	Nazwa obecna
WebCon.WorkFlow.SDK.Documents.ItemsLists	WebCon.WorkFlow.SDK.Documents.**Model**.ItemsLists
WebCon.WorkFlow.SDK.Documents.Fields	WebCon.WorkFlow.SDK.Documents.**Model**.Fields

3.5. Niestandardowe źródło danych – zmiany pobierania ID procesu.

Dla dodatków typu: niestandardowe źródło danych, napisanych w wersji poniżej 2019.1.3, w przypadku odwołania się do ProcessID konieczne jest zmigrowanie kodu według poniższego przykładu.

Było:

```
public override DataTable GetData(SearchConditions searchConditions)
{
            var processId = Context.CurrentProcessID;

            …
 }
```

Jest:

```
public override DataTable GetData(SearchConditions searchConditions)
{
            var processId = Context.CurrentProcessID.HasValue ?
            Context.CurrentProcessID.Value : 0;

            …
}
```

4. Zmiany w REST API

4.1. Nowa wersja REST API 2.0

W wersji 2020.1 została dodana nowa wersja REST API 2.0

4.2. Zmiana przekazywana ścieżki przejścia

W wersji Beta w stosunku do wersji 2019.1 został zmieniony sposób parametry przekazujący ścieżkę przejścia, było:

POST, PUT */api/data/v2.0/db/{dbId}/elements? pathId={pathID}*

POST, PUT */api/data/v2.0/db/{dbId}/elements? pathGuid={ pathGuid}*

POST, PUT */api/data/v2.0/db/{dbId}/elements? defaultPath =1*

Jest:

POST, PUT */api/data/v2.0/db/{dbId}/elements? path={path}*

Jako wartość path podajemy id, guid lub wartość ‘default’.

Taki sposób przekazywania ścieżki przejścia przyjęto również wersji REST API 2.0.