Крис Уэбб. Параметры RelativePath и Query функции M Power Query Web.Contents()

Это перевод нескольких связанных статей Криса Уэбба, дополненный моими комментариями с использованием Chat GPT (набраны с отступом).

Функция Web.Contents() в языке M Power Query позволяет получать данных с веб-страниц и веб-служб и имеет ряд полезных, но плохо документированных опций, которые упрощают создание URL-адресов для вызовов веб-служб.

Рассмотрим следующий URL-адрес:

https://data.gov.uk/api/3/action/package_search?q=cows

Это вызов API метаданных с сайта https://www.data.gov.uk/, портала открытых данных правительства Великобритании. По этому адресу возвращается документ JSON (рис. 1), в котором перечислены наборы данных, найденные при поиске по ключевому слову *cows* (коровы).

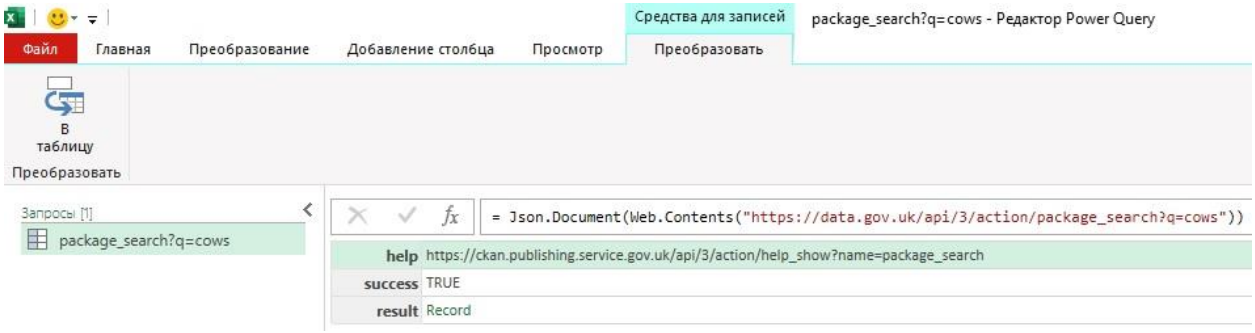

Рис. 1. Документ JSON, возвращаемый по Запросу 1

Этот вызов можно сделать с помощью функции Web.Contents():

Запрос 1[1]

```
Web.Contents("https://data.gov.uk/api/3/action/package_search?q=cows")
```

Вместо того, чтобы использовать одну длинную строку для URL-адреса, можно создать её налету с помощью параметров *RelativePath* и *Query*. RelativePath добавляет текст к базовому URL-адресу, указанному в первом параметре функции, а Query – параметры запроса к URL-адресу. RelativePath является текстом, а Query – записью.

В нашем примере, если базовый URL-адрес https://data.gov.uk/api, мы можем использовать параметры следующим образом:

Запрос 2

```
Web.Contents(
  "https://data.gov.uk/api",
  [
    RelativePath="3/action/package_search",
    Query=[q="cows"]
  ]
)
```

RelativePath – это просто строка "3/action/package_search", которая добавляется к базовому URL-адресу. В Query представлен единственный параметр запроса с именем "q" и значением "cows". Поэтому Query – запись с одним полем: [q="cows"]. Если вы хотите указать несколько параметров запроса, добавьте больше полей в запись Query. Например:

Запрос 3

```
Web.Contents(
  "https://data.gov.uk/api",
  [
    RelativePath="3/action/package_search",
```

[1] Номер соответствует запросу в приложенном Excel-файле. – *Прим. Багузина*

```
    Query=
    [
      q="cows",
      rows="20"
    ]
  ]
)
```

Запрос 3 создает вызов, который возвращает 20 результатов, а не 10 (по умолчанию):

https://data.gov.uk/api/3/action/package_search?q=cows&rows=20

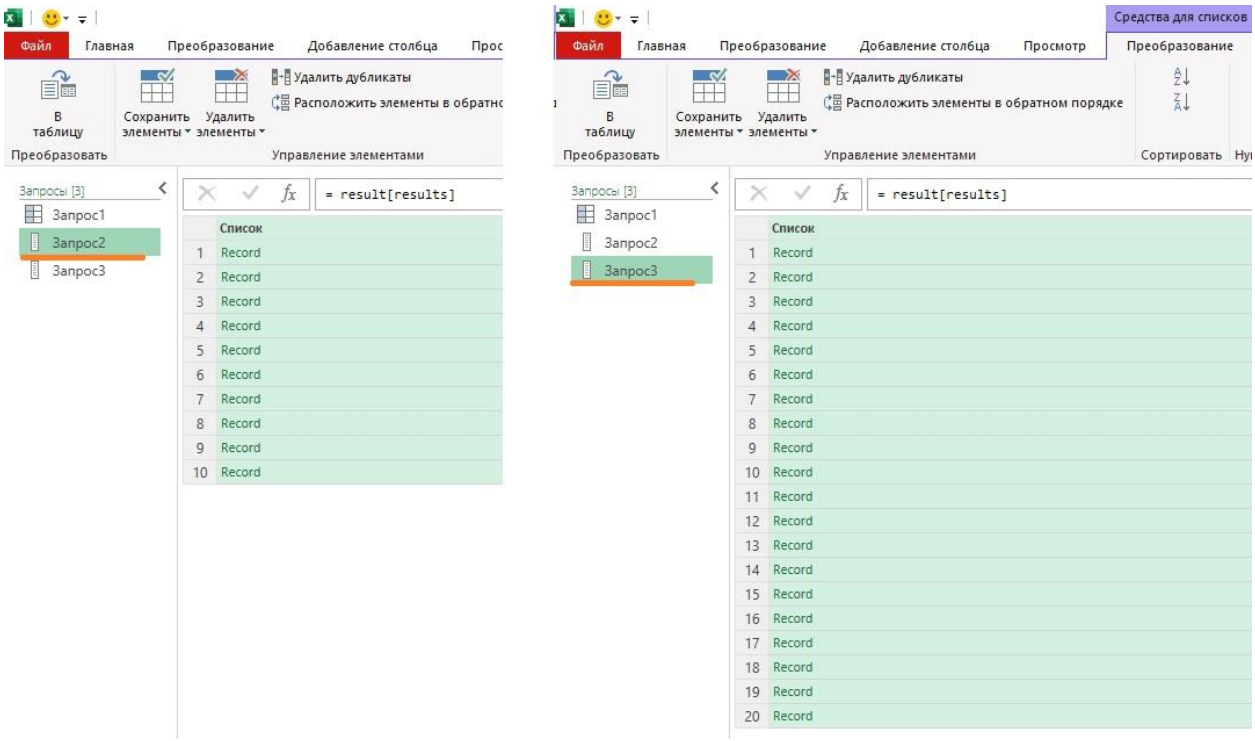

Рис. 2. Результаты Запроса 2 и Запроса 3

Очевидно, что параметры упрощают создание URL-адресов, а код делают намного понятнее.

Ошибки обновления набора данных

Иногда написанный вами код M отлично работает Power Query, но после его публикации в PowerBI.com вызывает ошибку при обновлении. Давайте рассмотрим пример с этой проблемой. Следующий запрос M использует функцию для API-вызова поиска метаданных с открытого сайта правительства Великобритании, а затем возвращает результат в таблицу:

Запрос 4

```
let
  Terms =
    #table(
      {"Term"},
      {{"apples"}, {"oranges"}, {"pears"}}
    ),
  SearchSuccessful = (Term) =>
  let
    Source =
      Json.Document(
        Web.Contents(
          "https://data.gov.uk/api/3/action/package_search?q="&Term
        )
      ),
    Success = Source[success]
```

```
    in
        Success,
        Output = Table.AddColumn(
            Terms,
            "Search Successful",
            each SearchSuccessful([Term])
        )
in
    Output
```

Вот результат:

Рис. 3. Результат выполнения Запроса 4

Код Запроса 4 делает следующее:

- Определяет таблицу с помощью #table() с тремя строками, содержащими три условия поиска.
- Определяет функцию, вызывающую API метаданных. Она принимает один параметр – условие поиска, и возвращает значение, указывающее, был ли поиск успешным или нет. То, что возвращает API, здесь не имеет значения. Важен сам факт, что API был вызван. Обратите внимание на URL-адрес, передаваемый в Web.Contents: он объединяет базовый URL-адрес со строкой, переданной через параметр Term пользовательской функции.
- Добавляет пользовательский столбец с именем *Search Successful* в таблицу, возвращаемую на первом шаге, вызывая функцию Search Successful, определенную на втором шаге, с помощью условия поиска, указанного в каждой строке первого столбца таблицы.

Этот запрос обновляется без проблем. Однако при публикации отчета, который использует этот код для PowerBI.com, и попытке обновить набор данных, вы увидите, что обновление завершается сбоем и возвращает довольно бесполезное сообщение об ошибке:

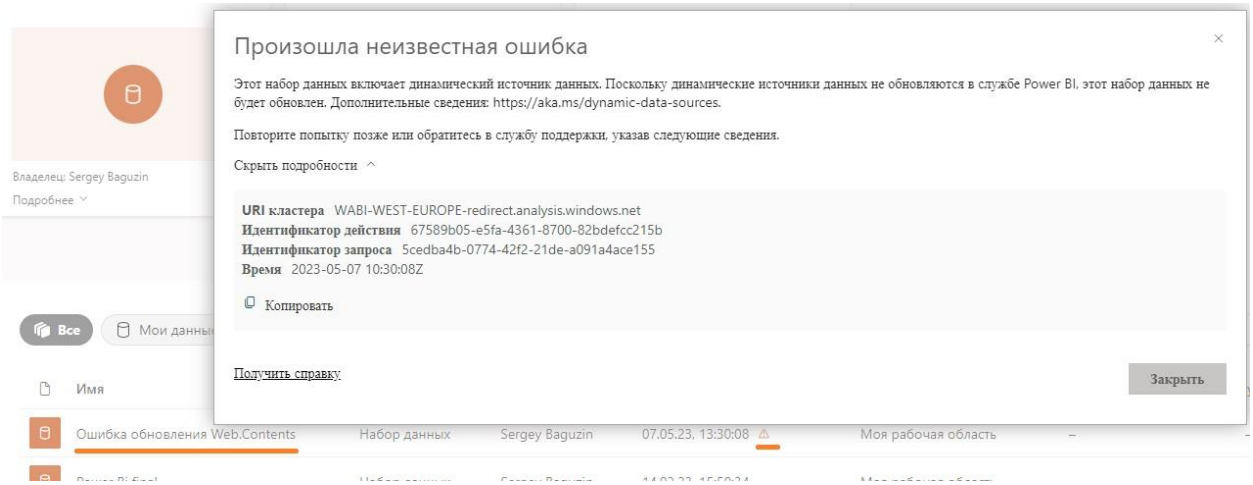

Рис. 4. Сообщение об ошибке

Проблема заключается в том, что при обновлении опубликованного набора данных Power BI выполняет статический анализ кода, чтобы определить, каковы источники данных для набора данных и верны ли предоставленные учетные данные. К сожалению, в некоторых случаях, например, когда определение источника данных зависит от параметров пользовательской функции M, этот статический анализ завершается сбоем, и поэтому набор данных не обновляется.

Хорошая новость заключается в том, что, когда, как в этом случае, источником данных является вызов функции Web.Contents(), Power BI проверяет только базовый URL-адрес, переданный в первом параметре. Поэтому, используя параметры RelativePath и Query, вы можете оставить значение, переданное первому параметру, в виде статической строки. Следующая версия запроса успешно обновляется в Power BI:

Запрос 5

```
let
  Terms =
    #table(
      {"Term"},
      {{"apples"}, {"oranges"}, {"pears"}}
    ),
  SearchSuccessful = (Term) =>
  let
    Source =
      Json.Document(
        Web.Contents(
          "https://data.gov.uk/api/3/action/package_search",
          [
            Query=[q=Term]
          ]
        )
      ),
    Success = Source[success]
  in
    Success,
    Output = Table.AddColumn(
      Terms,
      "Search Successful",
      each SearchSuccessful([Term])
    )
in
  Output
```

Метод будет работать только в том случае, если URL-адрес, переданный в первом параметре Web.Contents(), действителен сам по себе, доступен и не возвращает ошибку. Но что, если это не так? К счастью, есть еще один трюк, который вы можете использовать: когда вы указываете параметр запроса, он может переопределить части URL-адреса, предоставленного в первом параметре. Например, возьмем следующее выражение:

```
Web.Contents(
  "https://data.gov.uk/api/3/action/package_search?q=apples",
  [Query=[q="oranges"]]
)
```

При проведении статического анализа перед обновлением набора данных оценивается url

https://data.gov.uk/api/3/action/package_search?q=apples

Однако при фактическом обновлении набора данных условие поиска в параметре Query переопределяет условие поиска в базовом URL-адресе, так что вызов веб-службы, который фактически выполняется и данные которого используются запросом, будет следующим:

https://data.gov.uk/api/3/action/package_search?q=oranges

Это означает, что вы можете указать некий базовый URL-адрес, чтобы статический анализ был успешным, а затем использовать параметр Query для создания URL-адреса, который вы действительно хотите использовать.

Конечно, все это немного хак, и я уверен, что в конце концов мы дойдем до точки, когда любой код M, который работает в Power BI Desktop и/или Power Query, будет работать в опубликованном отчете. Однако не похоже, что это произойдет в ближайшем будущем, поэтому полезно знать, как обойти эту проблему.

Примечание. Параметр *Skip Test Connection* в источниках данных Power BI, добавленный в апреле 2019 г., решает некоторые проблемы, с которыми вы сталкиваетесь, когда не можете использовать RelativePath или Query для создания URL-адреса. Смотрите здесь.

> Параметр *Skip Test Connection* (Пропустить проверку соединения) используется в Power BI для источников данных, чтобы пропустить тестовое соединение с источником данных при обновлении отчета.
>
> При создании подключения к источнику данных в Power BI, он обычно выполняет тестовое соединение для проверки доступности и правильности настроек подключения. Это позволяет обнаружить возможные проблемы соединения заранее и предупредить пользователя.
>
> Однако, в некоторых случаях, особенно при настройке сложных или временных источников данных, тестовое соединение может быть не желательным или невозможным. Например, если источник данных доступен только в определенное время или требует ввода временного кода доступа.
>
> В таких случаях вы можете использовать параметр *Skip Test Connection*, чтобы пропустить тестовое соединение с источником данных при обновлении отчета. Это позволит вам настроить подключение без выполнения тестового соединения и сохранить настройки без проверки доступности источника данных.
>
> Параметр *Skip Test Connection* обычно представлен флажком или флажками в настройках подключения к источнику данных в Power BI. Вы можете отметить или выбрать этот параметр, чтобы указать, что тестовое соединение должно быть пропущено при обновлении отчета.
>
> Важно отметить, что использование параметра *Skip Test Connection* может привести к отсутствию предупреждений или ошибок при обновлении отчета, даже если фактическое соединение с источником данных не установлено или настроено неправильно. Поэтому внимательно проверяйте и настраивайте параметры подключения, прежде чем пропустить тестовое соединение.

Обработка нескольких параметров запроса URL-адреса с одинаковым именем

В примерах я буду использовать бесплатный, поддельный веб-сервис, который не требует аутентификации, поэтому вы сможете запустить код, представленный ниже. Начнем с рассмотрения вызова ресурса комментариев этого API-сервиса:

Запрос 6
http://jsonplaceholder.typicode.com/comments?postId=1

В M вы можете использовать функцию Web.Contents и параметр Query для вызова API следующим образом:

Запрос 7
```
//Создает URL-адрес http://jsonplaceholder.typicode.com/comments?postId=1
Web.Contents("http://jsonplaceholder.typicode.com/comments",[Query=[postId="1"]])
```

Этот API позволяет передавать несколько параметров URL-запроса с одним и тем же именем. Например, вызов следующего адреса является допустимым:

Запрос 8
http://jsonplaceholder.typicode.com/comments?postId=1&postId=2

А вот код M:

Запрос 9
```
Web.Contents("http://jsonplaceholder.typicode.com/comments",[Query=[postId="1",postId="2"]])
```

... вернет сообщение об ошибке, так как параметр Query принимает запись, а запись не может иметь два поля с одинаковым именем.

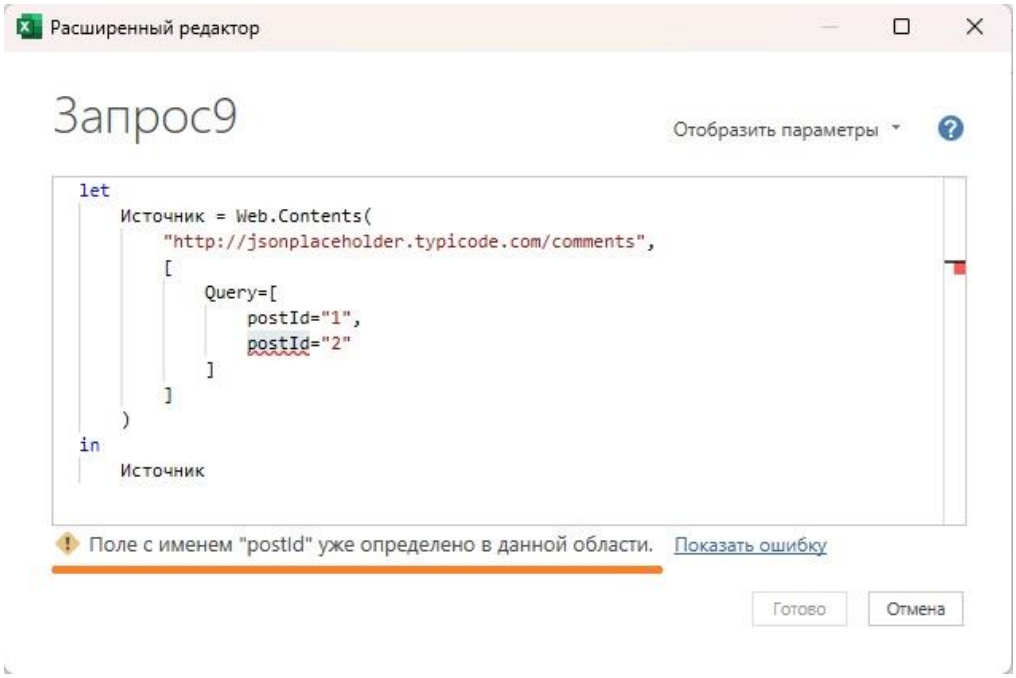

Рис. 5. Ошибка: два поля с одинаковым именем (чтобы сохранить запрос в файле Excel, параметр Query пришлось взять внутрь комментария)

В примере выше поле *postId* в записи Query содержит текстовое значение. Однако вместо этого поле может содержать список текстовых значений, и вот как можно обойти ограничение:

Запрос 10

```
//Создает URL-адрес http://jsonplaceholder.typicode.com/comments?postId=1&postId=2
Web.Contents("http://jsonplaceholder.typicode.com/comments",[Query=[postId={"1","2"}]])
```

Сработало! Мы получили URL-адрес с двумя параметрами запроса, которые оба имеют имя *postId*, и значения 1 и 2 соответственно.

Иногда, когда вы создаете URL-адрес, вы можете не захотеть добавлять к нему параметр запроса, если значение равно null. Один из способов решения этой проблемы — начать с пустой записи, а затем добавить в нее поля с помощью функции Record.AddField. Однако использование пустого списка обеспечивает другой подход:

Запрос 11

```
// Создает URL-адрес с postId = null http://jsonplaceholder.typicode.com/comments
Web.Contents("http://jsonplaceholder.typicode.com/comments",[Query=[postId={}]])
```

Это означает, что вы можете написать функцию с необязательным параметром для postId:

Запрос 12

```
(optional myPostId as text) =>
Json.Document(
  Web.Contents("http://jsonplaceholder.typicode.com/comments",[Query=[postId=myPostId ?? {}]])
)
```

Если текстовое значение передается в myPostId, то к URL-адресу добавляется параметр запроса postId; если значение не передано, myPostId равен null, а оператор объединения с null (??) вернет пустой список {} (подробнее см. Ben Gribaudo. Язык M Power Query. Структура управления).

> В нотации языка M Power Query два знака вопроса (??) обозначают оператор "опционального доступа" (optional access). Оператор "опционального доступа" позволяет обратиться к свойству или элементу структуры данных только в том случае, если оно существует. Если свойство или элемент отсутствует, оператор "опционального доступа" вернет значение по умолчанию или null, вместо генерации ошибки.

Двойные знаки вопроса применяются в следующем синтаксисе:

optionalValue??defaultValue

где optionalValue – значение, к которому вы хотите обратиться опционально, defaultValue – значение, которое будет возвращено, если optionalValue отсутствует или равно null.

Пример использования оператора "опционального доступа":

```
let
   data = [Name = "John", Age = 25],
   name = data[Name]?? "Unknown",
   address = data[Address]?? "Not specified"
in
   [Name = name, Address = address]
```

В этом примере, data – это структура данных с полями Name и Age, но без поля Address. Операторы ?? используются для обращения к Name и Address. Поскольку Name существует, переменная name будет содержать значение "John". Однако, поскольку Address отсутствует, переменная address будет содержать значение "Not specified".

Таким образом, использование двойных знаков вопроса позволяет безопасно обращаться к свойствам или элементам данных, учитывая возможное отсутствие или null-значение, и задавать значения по умолчанию в таких случаях.

Примечание. Если вам нужно увидеть вызовы веб-служб, созданные Web.Contents, при тестировании в редакторе Power Query, вы можете использовать функцию диагностики запросов, как я показываю здесь.