

# Содержание

**Где интерфейс настройки триггеров? ..... 3**  
**Отфильтровать ..... 3**  
**Отреагировать ..... 5**



# Принцип работы триггеров

Триггеры пишутся на языке C#.



## Где интерфейс настройки триггеров?

Триггеры – это отдельный подключаемый модуль. Чтобы получить к нему доступ, обратитесь к вашему менеджеру или в техническую поддержку.

## Отфильтровать

Половина работы триггера: это понять, на какие события он должен реагировать.

Фильтрация событий проходит в несколько этапов с помощью 3 инструментов, которые дополняют друг друга.

1. Простой фильтр на события, который есть прямо в интерфейсе модуля триггеров.
2. LINQ-запрос(ы). Может как использоваться, так и не использоваться. Удобно, чтобы выбрать из большой базы данных какие-то специфичные объекты.
3. В рамках написания прописываются дополнительные условия – как раз здесь используется настроенный LINQ-запрос для последней тонкой фильтрации событий.

### Как это выглядит в интерфейсе модуля триггеров:

Фильтрация на уровне интерфейса модуля триггеров:

Версия № 2 3F33C7

## 02.Создание следующего периодического совещания

#Периодическое совещание

0 УСПЕШНО	0 ОШИБКА	0 В ОЧЕРЕДИ	0.000С СРЕДНЕЕ ВРЕМЯ
--------------	-------------	----------------	-------------------------

ДРУГИЕ ПУБЛИКАЦИИ

Название  
02.Создание следующего периодического совещания

Описание

Теги  
Периодическое совещание ✕

Тайм-аут, сек

Порядок обработки событий  
Через очередь данного триггера

Выполнять повторно при сбое

Количество попыток  
3

Тайм-аут между попытками, сек

События  
Изменение Статуса ✕ Варианты

Объекты  
Совещание (объект) ✕ Варианты

Тонкая фильтрация в условии триггера:

## Условие. 02.Создание следующего периодического совещания

```
1  var linqGetMeeting = "Trigger02.GetMeetingData";
2
3  TriggerConsole.WriteLine($"ID события - {Context.EventId}, отправлено - {Context.EventSentTime}, вызвал пользователь - {Context.PersonId}");
4  if (Event.NewStatus.ToString() != "Completed"){
5      TriggerConsole.WriteLine($"Новый статус != Закрето");
6      return false;
7  }
8
9  // Получение Сделки и ее параметров
10 var jMeeting = await Api.Queries.Get(x => x
11     .DataSourceKey(linqGetMeeting)
12     .Parameters(p =>
13         {
14             p["ObjectId"] = $"{Event.ProjectId}";
15         })
16     .PageStartIndex(0)
17     .PageSize(1)
18     );
19 // Конвертируем в переменные результат запроса
20 TriggerConsole.WriteLine($"Результат запроса - {jMeeting}");
21 var jsonjMeeting = JsonConvert.DeserializeObject<List<Newtonsoft.Json.Linq.JObject>>(jMeeting);
22 TriggerConsole.WriteLine($"Число найденных Совещаний с повторением - '{jsonjMeeting.Count()}'");
23 if (jsonjMeeting.Count() == 0){
24     TriggerConsole.WriteLine($"Не найдено совещания с периодом");
25     return false;
26 }
27
28 // Сохраним переменные
29 Context["jMeeting"] = jMeeting;
30
31 return true;
32
```

## Отреагировать

События, которые попали под условия фильтров, запускают написанный скрипт. Результатом работы скрипта могут быть небольшие вычисления, создание новых записей в справочнике, новых объектов, изменение статусов, изменение значений реквизитов, их блокировка и т.д.

Триггеры работают через [интеграционное API ADVANTA](#). ⇒ Всё, что можно сделать через API, триггер может запустить в системе.

### Как это выглядит в интерфейсе модуля триггеров:

## Скрипт. 02.Создание следующего периодического совещания

```
1 var OneTimeAWeekId = "fa6ec5d3-0c34-41d9-9ad3-9ab1bf75fa48"; // Раз в неделю
2 var recvPeriodicId = "bc05b7f3-bf35-49e4-8eb9-843e86b5ed1f"; // Периодичность повтора
3 var guidMeetingId = "649da292-2eb1-4ea0-a82d-cc67c855fb99"; // совещание
4 var guidGoalId = "саааааас-8920-4ab0-a553-b4516a8417a2"; // Цель/Описание
5 var guidTitleId = "ec7aa47d-b2fa-4859-95e5-05f3b970aa13"; // Повестка
6
7 // Получение Сделки и ее параметров
8 var jMeeting = Context.GetObject<String>("jMeeting");
9
10 var jsonjMeeting = JsonConvert.DeserializeObject<List<Newtonsoft.Json.Linq.JObject>>(jMeeting);
11 TriggerConsole.WriteLine($"Число найденных Совещаний с повторением - '{jsonjMeeting.Count()}'## jMeeting - '{jMeeting}'");
12 if (jsonjMeeting.Count() == 0){
13     TriggerConsole.WriteLine($"Не найдено совещания с периодом");
14     return;
15 }
16
17 // Разберем на отдельные переменные
18 var parentId = jsonjMeeting.First()["ParentId"].ToString();
19 var startDate = jsonjMeeting.First()["StartDate"].ToObject<DateTime?>();
20 var endDate = jsonjMeeting.First()["EndDate"].ToObject<DateTime?>();
21 var responceId = jsonjMeeting.First()["ResponsibleId"].ToString();
22 var bossId = jsonjMeeting.First()["BossId"].ToString();
23 var goalNote = jsonjMeeting.First()["GoalNote"].ToString();
24 var agenda = jsonjMeeting.First()["Agenda"].ToString();
25 var timeIntervalId = jsonjMeeting.First()["TimeIntervalId"].ToString();
26 var memberIds = jsonjMeeting.First()["pIds"].ToObject<List<Newtonsoft.Json.Linq.JObject>>();
27
28 var dStart = DateTime.Now;
29 var dEnd = DateTime.Now;
30
31 // Пока есть обработка только на раз в неделю, если она, то берем.
32 if (timeIntervalId == OneTimeAWeekId) {
33     dStart = startDate.Value.AddDays(7); // +7 дней
34     dEnd = endDate.Value.AddDays(7); // +7 дней
35     TriggerConsole.WriteLine($"Начало нового совещания - {dStart}, окончание - {dEnd}");
36 }
37 else {
38     TriggerConsole.WriteLine($"Временной интервал не раз в неделю - {timeIntervalId}");
39     return;
40 }
41
```

From:  
<https://wiki.a2nta.ru/> - Wiki [3.x]

Permanent link:  
[https://wiki.a2nta.ru/doku.php/product/triggers/how\\_it\\_works?rev=1660909444](https://wiki.a2nta.ru/doku.php/product/triggers/how_it_works?rev=1660909444)

Last update: **19.08.2022 11:44**

