



ОНЛАЙН-ОБРАЗОВАНИЕ


Онлайн-образование

Проверить, идет ли запись!





Меня хорошо видно && слышно?

Ставьте  , если все хорошо
Напишите в чат, если есть проблемы

Работа с методами как с переменными (delegates, events)



Виктор Дзицкий

TeamLead, Full Stack .Net Developer

SolarLab

Telegram: @Dzitskiy

Правила вебинара



Активно участвуем



Задаем вопрос в чат или голосом



Off-topic обсуждаем в Slack #канал группы или #general



Вопросы вижу в чате, могу ответить не сразу

Маршрут вебинара (примерно 1 час 30 мин)

1. Вводная
2. Тестирование
3. Делегаты (delegates)
4. События (events)
5. Итоги тестирования
6. Практика
7. Вопросы и ответы

Цели вебинара

- Узнать что такое делегаты и события и для чего они нужны
- Узнать во что компилируются делегаты
- Узнать про возможные утечки памяти при работе с событиями
- Написать создание и обработку событий в приложении приема документов для заявлений
- Поработать с `System.IO.FileSystemWatcher`
- Поработать с `System.Timers.Timer`

The background of the slide is a high-angle, blue-tinted aerial photograph of a dense urban skyline, likely New York City. Overlaid on this image is a semi-transparent blue band that contains a white network diagram. The diagram consists of numerous small dots connected by thin white lines, forming a complex web that suggests digital connectivity or a global network. The text "Что такое делегат?" is centered within this blue band in a large, white, sans-serif font.

Что такое делегат?

Что такое делегат?

Делегат это тип, который определяет сигнатуру используемой функции

Похож на указатель на функцию из C++

The background of the slide is a composite image. The top and bottom portions show an aerial view of a dense city skyline, likely New York City, with numerous skyscrapers. A semi-transparent blue band runs horizontally across the middle of the image. Overlaid on this band is a white network diagram consisting of numerous small dots connected by thin lines, creating a web-like pattern. The text "Для чего нужны делегаты?" is centered within this blue band in a large, white, sans-serif font.

Для чего нужны делегаты?

Объявление делегата

Реальный делегат из пространства System.Threading:

```
public delegate void TimerCallback(object? state);
```

Название делегата

Тело отсутствует

Обязательное ключевое слово

Делегат не привязан к классу

Как и любой другой тип его можно объявить внутри namespace:

```
namespace X
{
    ▪class
    ▪struct
    ▪interface
    ➤delegate
    ▪enum
}
```

Создание экземпляра делегата

```
public void TimerCallbackHandler(object? state)
{
    Console.WriteLine("x");
}


async Task Main(string[] args)
{
    TimerCallback timerCallback = TimerCallbackHandler;
}
```


Передача делегата в качестве аргумента

```
...public sealed class Timer : MarshalByRefObject, IAsyncDisposable, IDisposable
{
    ...public Timer(TimerCallback callback);
    ...public Timer(TimerCallback callback, object? state, int dueTime, int period);
    ...public Timer(TimerCallback callback, object? state, long dueTime, long period);
    ...public Timer(TimerCallback callback, object? state, TimeSpan dueTime, TimeSpan period);
    ...public Timer(TimerCallback callback, object? state, uint dueTime, uint period);
}
```

Передача делегата в качестве аргумента

```
...public static class Dns
{
    ...public static IAsyncResult BeginGetHostAddresses(string hostNameOrAddress, AsyncCallback requestCallback, object state);
    ...public static IAsyncResult BeginGetHostByName(string hostName, AsyncCallback requestCallback, object stateObject);
    ...public static IAsyncResult BeginGetHostEntry(IPAddress address, AsyncCallback requestCallback, object stateObject);
    ...public static IAsyncResult BeginGetHostEntry(string hostNameOrAddress, AsyncCallback requestCallback, object stateObject);
    ...public static IAsyncResult BeginResolve(string hostName, AsyncCallback requestCallback, object stateObject);
}
```



Делегат

Делегаты и методы

Могут принимать любые методы с подходящей сигнатурой (static)

```
public static void TimerCallbackHandler(object? state)
{
    ...
}

public static void AnotherCallbackHandler(object? state)
{
    ...
}

static async Task Main(string[] args)
{
    TimerCallback timerCallback1 = TimerCallbackHandler;
    TimerCallback timerCallback2 = AnotherCallbackHandler;
    ...
}
```

Делегаты и методы

Могут принимать любые методы с подходящей сигнатурой (instance)

```
public void TimerCallbackHandler(object? state)
{
    Console.WriteLine("x");
}

public void AnotherCallbackHandler(object? state)
{
    Console.WriteLine("y");
}

async Task Main(string[] args)
{
    TimerCallback timerCallback1 = TimerCallbackHandler;
    TimerCallback timerCallback2 = AnotherCallbackHandler;
}
```


Во что компилируется делегат?

Все делегаты, которые мы объявляем наследуют System.MulticastDelegate -> System.Delegate.

```
public class TimerCallback : System.MulticastDelegate
{
    public MyDelegate(object? state, IntPtr method);
    public virtual void Invoke(object? state);
    public virtual IAsyncResult BeginInvoke(object? state, AsyncCallback callback, object @object);
    public virtual void EndInvoke(IAsyncResult result);
}
```

Повторяет сигнатуру

Вызов делегата

Вызов делегата как метода и через Invoke – одно и тоже

```
static async Task Main(string[] args)
{
    TimerCallback timerCallback = TimerCallbackHandler;

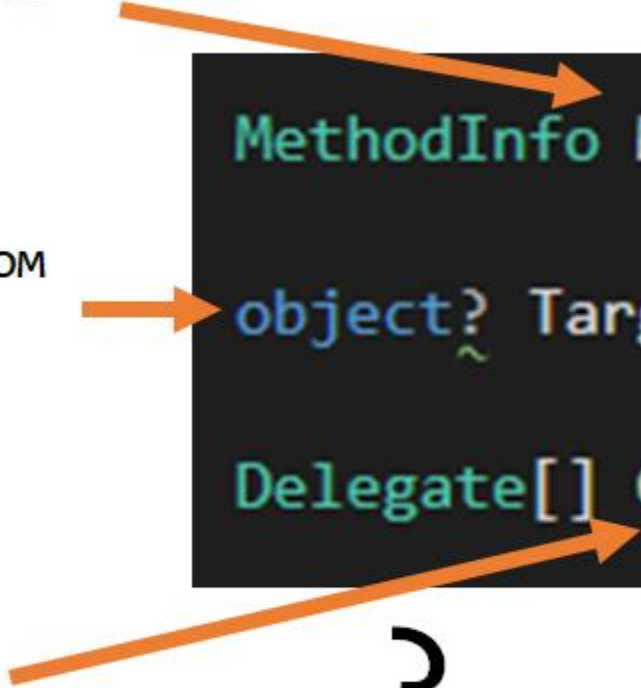
    timerCallback(new object());
    timerCallback.Invoke(new object());
}
```


System.Delegate

Вспоминаем Reflection, это метаданные метода

Ссылка на объект в котором вызывается делегат (null – если static)

Список делегатов, которые нужно вызвать



```
MethodInfo Method { get; }
```

```
object? Target { get; }
```

```
Delegate[] GetInvokationList();
```

?

InvokationList

- Любой делегат, может содержать в себе ссылки на другие делегаты такого же типа
- Все эти ссылки содержаться в InvokationList
- При вызове делегата они вызываются в порядке добавления

Арифметика делегатов

```
async Task Main(string[] args)
{
    TimerCallback timerCallback1 = TimerCallbackHandler;
    TimerCallback timerCallback2 = AnotherCallbackHandler;

    TimerCallback timerCallback3 = timerCallback1 + timerCallback2;

    timerCallback3 = timerCallback3 - timerCallback2;
}
```

Что будет при вызове timerCallback3?

Арифметика делегатов

Операторы «+» и «−» у делегатов:

- Это сложение и вычитание множеств
- В качестве множеств выступают `InvocationList`'ы делегатов

Арифметика делегатов

Чаще всего используют операторы += и -= с одним делегатом в правой части

```
static async Task Main(string[] args)
{
    TimerCallback timerCallback = TimerCallbackHandler;
    timerCallback += AnotherCallbackHandler;
    timerCallback -= TimerCallbackHandler;
}
```

Что они делают?

Готовые generic делегаты в .NET

```
public delegate void Action();  
public delegate void Action<in T>(T obj);  
public delegate void Action<in T1, in T2>(T1 arg1, T2 arg2);  
...  
public delegate TResult Func<out TResult>();  
public delegate TResult Func<in T1, out TResult>(T obj);
```


Готовые generic делегаты в .NET

```
public delegate void Action();  
public delegate void Action<in T>(T obj);  
public delegate void Action<in T1, in T2>(T1 arg1, T2 arg2);  
...  
public delegate TResult Func<out TResult>();  
public delegate TResult Func<in T1, out TResult>(T obj);  
...  
delegate bool Predicate<in T>(T obj);
```

Нужно здесь останавливаться подробнее?

Выводы

- Делегат компилируется в специальный класс-наследник `System.MulticastDelegate`
- `+=` и `-=` это добавление/удаление делегата в список вызовов

The background of the slide is a high-angle, blue-tinted aerial photograph of a dense urban skyline, likely New York City. Overlaid on this image is a semi-transparent blue band that contains a white network diagram. The diagram consists of numerous small dots connected by thin white lines, creating a web-like structure that spans the width of the slide. In the center of this band, the Russian text "Что такое события?" is written in a large, white, sans-serif font.

Что такое события?

События

Событие (event) — это специальный член класса/интерфейса/структуры (как, например property) , с помощью которого они предоставляют уведомления.

```
...public delegate void ElapsedEventHandler(object sender, ElapsedEventArgs e);
```

Делегат

```
...public event ElapsedEventHandler Elapsed;
```

Ключевое слово event

Объявление события

Простая форма:

```
public event Action SomeActionHappened;
```

Развернутая форма:

Подписка +=

Отписка -=

```
private Action _someActionHappened;  
public event Action SomeActionHappened  
{  
    add  
    {  
        _someActionHappened += value;  
    }  
    remove  
    {  
        _someActionHappened -= value;  
    }  
}
```


Подписка и отписка на событие

```
private static void OnTimerElapsed(object sender, System.Timers.ElapsedEventArgs e)
{
    Console.WriteLine("tick");
}

static async Task Main(string[] args)
{
    System.Timers.Timer timer = new System.Timers.Timer();
    // Подписались
    timer.Elapsed += OnTimerElapsed; // Проваливаемся в секцию add

    // ...

    // Отписались
    timer.Elapsed -= OnTimerElapsed; // Проваливаемся в секцию remove
}
```

Вопрос на собеседовании: в чем отличие события от делегата?

- Событие – это член класса/интерфейса/структуры, а делегат это отдельный тип
- Событию нельзя присвоить другой делегат, доступны только операторы += и -= (подписка и отписка) через add и remove
- Событие нельзя вызвать вне его владельца (класса или структуры)
- Делегат можно передавать по ссылке в метод, а событие - нет
- Во время выполнения программы никакой разницы между простыми событиями и ссылками на делегат нет

Событие в System.Timers.Timer

Событие срабатывания таймера

```
...public class Timer : Component, ISupportInitialize
{
    ...public Timer();
    ...public Timer(double interval);

    ...public bool AutoReset { get; set; }
    ...public bool Enabled { get; set; }
    ...public double Interval { get; set; }
    ...public override ISite Site { get; set; }
    ...public ISynchronizeInvoke SynchronizingObject { get; set; }

    ...public event ElapsedEventHandler Elapsed;

    ...public void BeginInit();
    ...public void Close();
    ...public void EndInit();
    ...public void Start();
    ...public void Stop();
    ...protected override void Dispose(bool disposing);
}
```


События в System.IO.FileSystemWatcher

События изменения в файловой системе

```
...public class FileSystemWatcher : Component, ISupportInitialize
{
    ...public FileSystemWatcher();
    ...public FileSystemWatcher(string path);
    ...public FileSystemWatcher(string path, string filter);

    ...public ISynchronizeInvoke SynchronizingObject { get; set; }
    ...public override ISite Site { get; set; }
    ...public string Path { get; set; }
    ...public NotifyFilters NotifyFilter { get; set; }
    ...public int InternalBufferSize { get; set; }
    ...public bool IncludeSubdirectories { get; set; }
    ...public Collection<string> Filters { get; }
    ...public string Filter { get; set; }
    ...public bool EnableRaisingEvents { get; set; }

    ...public event ErrorEventHandler Error;
    ...public event FileSystemEventHandler Changed;
    ...public event FileSystemEventHandler Created;
    ...public event FileSystemEventHandler Deleted;
    ...public event RenamedEventHandler Renamed;

    ...public void BeginInit();
    ...public void EndInit();
    ...public WaitForChangedResult WaitForChanged(WatcherChangeTypes changeType, int timeout);
    ...public WaitForChangedResult WaitForChanged(WatcherChangeTypes changeType);
    ...protected override void Dispose(bool disposing);
    ...protected void OnChanged(FileSystemEventArgs e);
    ...protected void OnCreated(FileSystemEventArgs e);
    ...protected void OnDeleted(FileSystemEventArgs e);
    ...protected void OnError(ErrorEventArgs e);
    ...protected void OnRenamed(RenamedEventArgs e);
}
```

Принятый стандарт делегатов событий в .NET

- Делегат события обычно заканчивается на слово EventHandler (исключение Blazor)
- Делегат события не возвращает никаких значений
- Делегат события обычно имеет 2 аргумента object и наследник EventArgs:

```
...public delegate void FileSystemEventHandler(object sender, FileSystemEventArgs e);
```

Практика применения делегатов и событий

- Делегат чаще используют в качестве параметра метода или конструктора и когда нужно иметь возвращаемое значение
- Событие используют для того, чтобы дать возможность подписки на уведомления конкретного объекта

```
...public delegate void FileSystemEventHandler(object sender, FileSystemEventArgs e);
```


Не забывайте отписываться от событий

- Источник событий хранит ссылку на подписчика
- **Каким образом?**
- Если источник событий живет дольше, чем подписчик, то отсутствие отписки может привести к утечки памяти!

```
timer.Elapsed -= OnTimerElapsed;
```

Выводы

- Событие это своего рода обертка над делегатом
- Событие имеет ряд ограничений в ходе компиляции, но *простое* событие в ходе выполнения программы не отличается от поля типа делегата

Полезные ссылки

Арифметика делегатов

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/delegates/how-to-combine-delegates-multicast-delegates>

Сайт, позволяющий быстро увидеть декомпилированный код и т.п.

<https://sharplab.io/>

Замыкания на переменных цикла в C# 5

<https://habr.com/ru/post/141270/>

Локальные функции в C#

<https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/local-functions>

Слабые события


<https://habr.com/ru/post/89529/>

5 Techniques to avoid Memory Leaks by Events in C# .NET you should know

<https://michaelscodingspot.com/5-techniques-to-avoid-memory-leaks-by-events-in-c-net-you-should-know/>

The background of the slide is a high-angle, blue-tinted aerial photograph of a dense urban skyline, likely New York City. Overlaid on this image is a semi-transparent blue band that contains a white network diagram. The diagram consists of numerous small dots connected by thin white lines, creating a web-like structure that spans the width of the slide. In the center of this band, the word "Вопросы" is written in a large, white, sans-serif font.

Вопросы

The background of the image is an aerial photograph of a city skyline, likely New York City, with numerous skyscrapers. The image is overlaid with a semi-transparent blue layer that features a white network pattern of interconnected dots and lines. The text is centered within this blue layer.

Заполните, пожалуйста,
опрос о занятии по ссылке в чате

The background of the entire slide is an aerial photograph of a dense city skyline, likely New York City, with numerous skyscrapers. A semi-transparent blue overlay covers the entire image. Overlaid on this blue background is a network of thin, light blue lines connecting various points, creating a geometric, web-like pattern. The text is centered in the middle of the slide.

Спасибо за внимание!
Приходите на следующие вебинары