Визуальное программирование

ЛЕКЦИЯ 7

Содержание лекции

01 События

02 Работа с событиями

03 EventHandler и EventArgs

04 Делегаты и события

СОБЫТИЯ

События

Событие представляет собой автоматическое уведомление о том, что произошло некоторое действие.

С точки зрения программы, событием называют сообщение, посланное объектом чтобы проинформировать о совершении некоторого действия.

События

Класс, содержащий описание события, называется издателем (Publisher). Объект такого класса уведомляет другие объекты, подписавшиеся на событие, о том, что это событие произошло.

Реакция на событие осуществляется с помощью так называемых обработчиков события. Обработчик события – это обычный метод, который выполняет некоторые действия в программе, в случае если состоялось (сгенерировалось) событие.

Подписчик (Subscriber) – это объект, который предоставляет обработчики событий.

События позволяют издателю (Publisher) уведомлять подписчиков (Subscribers) о возникновении каких-либо ситуаций.

События

События действуют по следующему принципу: объект, проявляющий интерес к событию, регистрирует обработчик этого события. Когда же событие происходит, вызываются все зарегистрированные обработчики этого события.

События основаны на делегатах и предоставляют им механизм публикации/подписки. Как и делегаты, события поддерживают групповую адресацию. Это дает возможность нескольким объектам реагировать на уведомление о событии.

Для того, чтобы можно было обрабатывать (запускать) списки обработчиков события, делегат не должен возвращать значение. То есть, делегат может возвращать тип void.

События

- Класс издатель определяет в какой момент будет вызвано его событие. Подписчики определяют ответное действие, которое будет выполнено по наступлению события издателя.
- На событие может подписаться несколько подписчиков. Подписчик может обрабатывать несколько событий от нескольких издателей.

События

- Если у события несколько подписчиков, то при его возникновении происходит синхронный вызов обработчиков событий.
- Для типа делегата, определяемого для события, тип возвращаемого значения должен быть void.
- События основаны на делегатах. Делегаты поддерживают многоадресную рассылку, это обеспечивает поддержку нескольких подписчиков для любого источника событий.

События

События представляют собой специальный вид многоадресного делегата, который можно вызвать только из класса или структуры, в которых он объявлен (класс Publisher). Если другие классы или структуры подписываются на событие, их методы обработчиков событий будут вызываться, когда класс Publisher будет вызывать событие.

РАБОТА С СОБЫТИЯМИ

Порядок работы

1. Объявить тип делегата в классе.
2. Объявить событие в данном классе или создать другой класс, который содержит объявления события.
3. В некотором методе создать список обработчиков, которые будут вызываться при вызове данного события. Это осуществляется с помощью операторов ‘=’ и ‘+=’. Создание списка означает регистрацию обработчиков для данного события.
4. Вызвать событие (запустить на выполнение) из этого метода.

Объявление события

События являются членами класса и объявляются с помощью ключевого слова event.

Чаще всего для этой цели используется следующая форма:

```
event делегат_события имя_события;
```

где делегат_события обозначает имя делегата, используемого для поддержки события, а имя_события — конкретный объект объявляемого события.

Объявление события

```
// Объявим тип делегата для обработчика событий
public delegate void MyEventHandler();
8 references
public class MyEvent
{
    // Объявляем событие
    public event MyEventHandler OnSomeEvent;

    // Этот метод вызывается для запуска события
    4 references
    public virtual void SomeEvent()
    {
        // Если событие не пустое
        //if (OnSomeEvent != null)
        //    // Вызывается обработчик(и) с помощью делегата
        //    OnSomeEvent();
        OnSomeEvent?.Invoke();
    }
```

Объявление события

Событие может быть объявлено как статическое событие с помощью ключевого слова static. Это делает событие доступным для вызывающих объектов в любое время, даже если экземпляр класса не существует.

Событие может быть помечено как виртуальное событие с помощью ключевого слова virtual. Это позволяет производным классам переопределять поведение события с помощью ключевого слова override.

Объявление события

Событие, переопределяющее виртуальное событие, также может быть запечатанным (sealed), что указывает, что для производных классов оно больше не является виртуальным. И наконец, можно объявить событие абстрактным (abstract), что означает, что компилятор не будет создавать блоки доступа к событиям add и remove.

Событие можно объявлять в интерфейсе.

Обработчики событий

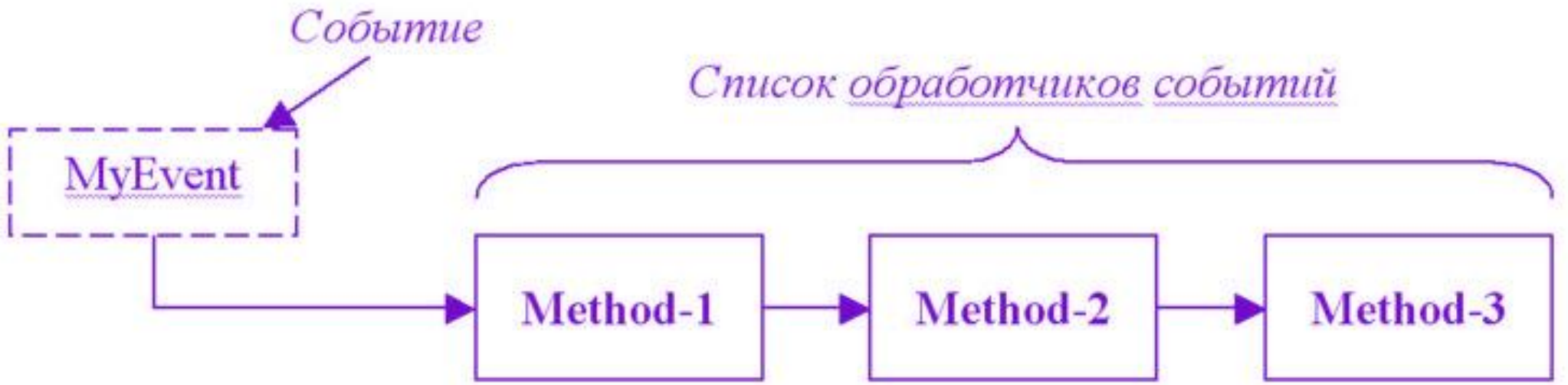
Методы экземпляра и статические методы могут быть использованы в качестве обработчиков событий, но между ними имеется отличие:

- Когда статический метод используется в качестве обработчика, уведомление о событии распространяется на весь класс.
- Когда в качестве обработчика используется метод экземпляра, события адресуются конкретным экземплярам класса. Следовательно, каждый объект определенного класса, которому требуется получить уведомление о событии, должен быть зарегистрирован отдельно.

Как привязать обработчики

```
// регистрация методов для события MyEvent
MyEvent += MyMethod1; // MyEvent => MyMethod1
MyEvent += MyMethod2; // MyEvent => MyMethod1 -> MyMethod2
MyEvent += MyMethod3; // MyEvent => MyMethod1 -> MyMethod2 -> MyMethod3
```

Обработчики событий



Обработчики событий

```
// Обработчик события
1 reference
static void FirstHandler()
{
    Console.WriteLine("Произошло событие");
}

0 references
public static void FirstDemo()
{
    // Создаем объект класса события
    MyEvent evt = new MyEvent();

    // Добавим метод Handler() в список обработчиков события
    evt.OnSomeEvent += FirstHandler;

    // Запустим событие
    // Здесь вызываются все методы, являющиеся обработчиками
    evt.SomeEvent();
}
```

EVENTHANDLER И EVENTARGS

Делегат EventHandler

Теоретически для объявления события можно использовать любой делегат. Но согласно стандартному шаблону определенного .NET Framework нужно использовать один из специально предопределенных делегатов EventHandler, предназначенных именно для событий.

```
public delegate void EventHandler(object sender, EventArgs e)
```

```
public delegate void EventHandler<T>(object sender, T e)
```

Делегат EventHandler

Стандартный делегат для событий содержит два аргумента: sender – отправитель события, e – аргументы события.

Для аргумента sender по соглашению следует использовать тип System.Object, даже если вероятно вам известен более точный производный тип.

Вторым аргументом является тип производный от System.EventArgs, в котором можно указать необходимые параметры события, которыми могут воспользоваться подписчики события при вызове обработчика на своей стороне.

Класс EventArgs

Данные, связанные с событием, могут быть предоставлены с помощью класса данных события. .NET предоставляет множество классов данных событий, которые можно использовать в приложениях.

Если требуется создать пользовательский класс данных события, создайте класс, производный от класса EventArgs, а затем укажите все члены, необходимые для передачи данных, связанных с событием.

Класс EventArgs

Данные, связанные с событием, могут быть предоставлены с помощью класса данных события. .NET предоставляет множество классов данных событий, которые можно использовать в приложениях.

Если требуется создать пользовательский класс данных события, создайте класс, производный от класса EventArgs, а затем укажите все члены, необходимые для передачи данных, связанных с событием.

System.EventArgs – предопределенный класс, имеющий только статическое свойство Empty.

Класс EventArgs

В большинстве случаев следует использовать схему именования .NET и завершать имя класса данных события ключевым словом EventArgs. Именуется класс осмысленно в соответствии с содержащейся в нем информацией (а не событием, для которого он будет использоваться).

Также рекомендуется делать свойства в типе аргумента события неизменяемыми (только для чтения). Таким образом, один подписчик не сможет изменить значения до того, как их увидит другой подписчик.

Класс EventArgs

Если тип события не требует дополнительных аргументов, необходимо предоставить оба аргумента для делегата. Существует специальное значение EventArgs.Empty, которое следует использовать для обозначения что событие не содержит никаких дополнительных сведений.

В таком случае событие должно быть представлено делегатом EventHandler параметризированный стандартным классом EventArgs.

ДЕЛЕГАТЫ И СОБЫТИЯ

Зачем использовать события

Основное назначение событий – предотвращение влияния подписчиков друг на друга.

Если в коде убрать ключевое слово event, чтобы некоторое событие стало обычным полем делегата, результаты будут теми же самыми. Однако класс станет менее надежным, потому что подписчики смогут предпринимать следующие действия:

- заменять других подписчиков, переустанавливая свойство (вместо операции +=);
- очищать всех подписчиков (устанавливая поле в null);
- выполнять групповую рассылку другим подписчикам путем вызова делегата.

Выбор между делегатами и событиями

Важным фактором является обязательность наличия подключенного подписчика.

Если код должен вызывать код, предоставленный подписчиком, следует использовать структуру на основе делегатов.

Если код может выполнить все задачи, не вызывая подписчиков, следует использовать структуру на основе событий.

Выбор между делегатами и событиями

Еще одним аспектом является прототип метода, который требуется для метода делегата.

Все делегаты, используемые для событий, имеют тип возвращаемого значения void. В случае если в методе нужно использовать возвращаемое значение, то предпочтение следует отдать обычным делегатам.

ВОПРОСЫ ПО ЛЕКЦИИ