



8. Делегати и събития. Observer Pattern.

Емил Стойчев

Програмиране с .NET Framework и WPF



Съдържание

- Делегати
 - Дефиниция
 - Вътрешно представяне
 - Анонимни делегати
 - Комбиниране
- Събития
- Observer Pattern
 - В .NET
 - Event Pattern



Делегати

- Тип (структура от данни), който държи референция към статичен метод или метод на инстанция
- Указател към функция в C/C++, но type-safe
- Позволява да ползвате **метод** като **параметър**
- Използват се за **динамично извикване на методи по време на изпълнение (runtime)**

```
public delegate int  
PerformCalculation( int x, int y );
```



Пример

```
public delegate void EchoDelegate( string text );  
  
public HandlerDemo()  
{  
    EchoDelegate echo = new EchoDelegate( this.Echo );  
    echo( ".NET Rocks!" );  
}  
  
private void Echo( string text )  
{  
    Console.WriteLine( text );  
}  
private void EchoToFile( string text )  
{  
    new StreamWriter("Console.txt").Write(text);  
}
```



Използване

- Делегатът се инстанциира като клас
- Методът се подава в конструктора
- При повече от един метод използваме `+ =`
- За да “разкачим” метод използваме `- =`

```
EchoDelegate echo = new EchoDelegate( this.ConsoleEcho );
echo += new EchoDelegate( this.LogEcho );
echo -= new EchoDelegate( this.LogEcho );
```



Вътрешно представяне (1)

- Клас, който наследява `System.MulticastDelegate` : `System.Delegate`
- `MulticastDelegate` държи свързан списък от методи (`invocation list`)
- `Target` – реферира инстанция на обект
- `Method` – реферира метод от инстанцията
- `GetInvocationList` – списък от методи викани от делегата
- `Invoke` – извиква метод(и) сочен от делегата



Вътрешно представяне (2)



- **Multicast** делегат – по подразбиране
- Два метода в **invocation list**-а
- Методите задължително трябва да имат същата сигнатура – тип и брой на параметрите и тип на резултата



Вътрешно представяне (3)

```
public delegate void EchoDelegate( string text );  
  
public sealed class EchoDelegate : System.MulticastDelegate  
{  
    public EchoDelegate(object target, int method);  
    public virtual void Invoke(string text);  
    public virtual IAsyncResult BeginInvoke(string text,  
    AsyncCallback callback, object obj);  
    public virtual void EndInvoke(IAsyncResult result);  
}
```



Демо

ДЕЛЕГАТИ



Анонимни делегати

- Делегати без име, често се използват като callback методи

```
public delegate void EchoDelegate( string text );  
  
public HandlerDemo()  
{  
    EchoDelegate echo = delegate( string text )  
    {  
        Console.WriteLine( text );  
    };  
    echo( ".NET Rocks!" );  
}
```



Комбиниране на делегати

```
public delegate void Greet( string name );
public HandlerDemo()
{
    Greet g1 = delegate( string name )
    {
        Console.WriteLine( "Hi, " + name );
    };
    Greet g2 = delegate( string name )
    {
        Console.WriteLine( "What's up " + name );
    };
    Greet g3 = g1 + g2;
    Greet g4 = g3 - g1;
    Console.WriteLine( "Invoking g1" );
    g1( "foo" );
    Console.WriteLine( "Invoking g2" );
    g2( "foo" );
    Console.WriteLine( "Invoking g3" );
    g3( "foo" );
    Console.WriteLine( "Invoking g4" );
    g4( "foo" );
}
```



Резултат

Invoking g1

Hi, foo

Invoking g2

What's up foo

Invoking g3

Hi, foo

What's up foo

Invoking g4

What's up foo



Generic делегати

- Делегат, който може да вика всеки метод, който връща *тип* и приема *точно няколко аргумента от определен тип*, дефинирани от извикващия код

```
public delegate T1 EchoDelegate<T1, T2, T3>( T2 x, T3 y );
EchoDelegate<string, string, int> echo =
    delegate( string x, int y )
{
    return x + y;
};
```



Action<T> и Func<T, TResult>

- Action<T> - generic делегат; капсулира метод, който може да приема 1-16 параметъра и не връща резултат

```
public delegate void Action<T>( T obj );
```

- Func<T, TResult> - generic делегат; капсулира метод, който може да приема 1-16 параметъра и връща стойност от тип TResult

```
public delegate TResult Func<T, TResult>( T obj );
```



Predicate<T>

- Generic делегат
- Определя набор от критерии и дали даден обект отговаря на тези критерии

```
public delegate bool Predicate<T>( T obj );
```



Резюме

- Подобни на указатели към функции в C++, но **type safe**
- Позволяват методи да се подават като параметри
- Могат да бъдат свързани няколко делегата последователно - позволява извикването на няколко метода
- Най-често се използват заедно със събития



Събития (Events)

- Позволяват клас да съобщи, че нещо е станало
- Събитията се декларират използвайки делегат – когато дадено събитие настъпи се извиква делегатът
- Важна роля при работа с GUI



Деклариране

- Делегат (EventHandler)

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

- Event – декларира се като field от тип
делегата

```
public event ClickEventHandler Click;
```



Извикване

- Използва се като field
- Винаги проверявайте за null

```
if ( this.Click != null )  
{  
    this.Click( this, EventArgs.Empty );  
}
```



Абониране

- “Закачане” +=
- “Откачане” -=
- Става от външни класове или от самия клас

```
this.Click += new ClickEventHandler( HandlerDemo_Click );  
this.Click -= new ClickEventHandler( HandlerDemo_Click );
```



Термини

- Sender – изпращац – обект, който предизвиква събитието
- Receiver – получател – обект, който получава събитието; може да има много получатели
- Event Handler – метода сочен от делегата



System.EventHandler и EventArgs

- System.EventHandler – вграден делегат, ползваме когато събитието не носи данни
- System.EventArgs – носи данните за събитието
- EventArgs.Empty – представя събитие без данни
- EventHandler<TEventArgs>



Демо

Делегати и събития



Observer Pattern (1)

- Цел – разделяне на отговорностите
 - UI и бизнес логика
 - Различни обекти в системата
- Логически модел
 - Observer (view) – грижи се за показване на данни
 - Subject (model) – бизнес абстракция
 - Когато настъпи промяна в subject обекта, observer-а я отбелязва (observes)



Observer Pattern (2)



- Пример – приложение, което следи акциите на борсата
 - Stock – информация за акция
 - StockDisplay – показва цената на акцията



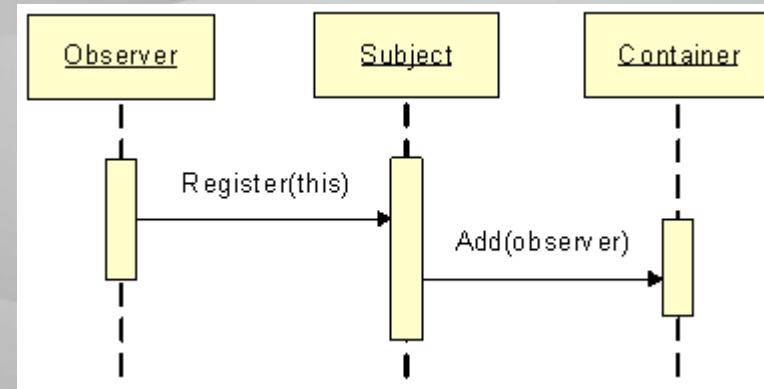
Observer Pattern (3)

- Физически модел
 - Регистрация (registration)
 - Известяване (notification)
 - От регистрацията(unregistration)



Observer Pattern (4)

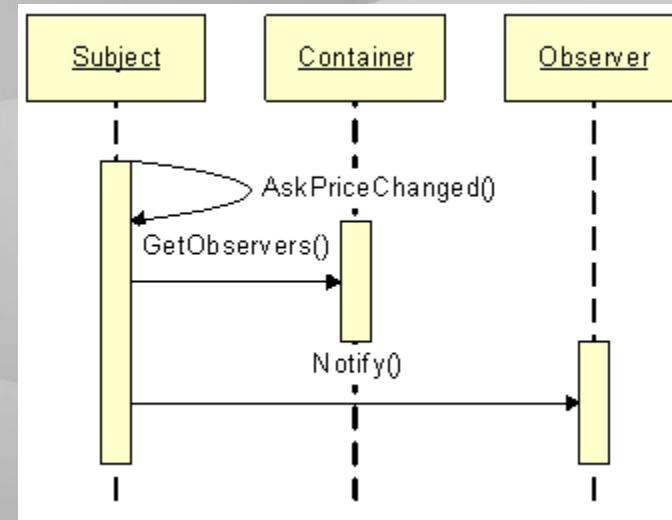
- Регистрация





Observer Pattern (5)

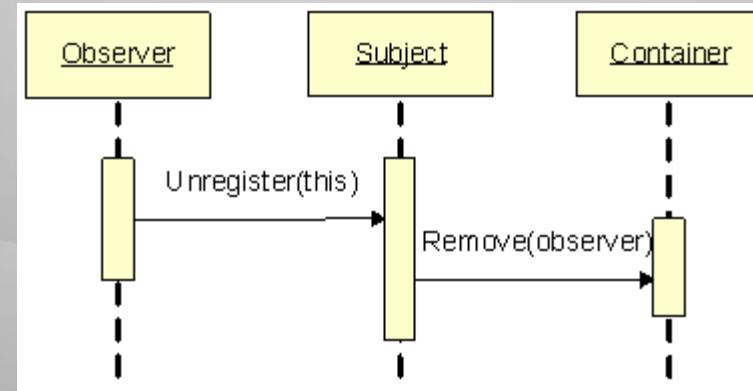
- Известяване





Observer Pattern (6)

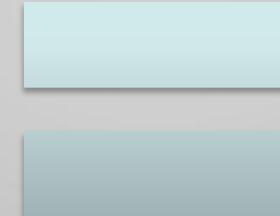
- От регистрация





Observer Pattern в .NET

Observer
Pattern



Делегати
и
събития
в .NET



Event Pattern (1)

- Naming Conventions

- Име на събитието – глагол
 - StatusChanged
- Име на делегата – име на събитието + EventHandler
 - StatusChangedEventHandler
- Сигнатура на делегата
 - Референция към изпращача – sender (System.Object)
 - Контекст – е (System.EventArgs)



Event Pattern (2)

- Контекст – име на събитието + EventArgs
 - StatusChangedEventArgs
- Метод, който хвърля събитието
 - Име - On + име на събитието – OnStatusChanged
 - Достъп - protected



Упражнение

- Реализирайте метода `accumulate` чрез използване на `delegate` и покажете как работи чрез пример по ваш избор (примерно сортиране).
- Реализирайте клас `Student` с properties за име, ФН и лист от записани курсове, който нотифицира при промяна на всяко property. Реализирайте клас `StudentDisplay`, който показва информация за един `Student`. Демонстрирайте класовете в примерно приложение.



Въпроси?

- Следете <http://fmi.silverlight.bg/>
- Решения на упражненията пускайте във форума или на email estoychev at completit.com