**20. Таймер (Timer). Работа с таймери. Пропъртита, събития. Пример.**

**Таймери**

Често в приложенията, които разработваме, възниква необходимост от

изпълняване на задачи през регулярни времеви интервали. Таймерите

предоставят такава услуга. Те са обекти, които известяват приложението

при изтичане на предварително зададен интервал от време. Таймерите са

полезни в редица сценарии, например, когато искаме да обновяваме

периодично потребителския интерфейс с актуална информация за статуса

на някаква задача или да проверяваме състоянието на променящи се

данни.

**System.Timers.Timer**

Класът предоставя събитие за изтичане на времевия интервал **Elapsed**,

което е делегат от тип **ElapsedEventHandler**, дефиниран като:

public delegate void ElapsedEventHandler(

object sender, ElapsedEventArgs e);

При изтичане на интервала, указан в свойството **Interval**, таймерът от

тип **System.Timers.Timer** ще извика записалите се за събитието методи,

използвайки нишка от пула. Ако използваме един и същ метод за

получаване на събития от няколко таймера, чрез аргумента **sender** можем

да ги разграничим. Класът **ElapsedEventArgs** чрез свойството **DateTime**

**SignalTime** ни предоставя точното време, когато е бил извикван метода.

За стартиране и спиране на известяването, можем да извикаме съответно

**Start()** и **Stop()** методите. Свойството **Enabled** ни позволява да инструк-

тираме таймера да игнорира събитието **Elapsed**. Това прави **Enabled**

функционално еквивалентно на съответните **Start()** и **Stop()** методи.

Когато приключим с таймера, трябва да извикаме **Close()**, за да осво-

бодим съответните системни ресурси.

**System.Threading.Timer**

**System.Threading.Timer** прилича на **System.Timers.Timer** и също

използва пула с нишки. Основната разлика е, че той позволява малко по-

разширен контрол – може да указваме кога таймера да започне да

отброява, както и да предаваме всякаква информация на метода за

обратни извиквания чрез обект от произволен тип. За да ползваме

**System.Threading.Timer**, трябва в конструктора му да подадем делегат от

тип **TimerCallback**, дефиниран като:

public delegate void TimerCallback(object state);

При всяко изтичане на времевия интервал, ще бъдат извиквани методите

в този делегат. Обикновено като обект за състояние има полза да

подаваме създателя на таймера, за да можем да използваме същия метод за обратни извиквания за обработка на събития от множество таймери.

Другият параметър в конструктора на таймера е времевият интервал. Той

може и да бъде променен впоследствие с извикване на **Change(…)** метода.

**System.Threading.Timer** не предлага удобен начин за стартиране и

спиране. Неговата работа започва веднага след конструирането му (по-

точно след изтичането на подаденото стартово време) и прекъсването му

става само чрез **Dispose()**. Ако искаме да го рестартираме трябва да

създадем нов обект.

**System.Windows.Forms.Timer**

Пространството от имена **System.Windows.Forms** съдържа още един клас

за таймер, който е със следната дефиниция:

public class Timer : Component, IComponent, Idisposable

{

public Timer();

public bool Enabled{virtual get ; virtual set;}

public int Interval {get; set;}

public event EventHandler Tick;

public void Start();

public void Stop();

}

Въпреки, че методите на **System.Windows.Forms.Timer** много приличат на

тези на **System.Timers.Timer**, то **System.Windows.Forms.Timer** не изпол-

зва пула с нишки за обратните извиквания към Windows Forms приложе-

нието. Вместо това, през определено време той пуска Windows съобще-

нието **WM\_TIMER** в опашката за съобщения на текущата нишка.

Използването на **System.Windows.Forms.Timer** се различава от употребата

на **System.Timers.Timer**, само по сигнатурата на делегата за обратни

извиквания, който в случая е стандартният **EventHandler**.