Иван Иванов
Петър Стойков

ВИЗУАЛНО ПРОГРАМИРАНЕ

Ръководство за дистанционно обучение

София, 2011

Въведение

В дисциплината Визуално програмиране се разглеждат най-често използваните контроли и контейнери за създаване на ***Windows***-приложения. Разглеждат се възможностите за създаване на лесни, но предлагащи големи възможности потребителски интерфейси. Разглеждат се основните свойства и събития на някои контроли и контейнери, като са приведени примери, демонстриращи тяхното използване. Също така се разглеждат различни методи за реализация на разнообразни събития с контроли и контейнери.

Конкретно се изучават:

- Контролите ***Label*** и ***LinkLabel*** за изобразяване на информация за потребителите;
- Контрола ***Button*** и съответното събитие ***Click*** за предоставяне възможност на потребителите за реализират различни функционалности в разработваните приложения;
- Контролите ***TextBox*** и ***RichTextBox*** за създаване на елементарен текстов редактор;
- Контролите ***CheckBox*** и ***RadioButton*** и начините за тяхното използване.
- Контейнера ***GroupBox*** и неговото влияние върху контролите ***CheckBox*** и ***RadioButton***;
- Контролите ***CheckedListBox*** и ***ListBox*** за работа със списъци;
- Контрола ***ListView*** и компонента ***ImageList***, които предоставят възможност за изобразяване на списъци по различни начини;
- Контейнера ***TabControl*** за групиране на контроли в различни страници в една форма.

Основна цел на дисциплината

Дисциплината е предназначена за напреднали, които вече познават езика C#. В курса обучаемите се запозбнават с основните контроли и контейнери за създаване на ***Windows***-приложения. След завършване на курса обучаемите трябва да могат самостоятелно да реализират изпълнението на средно големи програмни проекти.

Тема първа: Въведение във визуалното програмиране

Учебната тема "Въведение във визуалното програмиране" въвежда в проблематиката на курса, обосновава необходимостта от изучаването на визуалното програмирането, разглежда няколко програми, реализирани във визуланата среда за програмиране ***Microsoft Visual Studio 2010 Express***.

След нейното усвояване обучаемите:

- ще добият представа за развитието и възможностите на визуалното програмиране;
- ще разберат йерархията на контролите в пространството на имената ***System.Windows.Forms***;
- ще изучат някои от основните свойства и събития на контролите;
- ще се научат да създават елементарни ***Windows***-приложения.

Темата включва седем учебни въпроса:

1. Развитие
2. Контроли
3. Свойства
4. Проектиране на форми
5. Свойствата Anchor и Dock
6. Обобщаващ пример
7. Събития

Съдържание на учебната тема

1.1. Развитие

През май 1991 година, скоро след въвеждането на операционната система ***Windows 3.0*** (1990 г.), ***Microsoft*** пусна и нова версия на ***Basic***, наречена ***Visual Basic***. И операционната система и езикът за програмиране станаха мигновено хитове и бяха посрещнати с огромен ентусиазъм. ***Visual Basic*** предложи на програмистите инструментални средства за създаване на извънредно детайлизирани интерфейси за потребителя посредством интуитивни конструкторски форми и прост за изучаване език за програмиране. Тези две успешни съчетавания създадоха средата за бързо разработване на приложения ***RAD*** (*Rapid Application Development*). Едно от предимствата, предоставени от средствата на ***RAD***, такава като ***Visual Basic***, е че те осигуряват достъп до редица предварително създадени контроли, които могат да се използват за бързо изграждане на потребителски интерфейс за приложения.

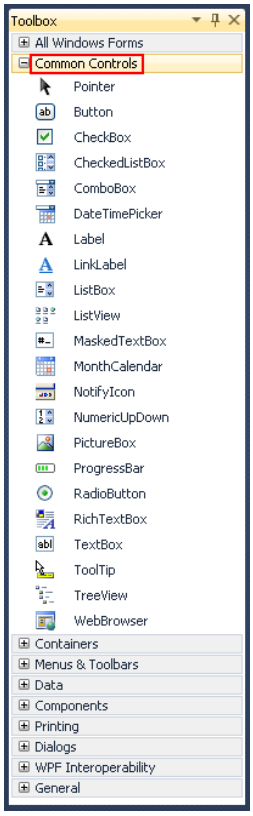
При разработване на повечето приложения с помощта на ***Visual Basic*** за ***Windows*** се използват средствата на ***Forms Designer*** (*конструктор на форми*). Създаването на потребителски интерфейс се реализира с преместване на контролите от панела с инструменти на формата и тяхното разполагане там, където те трябва да се изобразяват по време на изпълнение на програмата. С двойно щракване върху контрола се добавя дескриптор на дадения контрол. Контролите, предоставени от ***Microsoft***, а така също и допълнителни нестандартни контроли, които могат да се придобият допълнително, снабди програмистите с невероятно обширен арсенал от многократно използваем, прецизно тестван код, достъпен единствено с щракване върху бутона на мишката. Този подход за изграждане на приложения е достъпен и на разработчиците на езика ***C#*** посредством ***Visual Studio***.

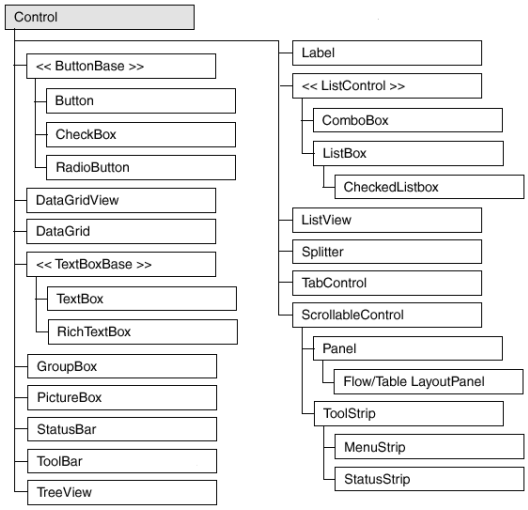
Всички досега създавани програми в изучавания курс за ***C#*** бяха конзолни, тоест не използваха богатите възможности на ***GUI*** (*Graphic User Interface – графичен потребителски интерфейс*). На практика такива конзолни приложения се създават извънредно рядко. Досега те бяха използвани само за фокусиране на вниманието при обсъждане възможностите на езика ***C#***. В дисциплината „Визуално програмиране” ще бъде разгледана техниката за създаване на пълноценни ***Windows*** приложения.

Тази техника в определена степен зависи от използваната система за програмиране. В случая ще бъде използвана ***Microsoft Visual Studio 2010 Express*** и по-конкретно ***Microsoft Visual C# 2010 Express***.

1.2. Контроли

Работейки с ***Windows Forms***, в действителност се работи с пространството на имената ***System.Windows.Forms***. Това пространство на имената е определено от директивата ***using*** в един от файловете, съдържащ класа ***Form***. Повечето контроли в структурата на ***.NET*** са производни от класа ***System.Windows.Forms.Control***. Този клас определя основните функционални възможности на контролите, вследствие на което много свойства и събития на контролите са идентични. Много от тези класове са базови за други контроли, както например с класовете ***ButtonBase***, ***TextBoxBase***, ***ListControl*** (фиг. 1.1). В графичната среда на ***Visual Studio*** контролите се намират в прозореца ***Toolbox***, в който същите са групирани в отделни секции – ***Common Controls***, ***Containers***, ***Menu & Toolbars*** и т.н.





Легенда:
<< Абстрактен клас >>

Фиг. 1.1. Йерархия на някои контроли

1.3. Свойства

Свойството е разновидност на член-класа. Като правило свойството съчетава в себе си поле[1] с методи за достъп към него. Полето най-често се създава за да бъде достъпно за потребителя на обекта, но при това е желателно да се съхрани управлението над операциите, разрешени за това поле. Например, да се ограничи диапазона на стойности, присвоявани на даденото поле.

Свойствата са специални методи, които на практика се състоят от ***име*** на свойството, два метода ***get( )*** и ***set( )*** и някаква текуща стойност, резервирана в думата ***value***. Методът ***get( )*** връща текущата стойност на свойството, а методът ***set( )***

[1] Стойков, П., И. Иванов. Програмиране на C#: Част 1. – София : Фараго, 2010.

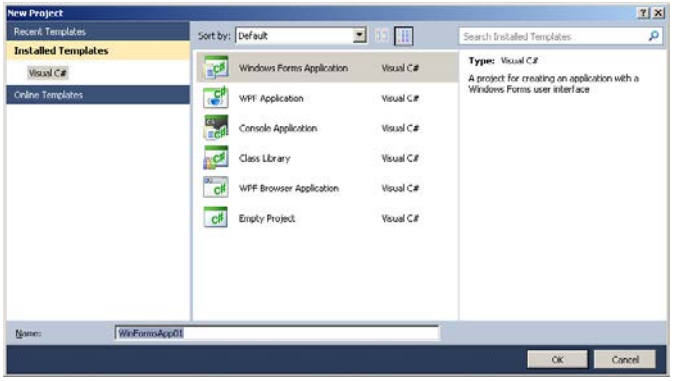
присвоява на свойството нова стойност на ***value***. Тези методи се извикват автоматично всеки път, когато трябва да се получи стойността на свойството или същата да се промени.

Всички контроли притежават редица свойства, които служат за манипулиране на тяхното поведение. Базовият клас на повечето контроли ***System.Windows.Forms.Control*** притежава редица свойства, които другите контроли наследяват непосредствено или ги заместват за осигуряване на едно или друго нестандартно поведение.

1.4. Проектиране на форми

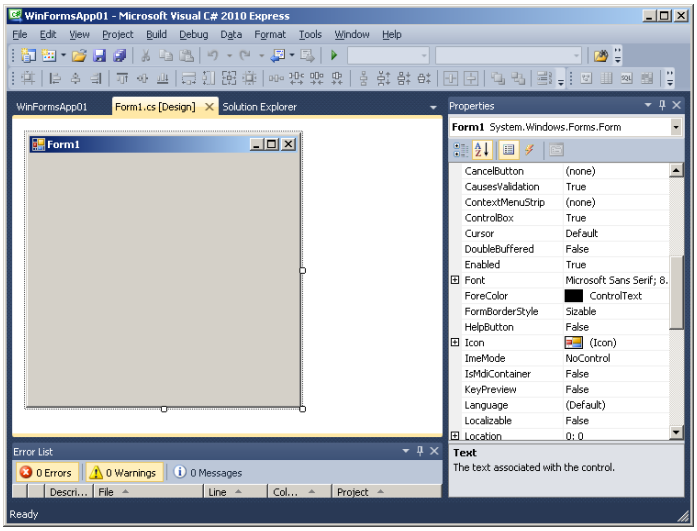
Ще бъде разгледан пример на проста диалогова програма, която позволява да се разкрият основните стъпки за създаване на ***Windows***-приложение.

Ако за създаването на конзолна програма е достатъчно само да се укажат използваните пространства на имената, то за създаване на ***Windows***-приложения е необходима известна подготвителна работа. За целта от главното меню на системата за програмиране се избира командата ***File*** ► ***New*** ► ***Project*** или се щраква на съответния инструментален бутон. В прозореца ***New Project*** се маркира ***Windows Forms Application***, както е показано на фиг. 1.2, задава се име на проекта в полето ***Name***, в случая ***WinFormsApp01***, и се натиска бутона ***OK***.



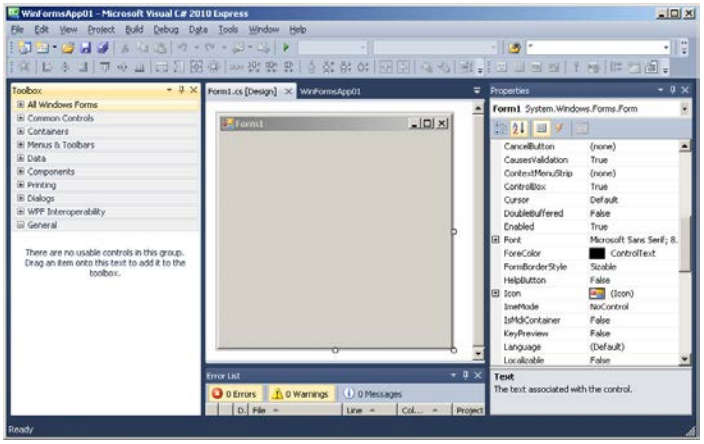
Фиг. 1.2. Прозорец ***New Project***

На екрана се появява прозорецът на дизайнера за форми с празна форма (фиг. 1.3).



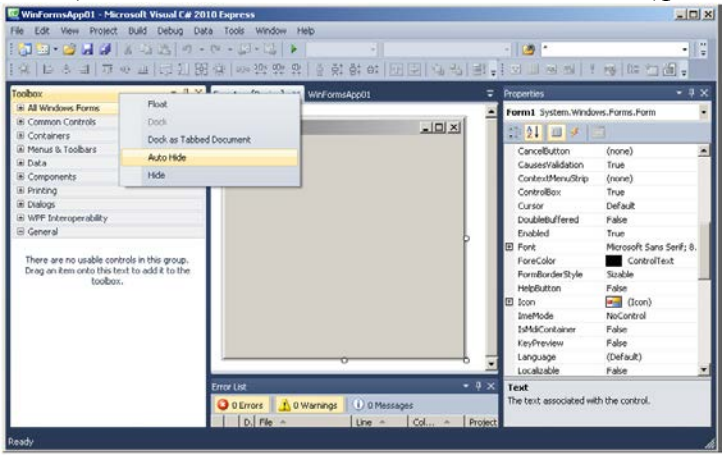
Фиг. 1.3 Прозорец на дизайнера на форми

Възможно е на екрана да липсват два важни прозореца ***Toolbox*** (с подготвени управляващи компоненти) и ***Properties*** (с указани значения на свойствата на избрания компонент). Същите могат да се активират от главното меню с командите ***View ► Toolbox и View ►Properties Window*** (фиг. 1.4.).



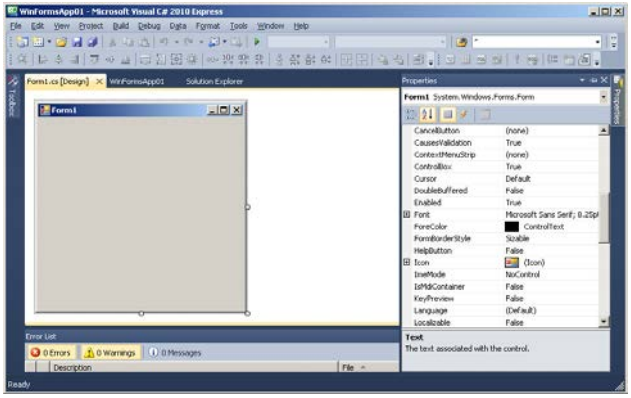
Фиг. 1.4. Прозорец на дизайнера на форми
с прозорците Toolbox и Properties

При щракване с десния бутон върху заглавната лента на съответния прозорец (***Toolbox*** или ***Properties***) се появява еднотипно контекстно меню (фиг. 1.5).



Фиг. 1.5. Контекстно меню появяващо се на прозорците ***Toolbox*** *и* ***Properties***

С помощта на командата ***Auto Hide*** прозорецът може да се преведе в състояние, когато се появява на екрана само след поставяне на маркера върху ивицата ***Toolbox*** (респективно ***Properties***). На фиг. 1.6 ***Toolbox*** и ***Properties*** са приведени в състояние ***Auto Hide*** и маркерът на мишката е поставен върху ивицата ***Properties*** с което той е активиран.

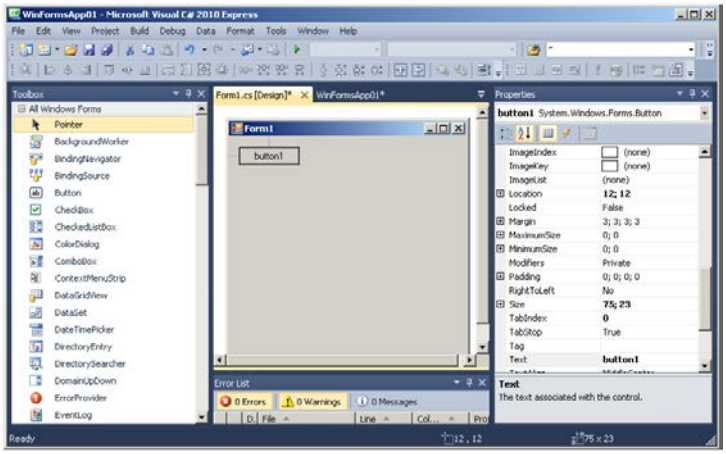


Фиг. 1.6. ***Toolbox*** *и* ***Properties*** *са приведени в състояние* ***Auto Hide***

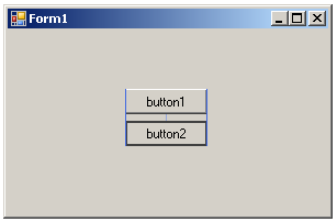
Проектиране на формите се изразява в пренасянето на необходимите компоненти от прозореца ***Toolbox*** във формата и фиксиране на необходимите стойности на някои от неговите свойства в прозореца ***Properties***. За пренасяне на компонент е необходимо да се щракне два пъти върху него в прозореца на Toolbox, или да се използва механизмът за *пренасяне* (***Drag&Drop***), като се хване компонента с мишката и се влачи до преместването му във формата.

Бутонът може да се премести към северозападния (горния ляв) ъгъл на формата. При приближаване на контрола към края на формата се появяват линии, които започват от северозападния край и са насочени към центъра на бутона. Контролът може да се премести извън границите на линиите за привързване или да остане на дадената позиция. Линиите за привързване при преместване на бутона в северозападния ъгъл на формата са показани на фиг. 1.7.

Ако бутонът се постави в центъра на формата и от ***Toolbox*** се добави още един бутон под съществуващия, то се появяват нови линии за привързване. Тези линии за привързване позволяват да се подравняват контролите така, че да се разполагат непосредствено над/под или на една височина с други контроли. При преместване на нов бутон спрямо съществуващ други линии за привързване позволяват да се разполагат бутоните на предварително определено разстояние между тях. Видът на формата при разполагане на бутоните в непосредствена близост един до друг е показан на фиг. 1.8.



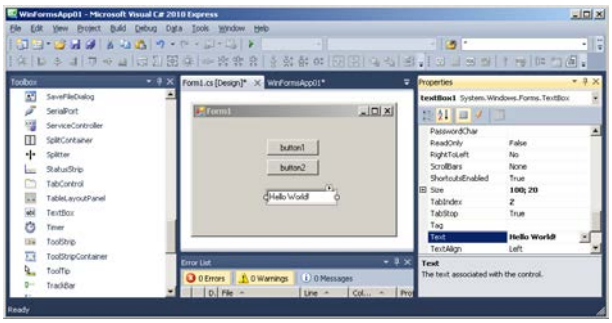
Фиг. 1.7. Линии за привързване при преместване на бутона в северозападния ъгъл на формата



Фиг. 1.8. Линии за привързване при разполагане на бутоните в непосредствена близост един до друг

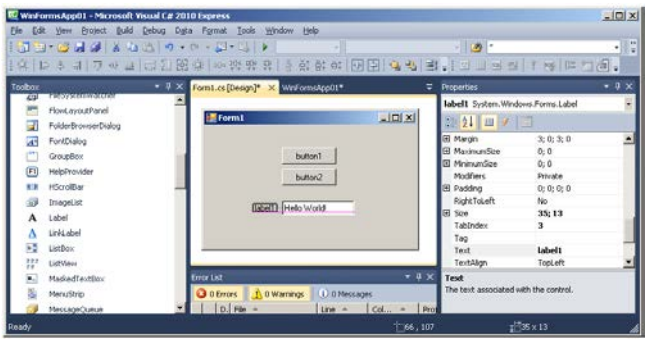
Ако се промени размерът на един от бутоните, например ***button1*** да се направи по-широк, а след това се пристъпи към промяна на размера на другия бутон, в случая ***button2***, когато неговата ширина стане равна на ширината на първия бутон се появяват линии за привързване (сини на цвят) с помощта на които се подпомага изравняването на ширината на бутоните.

Във формата, аналогично на добавянето на бутон, могат да се добавят и други контроли, например ***TextBox***. Същият може да се разположи под бутоните и в него да се напише текста „*Hello World!*”. Въвеждането на „*Hello World!*” става с помощта на промяна на неговото свойство ***Text*** в ***Properties*** (фиг. 1.9).



*Фиг. 1.9. Въвеждане на текст в **TextBox***

Отляво на контрола ***TextBox*** може да се добави контрол ***Label***. При разполагането на контрола ***Label*** се появяват линии за привързване, които позволяват привързването му към горния или долния край на контрола ***TextBox***. Тук се появява и трета линия за привързване (виолетова на цвят), която позволява да се подравнят текстовете в контролите ***TextBox*** и ***Label*** (фиг. 1.10).



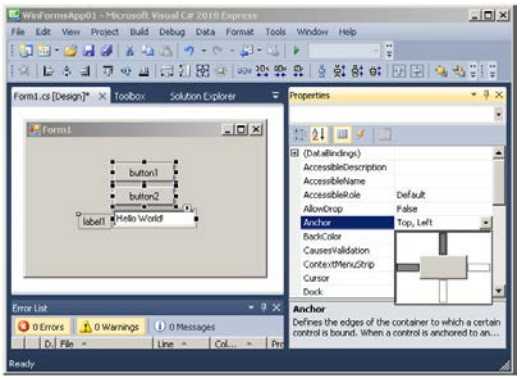
Фиг. 1.10. Линия за привързване, чрез която се подравняват текстовете

1.5. Свойствата Anchor и Dock

Тези две свойства са особено полезни при конструирането на форми. Запазването на първоначалния дизайн на формите при изменение на размера на прозореца е от съществено значение. Преди за тази цел се изискваше да се напишат много редове програмен код. Много програмни продукти решават този проблем по контролиран начин като просто забраняват изменение на размера на прозореца. Такова решение е най-лесно, но не винаги е най-добро. Свойствата ***Anchor*** и ***Dock***, появили се в структурата на ***.NET***, позволяват решаването на този проблем без необходимостта от написване на допълнителен сорс код.

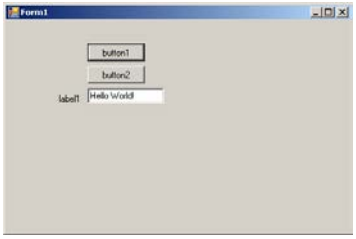
Свойството ***Anchor***[2] указва поведението на контрола при изменение на размера на прозореца. Контролът може да се настрои така, че да съхранява размерите си, като запазва позицията си спрямо краищата на прозореца, или да променя размерите си като запазва пропорциите към краищата на прозореца.

На фиг. 1.11 е показано как с помощта на свойството ***Anchor*** трите контрола са фиксирани в горен ляв ъгъл. За да се реализира групово изменение на свойството е необходимо всичките контроли да бъдат маркирани, както е показано на фигурата.



Фиг. 1.11. Групово установяване на свойството ***Anchor*** *в състояние* ***Top, Left***

След стартиране на приложението, при изменение размера на формата всичките контроли остават закотвени към горния ляв ъгъл, което е видно от фиг. 1.12.

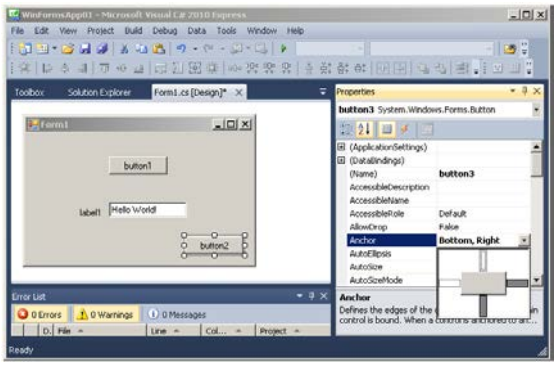


Фиг. 1.12. При изменение размера на формата, закотвените контроли остават фиксирани към горния ляв ъгъл на същата

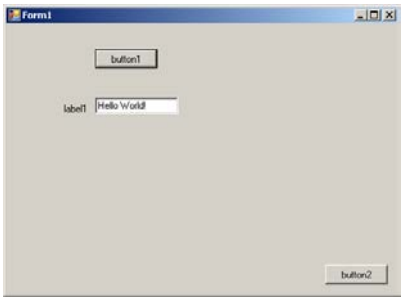
Ако се фиксира бутон (в случая ***button2***) към долния десен ъгъл (виж фиг. 1.13), то при изменение на размерите на формата, същият контрол вече е привързан към югоизточния ъгъл на формата (виж фиг. 1.14).

---

[2] anchor – котва, анкър, желязна съединителна скоба.

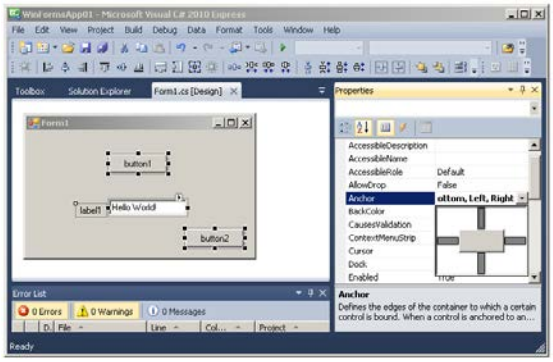


Фиг. 1.13. Установяване на свойството ***Anchor*** *за* ***button2*** *в състояние* ***Bottom, Right***



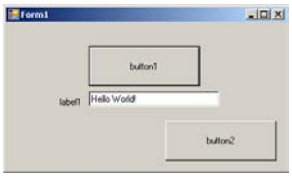
Фиг. 1.14. При изменение размера на формата, закотвените контроли остават фиксирани към указаните ъгли

За да се запазят пропорциите на контролите спрямо размера на формата е необходимо да се фиксират контролите, чрез свойството ***Anchor*** към четирите страни на формата, както е показано на фиг. 1.15.



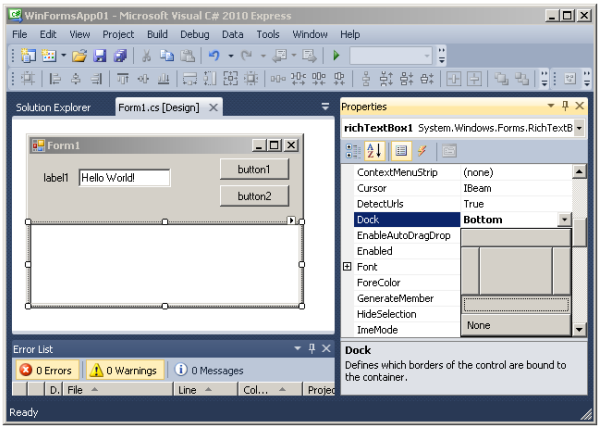
Фиг. 1.15. Установяване на свойството ***Anchor*** *за всички контроли от формата в състояние* ***Top***, ***Bottom, Left, Right***

След стартиране на приложението, при изменение размера на формата бутоните изменят пропорционално размерите си спрямо размера на формата, а текстовото поле изменя само ширината, което е видно от фиг. 1.16.



Фиг. 1.16. Контролите променят размерите си пропорционално на размерите на формата

Свойството ***Dock***[3] указва, че контролът трябва да се допира към края на своя контейнер. Ако потребителят промени размера на прозореца, контролът остава допрян към края на прозореца. Ако, контролът трябва да се допира до долния край на контейнера си, то контролът променя размерите си и/или се премества така, че винаги да заема долната част на прозореца независимо от промяната на неговите размери. На показания пример (фиг. 1.17) двата бутона са закотвени към горния десен ъгъл, текстовото поле и етикетът – към горен ляв ъгъл, а контролът ***richTextBox1*** с помощта на свойството ***Dock*** в положение ***Bottom***.



*Фиг. 1.17. Използване на свойството **Dock** за привързване на контрола **richTextBox1** към долния край на контейнера*

След стартиране на приложението, при изменение размера на формата контролите, фиксирани със свойството ***Anchor***, остават закотвени към съответните ъгли на контейнера. Контролът ***richTextBox1***, който е фиксиран към долния край на контейнера с помощта на свойството ***Dock*** продължава да заема цялата ширина на контейнера, без да се променя по височина (виж фиг. 1.18).

*Фиг. 1.18. Контролът **richTextBox1** продължава да заема цялата ширина на контейнера при изменение на размера на формата*

[3] dock – док, пристанище, скачвам космически кораби в космоса.

1.6. Обобщаващ пример

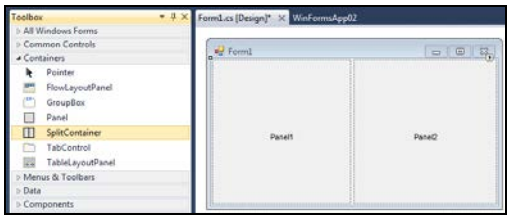
Да се разработи приложение с дизайн, показан на фиг. 1.19.

Фиг. 1.19. Разработваното приложение трябва да има следния дизайн

За реализацията на заданието ще бъдат изпълнени следните стъпки:

Стъпка 1. Създаване на ново приложение с име ***WinFormsApp02*** (фиг. 1.2 и 1.3);

Стъпка 2. Вмъкване във формата на контейнера ***SplitContainer*** (фиг. 1.20);

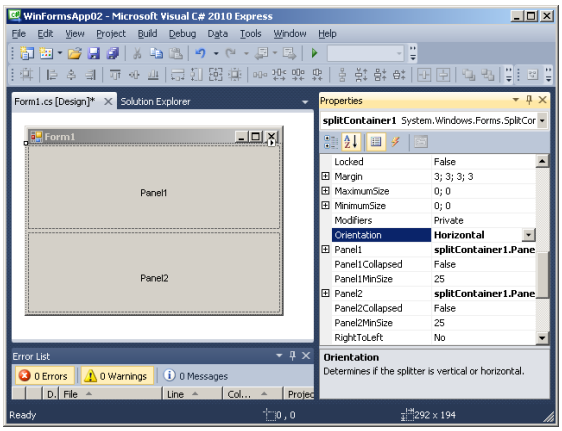


Фиг. 1.20. Добавяне на контейнера ***SplitContainer***

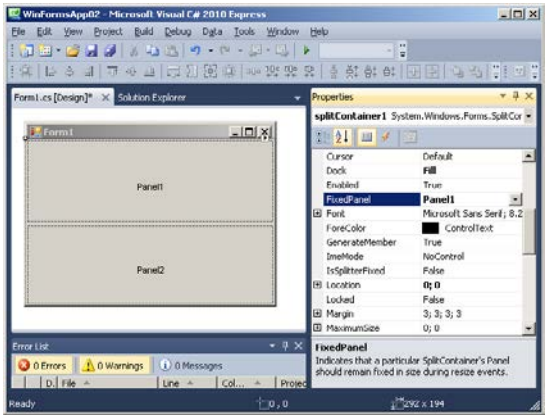
Стъпка 3. На свойството ***Orientation*** от контейнера ***SplitContainer*** се задава стойност ***Horizontal*** (фиг. 1.21);

Стъпка 4. На свойството ***FixedPanel*** от контейнера ***SplitContainer*** се задава стойност ***Panel1*** (фиг. 1.22). Вертикалният размер на ***Panel1*** се фиксира съгласно текущия размер от дизайнера на формата;

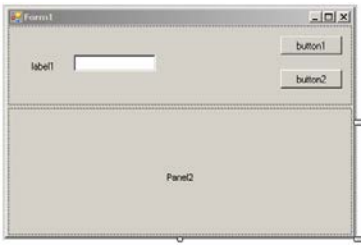
Стъпка 5. В ***Panel1*** се добавят необходимите контроли на формата съгласно фиг. 1.23. С помощта на свойството ***Anchor*** контролите ***button1*** и ***button2*** се фиксират в положение ***Top***, ***Right***. Свойството ***Anchor*** на контролите ***label1*** и ***textBox1*** по подразбиране са със стойности ***Top***, ***Left***;



Фиг. 1.21. Свойството ***Orientation*** получава стойност ***Horizontal***

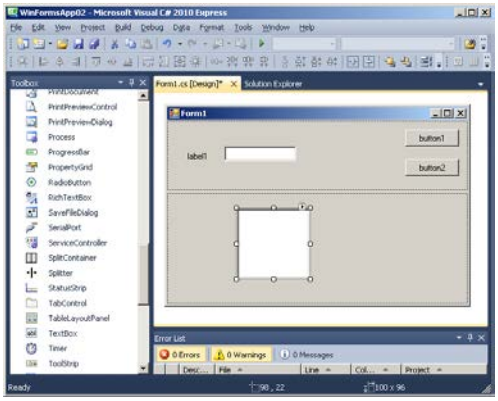


Фиг. 1.22. Фиксиране на вертикалния размер на ***Panel1***



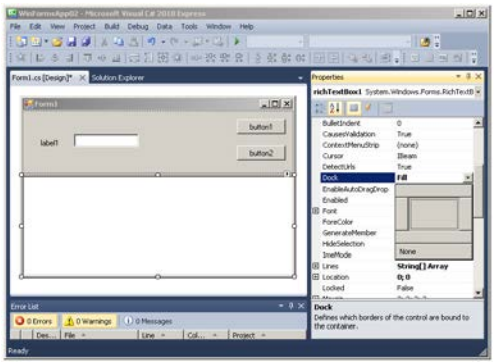
Фиг. 1.23. В ***Panel1*** *са добавени необходимите елементи, съгласно заданието*

Стъпка 6. В ***Panel2*** се добавя контрола ***RichTextBox*** (виж фиг. 1.24);



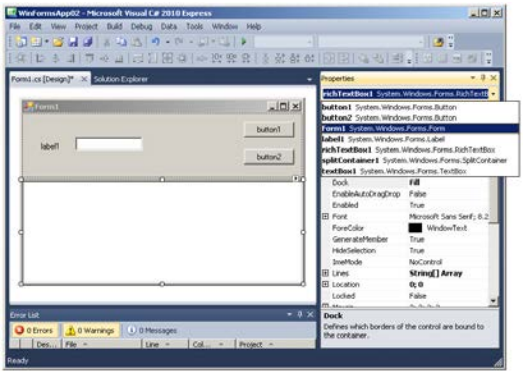
Фиг. 1.24. В ***Panel2*** *е добавен контрола* ***richTextBox1***

Стъпка 7. На свойството ***Dock*** за контрола ***richTextBox1*** се присвоява стойност ***Fill*** (виж фиг. 1.25);



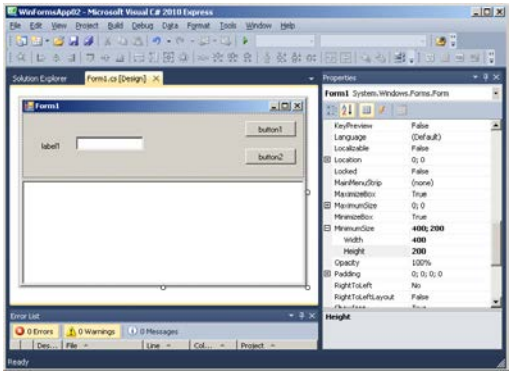
Фиг. 1.25. Свойството ***Dock*** *на контрола* ***richTextBox1*** *е със стойност* ***Fill***

Стъпка 8. От падащия списък на прозореца ***Properties*** се избира ***Form1*** (виж фиг. 1.26), с което се премества фокуса върху ***Form1*** на проекта;



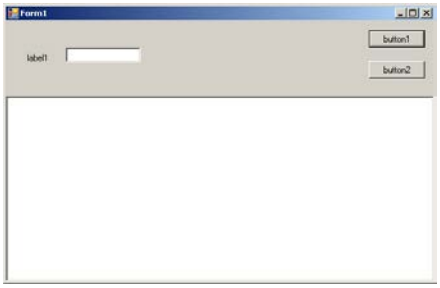
Фиг. 1.26. Преместване на фокуса върху ***Form1***

Стъпка 9. На свойството ***MinimumSize*** се задават стойности ***Width=400 и Height=200*** (виж фиг. 1.27), с което се фиксират минималните размери на формата ***Form1*** при изменението на размера от потребителя;



Фиг. 1.27. Фиксиране на минималните размери на формата

С това заданието е изпълнено. На следващите фигури са показани различни състояния на стартираното приложение. На фиг. 1.28 е показано приложението с увеличени размери, а на фиг. 1.29 – минималният размер на приложението.



Фиг. 1.28. Стартираното приложение с увеличени размери

Фиг. 1.29. Стартираното приложение с минималните размери на формата

Към разработената форма впоследствие ще бъдат добавени необходимите събития.

1.7. Събития

Събитията (***events***) определят реакцията на класа на едни или други действия на потребителя (при преместване на мишката, например) или на изменения в работната програма (например, като реакция на възникнала изключителна ситуация). Със събитията са свързани делегати – специални обекти, които предават управлението при възникване на събития. За деклариране на събития се използва запазената дума ***event***, а за деклариране на делегат – ***delegate***.

Делегатите представляват ***.NET*** типове, които описват сигнатурата на даден метод (броя, типа и последователността на параметрите му) и връщания от него тип. Делегатите приличат на указателите към функции в C и C++ – съдържат силно-типизиран указател (референция) към метод. Те са структури от данни, които приемат като стойност методи, отговарящи на описаната от делегата сигнатура.

В обектно-ориентираното програмиране обектите изпращат събития (***events***) към своя притежател за да го уведомят при настъпване на интересна за него ситуация. Обектът, който предизвиква дадено събитие, се нарича изпращач на събития (***event sender***). Обектът, който получава дадено събитие, се нарича получател на събитието (***event receiver***).

За да получават дадено събитие получателите му предварително се абонират за него (***subscribe for event***). Под абонамент на събитие се разбира предоставяне на код, който трябва да се изпълнява при генериране на дадено събитие, във вид на манипулатор на събитие (***event handler***[4]).

В ***.NET Framework*** абонирането, изпращането и получаването на събитията се поддържа чрез делегати и събития. Събитията в C# са специални инстанции на делегати, декларирани с ключовата дума ***event***. За променливите от тип събитие ***C#*** компилаторът автоматично дефинира операторите += и -= съответно за абониране за събитието и за премахване на абонамент. Събитията могат да предефинират кода за абониране и премахване на абонамент.

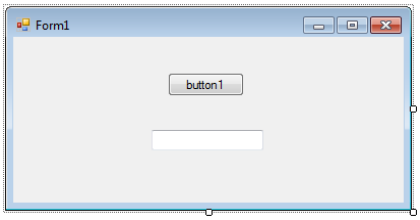
За събитието може да се абонират няколко контрола за обработване и тогава те всичките ще бъдат извиквани при генериране на това събитие. Тези контроли за обработване могат да са както и части от същия клас на обекта, който генерира даденото събитие, така и части от други класове.

---

[4] event handler – манипулатор (хендлър) на събитие = 1. Метод в програма, който се извиква автоматично винаги, когато възникне определено събитие. Английско-български тълковен компютърен речник : т. 1, стр. 285. – София : Софтпрес, 2005.

Самите контроли за обработване на събития са методи. Единственото ограничение за такава методи е, че типът на връщаната стойност и на параметрите трябва задължително да съответства на тези, които изисква събитието.

Последователността за обработка изглежда по следния начин: първоначално в приложението се създава обект, който може да генерира събитие. Като пример ще бъде разгледано приложение, в което обектът е ***button1*** *(бутон)*, а събитието е ***Click*** *(щракване върху бутона)* – (фиг. 1.30).

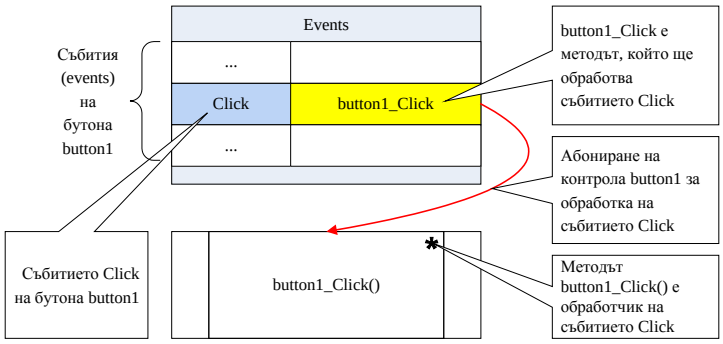


Фиг. 1.30. Обект ***button1***

Създава се метод за обработка на събитието ***button1_Click***, който ще обработва събитието ***Click***. Методът ще изобразява текст в полето ***textBox1***. Самият метод има следния вид:

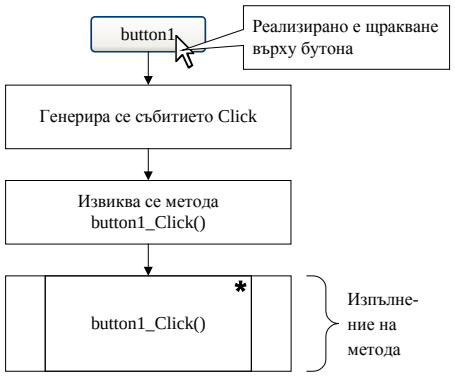
```
private void button1_Click(object sender, EventArgs e)
{
   textBox1.Text = "button1 е щракнат";
}
```

След това обектът ***button1*** се абонира за събитието ***Click***, чрез привързване на метода ***button1_Click*** към събитието ***Click*** на обекта ***button1*** – фиг. 1.31.

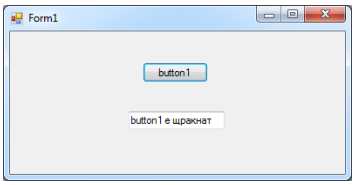


Фиг. 1.31. Абониране за събитие

И накрая, при генериране на събитието ***Click***, абонатът получава съобщение, което предизвиква извикване на метода ***button1_Click*** за обработване на събитието. Доколкото се използват стандартни методи, обектът, генериращ събитието, може да предава информация посредством параметри и по този начин събитията са много разнообразни. За примера такъв параметър може да бъде текстът на мигновеното съобщение, който се изобразява в текстовото поле (виж фиг. 1.32).



Резултат от изпълнение на събитието



Фиг. 1.32. Генериране на събитие и обработване на същото

В дадения случай се разглеждат конкретни видове събития, а именно, тези, които се генерират от контролите на ***Windows Forms***. Обикновено тези събития са свързани с действия, изпълнявани от потребителите. Например, когато потребителят щраква на бутон, той генерира събитие, указващо, че с него нещо е станало. Обработката на събитието е средство, посредством което програмистът може да снабди даден бутон с една или друга функционална възможност.

Класът ***Control*** определя редица събития, които са присъщи за всички контроли, използвани в тази част на учебника. Някои от тях са описани в табл. 1.2[5].

Таблица 1.2. Често използвани събития в класа ***Control***

Събитие	***Описание***
Click	Настъпва при щракване върху контрол. В някои случаи това събитие настъпва също при натискане на клавиш <***Enter***>
Doubleclick	Настъпва при двойно щракване върху контрол. Обработването на събитието ***Click*** за някои контроли, такива като ***Button***, напълно изключват възможността от извикване на събитието ***Doubleclick***.
DragDrop	Настъпва при завършване на операцията ***drag&drop***, т.е. при преместване на обекта върху контрола и освобождаване бутона на мишката.
DragEnter	Настъпва, когато преместваният обект се премества в границите на контрола.
DragLeave	Настъпва, когато преместваният обект напуска границите на контрола.
DragOver	Настъпва, когато обектът се пренася върху контрола.
KeyDown	Настъпва при натискане на клавиши в момента когато контролът се намира на фокус. Това събитие настъпва винаги преди събитията ***KeyPress*** и ***KeyUp***. Разликата между събитията ***KeyDown*** и ***KeyPress*** се състои в това, че ***KeyDown*** предава кода на натиснатия клавиш, а **KeyPress** – съответстващата стойност на ***char*** клавишите.
KeyUp	Настъпва при освобождаване на клавиш в момента, когато контролът се намира на фокус. Това събитие настъпва винаги след събитията ***KeyDown*** и ***KeyPress***.

[5] Пълното изброяване на събитията е дадено в документацията на ***.NET Framework SDK***.

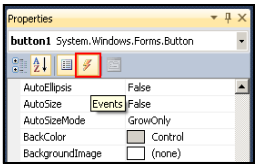
Събитие	*Описание*
GotFocus	Настъпва когато контролът получава фокус. Това събитие не следва да се използва за извършване на проверка за допустимост на контролите. В този случай вместо него трябва да се използват събитията ***Validating*** и ***Validated***.
LostFocus	Настъпва когато контролът получава фокус. Това събитие не следва да се използва за извършване на проверка за допустимост на контролите. В този случай вместо него трябва да се използват събитията ***Validating*** и ***Validated***.
MouseDown	Настъпва при разполагане на указателя на мишката върху контрола и натискане на левия бутона на мишката. Това събитие не е еквивалентно на събитието ***Click***, понеже ***MouseDown*** настъпва веднага след натискане на бутона на мишката и преди освобождаването му.
MouseMove	Настъпва непрекъснато по време на преместването на указателя на мишката над контрола.
MouseUp	Настъпва при разполагане на указателя на мишката върху контрола и освобождаване бутона на мишката.
Paint	Настъпва при рисуване в контрола.
Validated	Стартира се, когато контролът, на който свойството ***CausesValidation*** е ***True***, е готов да приеме фокуса. Това събитие се стартира след завършване на събитието ***Validating***, и показва, че проверката е завършена.
Validating	Стартира се, когато контролът, на който свойството ***CausesValidation*** е ***True***, е готов да приеме фокуса. Проверяваният контрол е този, който губи фокуса, а не този, който го получава.

Много от тези събития се използват в примерите в този учебник. Всички примери следват един и същи формат: първоначално се извършва визуално представяне на формата, избират се и се разполагат контролите, и чак след това се пристъпва към добавяне на методи за обработване на събитията. Вграждането на функционалност в методите е най-съществената част при създаването на проекта.

Съществуват три основни подхода за създаване на манипулатор на конкретно събитие.

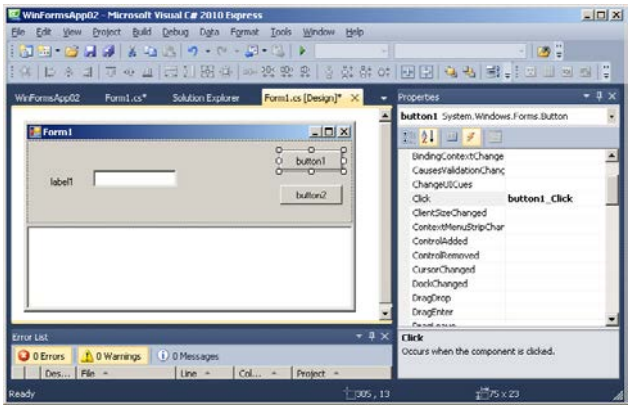
При двойно щракване върху контрола започва обработване на събитието, използвано по премълчаване за дадения контрол. Това събитие е различно за различните контроли. Ако това е необходимото събитие, това е много добре. В противен случай съществуват още два възможни подхода.

Вторият подход е посредством използването на списъка от събития в прозореца ***Properties***, който се изобразява при щракване на бутона ***Events*** (с пиктограмата мълния, показан на фиг. 1.33).

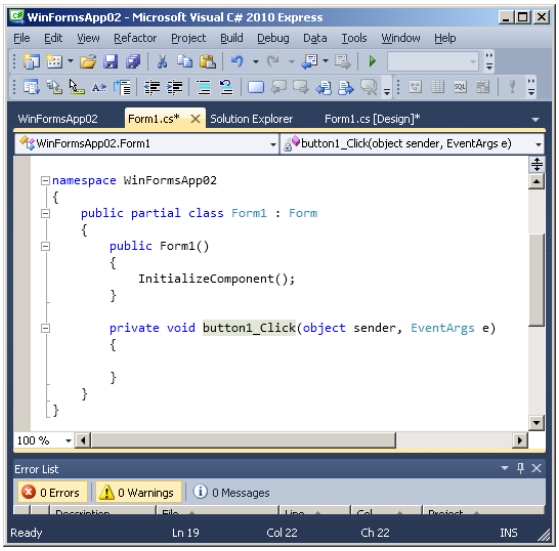


Фиг. 1.33. Извикване на списъка от събития

Маркираното събитие ***Click*** на фиг. 1.34 е събитие на контрола ***button1***, използвано по подразбиране. За да се добави типа на обработване на конкретното събитие е необходимо да се щракне два пъти върху името на събитието в списъка от събития. В резултат на това ще бъде генериран код за абонамент към контрола на събитието ***Click*** и сигнатура на метода ***button1_Click*** за обработване на същото събитие (виж фиг. 1.35).

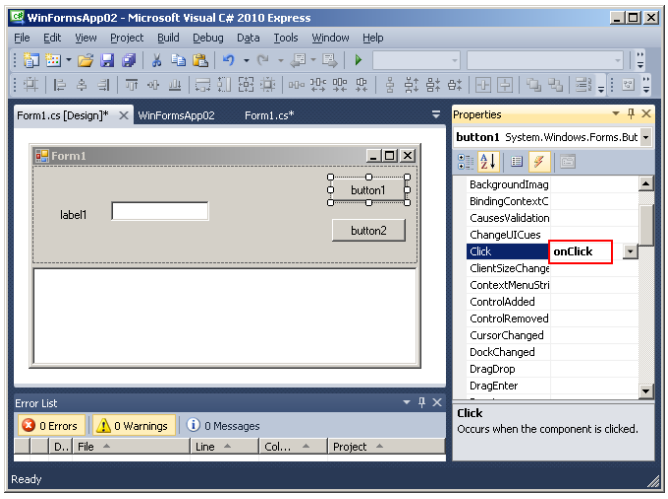


Фиг. 1.34. Изобразяване на списък от събития

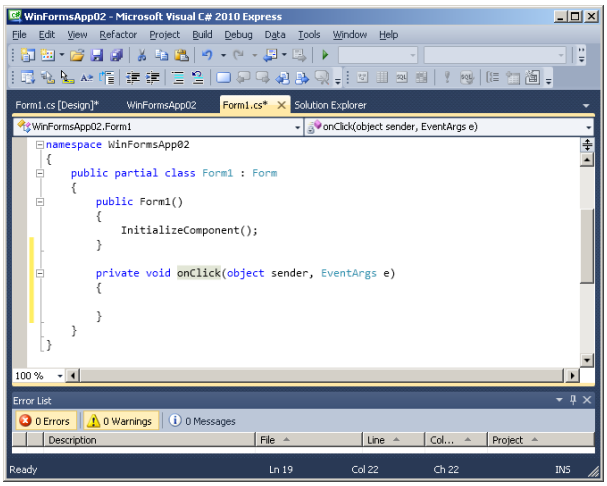


*Фиг. 1.35. Методът **button1_Click** за обработване на събитието **Click***

Друга възможност е да се въведе име на метода за обработване на конкретно събитие в поле, разположено непосредствено до даденото събитие в списъка ***Events***. На фиг. 1.36 е въведено име на метод ***onClick*** срещу събитието ***Click***. След натискане на клавиша **<*Enter*>** се генерира типът на обработване на събитието с името ***onClick*** (виж фиг. 1.37).



Фиг. 1.36. Изобразяване на списък от събития



Фиг. 1.37. Методът ***onClick*** *за обработване на събитието* ***Click***

Третият подход е посредством ръчно добавяне на код към събитието. Този подход ще се използва често в следващата глава като се добавя код в конструктора на форми след извикване на метода ***InitializeComponent( )***. Дори при въвеждане на кода, необходим за свързване със събитието, ***Visual Studio*** открива изпълняваното действие и предлага да добави в кода сигнатура на метода, като че ли операцията се изпълнява от ***Forms Designer***.

Всеки от последните два подхода изисква изпълнението на две действия – абониране за събитието (т.е. свързване с него) и създаване на съответствуваща сигнатура на метода за обработване. Двойното щракване върху контрола довежда до изпълнение на събитието. Ако опитът да се обработи друго събитие посредством заменяне на сигнатурата на метода на събитието, използвано по подразбиране, със събитие, което действително е необходимо да се обработва, се окаже безуспешен е необходимо да се измени също кодът за абонамент на събитието в метода ***InitializeComponent( )***. Поради това споменатият обходен път в действителност не може да послужи като по-бърз метод за обработване на конкретни събития.

Ключови думи

контроли контейнери	свойства събития

Препоръчана литература

1. **Иванов, И.**, Стойков П., Визуално програмирането, Част 1, Издателство “Фараго” С. 2011, - 128 с.
2. **Стойков, П.**, И. Иванов., Програмиране на C#: Част 1. - София : Фараго, 2010. – 128 с.
Стойков, П., И. Иванов. Програмиране на C#: Част 2. - София : Фараго, 2011. - 124 с.