5. Израда корисничког интерфејса: *Web* апликације (*ASP.NET Web Forms*) + слојевита архитектура

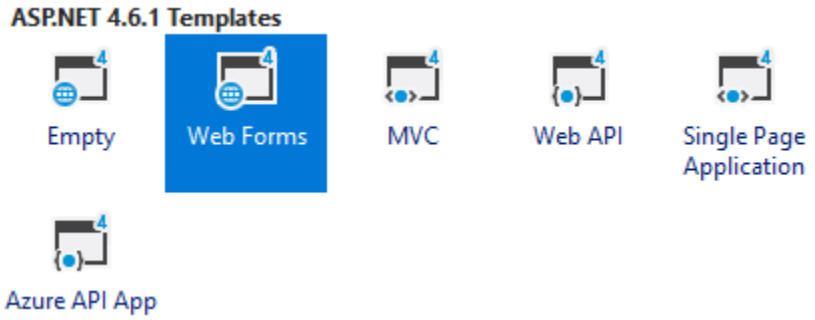
ASP.NET је део *.NET Framework*-а и представља *Microsoft*-ову платформа за развој *web* апликација. *ASP.NET* заправо нуди три основна развојна модела: ***Web Forms***, *MVC* и *Web Pages*. На овим вежбама бавићемо се *Web* формама које функционишу на доста сличан начин као класичне *Windows Forms* апликације јер имају исту *event-driven* структуру.
Свака *web* страница сачињена је од *HTML* кôда, па тако и оне које се креирају помоћу *ASP.NET Web* форми. Предност у раду са овим формама јесте у томе што није потребно напредно познавање *HTML*-а, већ ће се образци и контроле које креирамо аутоматски "претворити" у *HTML* који нама уопште и није битан како изгледа и не морамо га уопште читати.

5.1 Креирање пројекта у *Visual Studio* алату

Креирање *web* пројеката у *VS*-у је доста једноставно:

- *File – New – Project*
- Под *Visual C#* па под *Web* постоји опција *ASP.NET Web Application (.NET Framework)*

Након тога појавиће се прозор где се могу изабрати различите врсте *web* апликација које нуди *.NET* окружење. Нас интересује *Web Forms*.



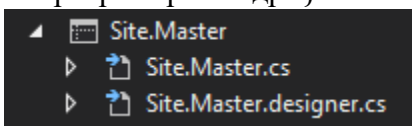
Када се креира овакав пројекат на основу *Web Forms* образца, по *default*-у добијамо већ креиран *basic web* сајт са убаченим основним елементима. Код *ASP.NET Web Forms* основна су три елемента:

- *Master page* - "главна" страница која се обично користи да дефинишемо "оквир" *web* сајта и у оквиру које убацујемо *web* форме. Обично постоји само једна у једном пројекту.
- *Web Form* - основни елемент са *.aspx* екстензијом који се највише користи у оваквом типу пројеката и који убацујемо на *master* страницу
- *User Control* – једна контрола у оквиру форме. Обично се користи за потребе поновне употребљивости (када неки део сајта хоћемо да поновимо на више места)

Осим ова три поменута елемента у подразумевани пројекат су убачени и многи други фајлови као што су фонтови, *CSS* стилови, *Javascript* скрипте од бројних *3rd party library* произвођача (нпр. *Bootstrap*). Као што је већ поменуто, за рад са *ASP.NET Web Forms* није потребно напредно познавање *HTML*-а, *CSS*-а и *JS*-а али је свакако пожељно.

5.2 *Master page*

У *ASP.NET Web Forms* ове странице нам омогућавају да креирамо једнообразни *layout* и дизајн у нашој апликацији. Није обавезно, али обично постоји само једна мастер страница у једном пројекту. Ова страница у себи садржи друге инвидуалне *content* странице као што је *web* форма. *Master page* се састоји из два дела: дизајна (*html*) и свог *code behind*-а где можемо у одређеним догађајима програмирати одређена понашања апликације:

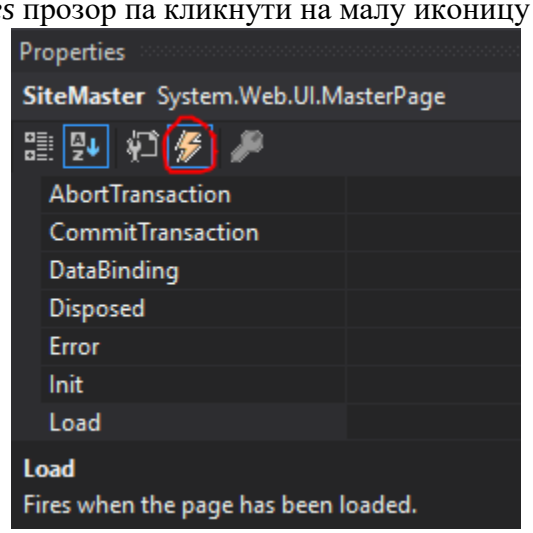


Site.Master без додатне екстензије је заправо *html* страница заједно са одређеним *asp* контролама које се комајлирају у *html*. *Site.Master.cs* је *code behind* где подразумевано већ имамо догађај *Page_Load* који нам омогућава да испрограмирамо нешто у тренутку када се страница учитава. *Site.Master.designer.cs* је фајл у коме скоро никад нећемо радити и садржи списак контрола које се налазе на страници (овај фајл је *auto-generated* и аутоматски се мења). На мастер страници може постојати већи број догађаја али користи се само неколико основних:

- *Load* (када се страница учита)
- *Init* (иницира се пре *Load* догађаја, када се контроле иницијализују)
- *Unload* (када се страница гаси)
- *Error* (када се догоди нека непредвиђена грешка)

Догађаје на мастер страници у *VS* алату можемо креирати на следећи начин:

- Десни клик на *master* страницу у *Solution Explorer*-у па затим *View Component Designer*
- Отворити *Properties* прозор па кликнути на малу иконицу у облику муње:



- Двоклик на догађај који желимо да креирамо и аутоматски ћемо прећи у *code behind* странице са креираним догађајем

5.3 *Web Form*

Представља један део на мастер страници који можемо потпуно одвојено дизајнирати, конфигурисати и програмирати. То су заправо *ASP.NET web* странице у којима се налази *C#* програмски кôд и имају екстензију ***.aspx*** . У нашем креираном пројекту аутоматски су се направила три таква фајла (*About.aspx, Default.aspx, Contact.aspx*) као примери ових страница. Да би смо "накачили" нашу *web* форму на мастер страницу потребна су два корака:

1. На мастер страници мора постојати *ASP.NET placeholder* којих може бити неограничен број и који се идентификују са својим *ID* атрибутом:

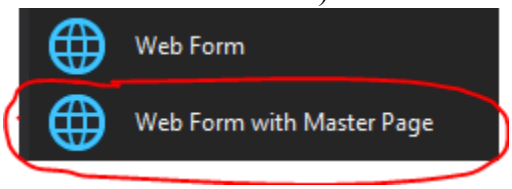
```
<asp:ContentPlaceHolder ID="MainContent" runat="server">
</asp:ContentPlaceHolder>
```

2. На нашој *web* форми треба креирати *ASP.NET content* таг који ће садржати атрибут *ContentPlaceHolderID* са вредношћу атрибута *ID* са мастер странице:

```
<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">
    <!-- neki sadržaj stranice -->
    <h1>TEST</h1>
</asp:Content>
```

Наравно, *Visual Studio* пружа и знатно лакше повезивање *web* форми и мастер страница и то на следећи начин:

- Десни клик на пројекат (или фолдер) па *Add* > *New Item* (на овај начин се и додају нови елементи у овој врсти пројеката)
- Изабрати *Web Form with Master Page* (обичан *Web Form* је независтан од мастер странице осим ако га ми сами не повежемо)



- Појавиће се прозор где треба да изаберемо мастер страницу са којом желимо да повежемо наш *web* образац

5.3.1 Догађаји *web* форми

Као и мастер странице, *web* обрасци такође имају догађаје и они се доста више користе него они код мастер страница. На *web* формама се могу додати исти догађаји који су горе поменути (*Load, Unload, Error*…), на исти начин (преко *VS* алата), али и неки додатни:

- *InitComplete* (окида се када се заврши иницијализација)
- *LoadComplete* (окида се након учитања странице)
- *PreLoad* (окида се пре учитавања странице)
- ...

5.4 Навигација

Прелазак са једне странице на другу се у *HTML*-у скоро увек врши помоћу најобичнијих линкова (елемент *<a>*). Исти случај је и у *ASP.NET Web Forms* апликацијама. На мастер страници постоји следећи део кôда:

```
<li><a runat="server" href="~/">Home</a></li>
<li><a runat="server" href="~/About">About</a></li>
<li><a runat="server" href="~/Contact">Contact</a></li>
```

Дакле, и овде се користе линкови с тим што се мора навести и добро познати атрибут *runat* са вредношћу "*server*". У оквиру *href* атрибута не поставља се вредност директно до неке друге *HTML* странице (како би се иначе радило са обичним *HTML*-ом) већ се поставља релативни (а може и апсолутни) линк до *.aspx* странице, односно до назива њене класе (не ставља се *About.aspx* већ само *About*).

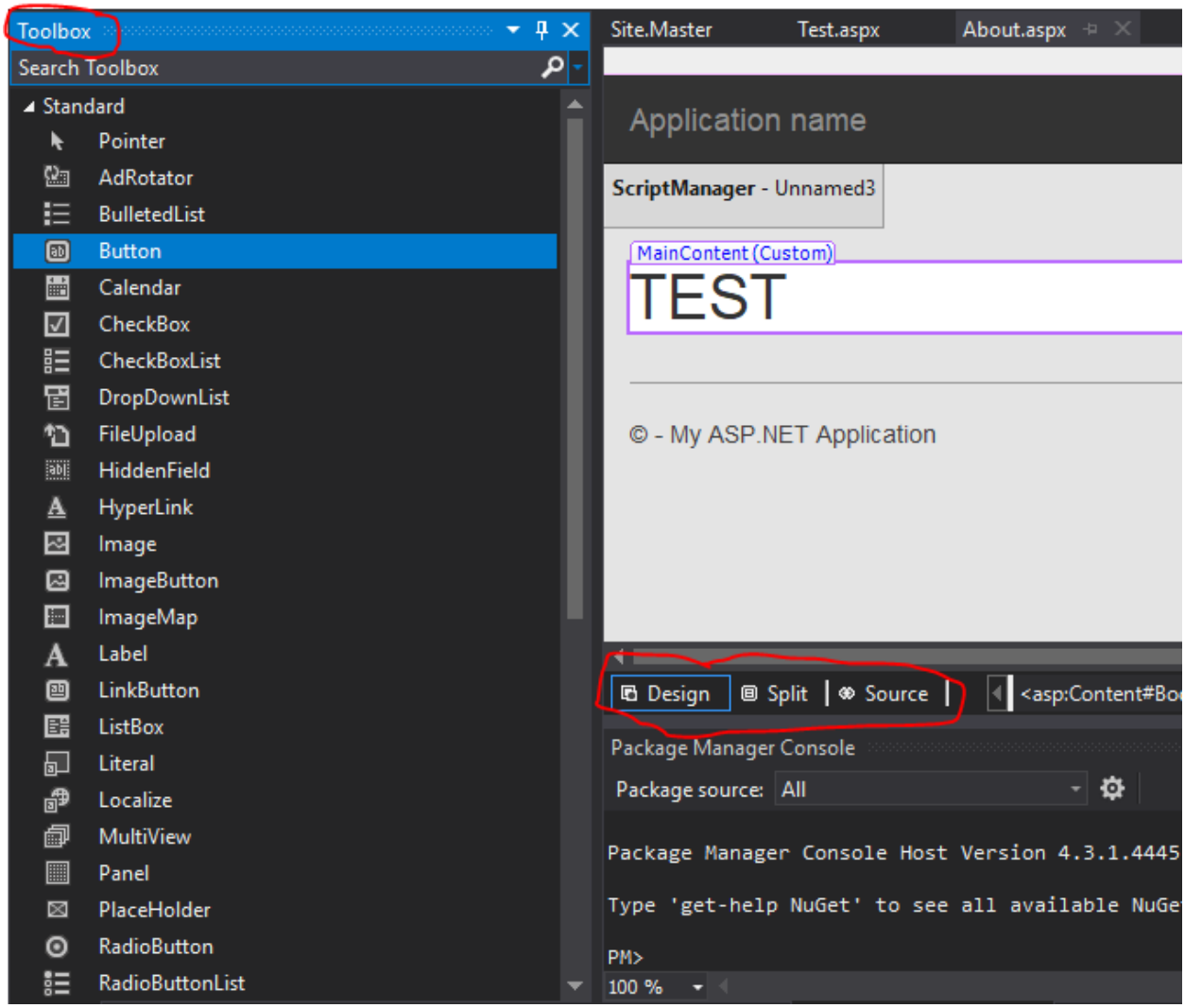
5.5 *Web* контроле

У *HTML*-у постоје бројне контроле које се користе у оквиру форми за различите врсте уноса или приказа података. Неке од њих су и: *textbox, radio button checkbox, button*… Такође ту су и бројни елементи на које се могу "качити" подаци као што је нпр. табела, *drop down* или обична листа. *ASP.NET Web Forms* нуди своје контроле које се након компајлирања заправо генеришу у класичне већ поменуте *HTML* елементе. Ове *Web Forms* контроле су у позадини заправо *C#* класе па самим тим нуде бројне могућности њиховог програмирања (подешавање вредности атрибута, креирање метода, догађаја итд.). Табела мапирања *ASP.NET Web Forms* контрола у *HTML* одговарајући елемент изгледа отприлике овако:

Класа (*ASP.NET* контрола)	Одговарајући *HTML* елемент
Label	**
Button	*<input type="submit">* (или *type="button"*)
TextBox	*<input type="text">* (или *type="password"*) или *<textarea>*
CheckBox	*<input type="checkbox">*
RadioButton	*<input type="radio">*
Hyperlink	*<a>*
LinkButton	*<a>* у коме се налази *<img>*
ImageButton	*<input type="image">*
Image	*<img>*
ListBox	*<select size="X">* (*X* је број редова)
DropDownList	*<select>*
CheckBoxList	листа или табела (*<table>*) са више тагова *<input type="checkbox">*
RadioButtonList	листа или табела (*<table>*) са више тагова *<input type="radio">*
BulletedList	уређена (* *) или неуређена (* *) листа
Panel	* *
Table, TableRow, TableCell	*<table>*, *<tr>*, *<td>* или *<th>*

Тагови *web* контрола се креирају на специфичан начин и то када се испред назива класе дода кључна реч *asp:* (са две тачке). Пример креирања *input* поља типа *text*: ***<asp:TextBox ID="txt" runat="server" />***. Специфични атрибут *runat* је доста чест у *ASP.NET* свету и подразумева то да ће се ова контрола "превести" на серверу у класичан *HTML* (како смо већ помињали). Поред тога, ове класе имају бројне атрибуте који се могу поставити или као класични атрибути у оквиру *<asp:...* тага, или кроз *code-behind* одређене странице или *web* форме.

Web контроле се могу поставити на страницу користећи и *VS* дизајнер па преко палете са контролама (*Toolbox*), слично као на обичним *Windows Forms* апликацијама али ипак је у овој ситуацији боље користити директно програмски кôд. Постоји чак и опција *split* која нам омогућава да истовремено користимо и *design* и *source* поглед:

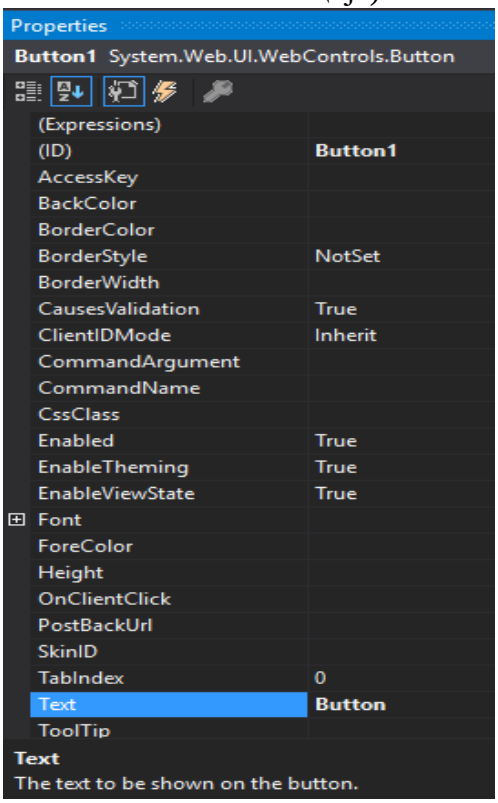


5.5.1 Атрибути

Скоро све поменуте контроле заправо наслеђују основну класу под називом *WebControl* па самим тим од ње наслеђују нека својства која се касније могу користити као атрибути на таговима. Нека основна својства (односно атрибути) су:

- *BackColor, ForeColor, BorderColor* (боја позадине, предњег плана и оквира контроле)
- *BorderWidth* (величина оквира контроле)
- *Enabled* (да ли је контрола активна или није, *true-false*)
- *Font* (фонт у оквиру контроле)
- *Height, Width* (висина и ширина)
- *ToolTip* (текстуална порука када корисник пређе курсором миша преко контроле)
- *Visible* (да ли је контрола видљива или не, *true-false*)

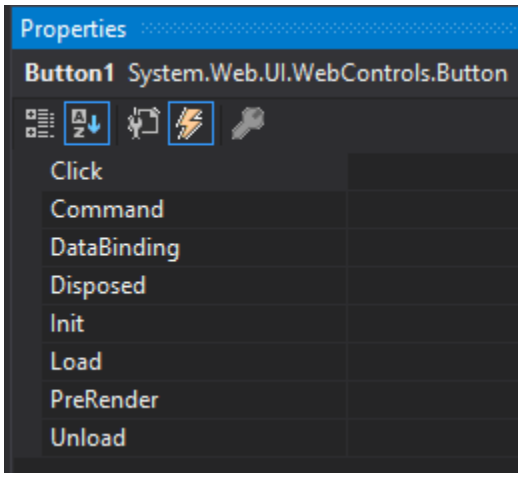
Сва својства једне контроле се у *VS* алату могу видети и кликом на ту контролу (у дизајн погледу) или када се селектује таг контроле (у *source* погледу) па у прозору *Properties* (слично као код класичних *Windows Forms* апликација).



5.5.2 Догађаји

Као што имају *master* странице, као што имају *web forme*, тако и контроле имају своје догађаје. Тип догађаја, као и њихов број за једну контролу зависи од врсте контроле. Тако нпр. за класу *Button* имамо догађај *Click* (када се кликне на контролу), за класу *TextBox* догађај *TextChanged* (када се промени текст контроле) или за класе *CheckBox* или *RadioButton* имамо догађај *CheckedChange* (када се чекира или дечекира контрола).

Догађаји се у *Visual Studio* алату могу креирати (или погледати) исто као код и својства само је потребно кликнути на малу иконицу у облику муње:



Битно је нагласити да се овде у листи налазе и догађаји стране, односно *web* форме на којој се контрола налази, па ту можемо видети и догађаје *Load*, *Init* или *PreRender* који су заправо догађаји *web* форме.

5.6 "Позивање" (приступање) контрола и других елемената

Када желимо да променимо вредност неке контроле, односно неко својство у оквиру програмског кôда (нпр. у оквиру догађаја *button click*) тој контроли можемо приступити преко њеног *ID*-а, нпр:

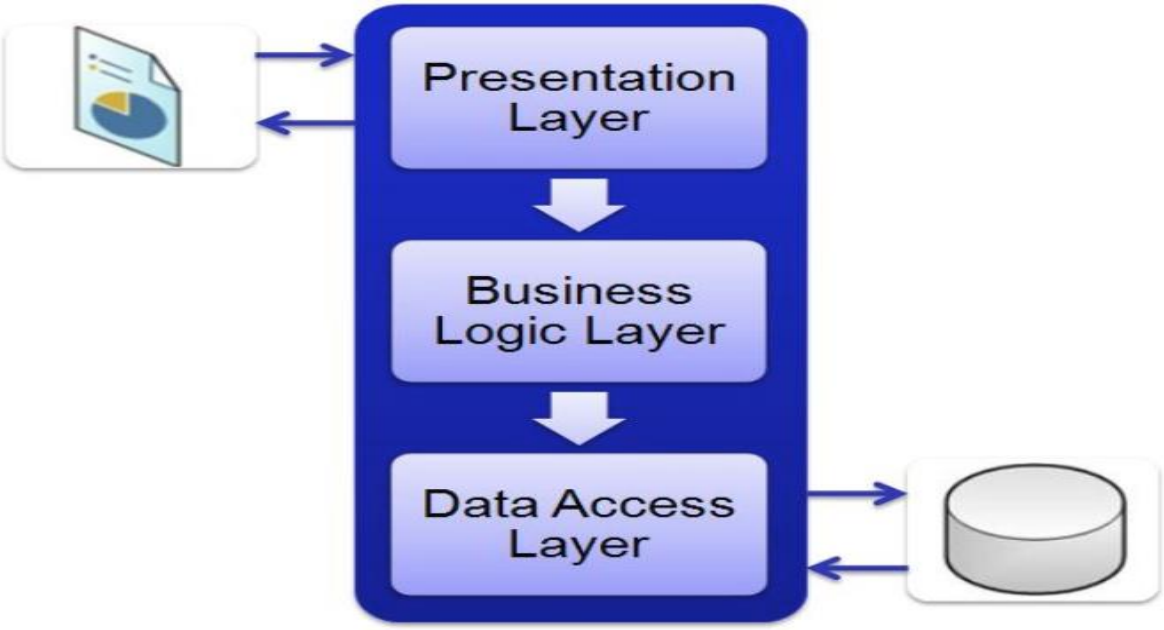
```
protected void Button1_Click(object sender, EventArgs e)
 {
     Label1.Text = "Kliknuo sam na dugme"; // Label1 je vrednost svojstva ID
 }
```

На исти начин се приступа било којој контроли која се налази на *web* форми.

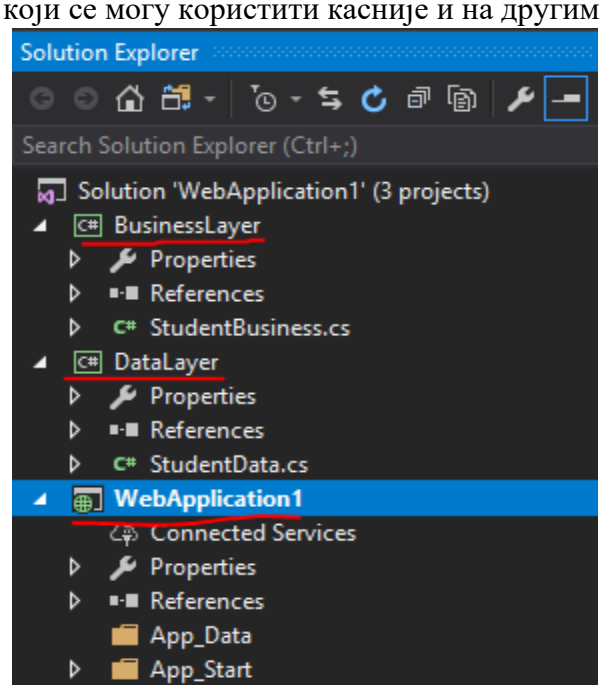
5.7 Слојевита архитектура

Када се развија било који софтверски производ, односно када се програмира, треба покушати пратити поједине стандардне који се свакодневно мењају што самим тим може бити доста тешко. Постоје бројни начини како се може дефинисатни архитетура једне апликације у *.NET* окружењу, али за апликације које су углавном *event-driven* као што су *Windows* и *Web Forms* (*ASP.NET*) најбоље је дефинисати архитектуру кроз два додатна слоја у оквиру *Solution*-а поред постојећег који се односи на сам *view* (*GUI*[1]). То су *Data* и *Business* слојеви (*DL* и *BL*). Ово се наравно односи на апликације које у позадини имају базу података или неки други извор података. Ово је класичан пример *Microsoft* тролојне архитектуре.

[1] *Graphic User Interface* (графички кориснички интерфејс)



Креирање два додатна слоја у *.NET* апликацијама, или у некој нашој већ креираној *Windows Forms* или *Web Forms* апликацији може се урадити тако што се додају још два пројекта у оквиру нашег *Solution*-а, један за *Data Layer* и други за *Business Layer*. Десним кликом на *Solution*, па на *Add* па *New Project*.. је акција која омогућава додавање још једног пројекта у нашем *Solution*-у. С обзиром да ће се у овом пројекту налазити само *C#* класе (нема неких *Windows* или *Web* форми нпр.) може се креирати тип пројекта: *Class Library (.NET Framework)*. Овакав пројекат осим тога што садржи само *C#* класе, заправо креира и *.dll* фајлове (након *build*-а) који се могу користити касније и на другим пројектима.



У оквиру *Data* слоја може се у једну класу заправо додати програмски кôд који смо креирали у прошлом поглаву (*Windows Forms*) и који се "качи" на базу података и враћа листу

студената из табеле *Student* (база *Faculty*). Овај слој заправо само ово и треба да ради, да се "качи" на базу података, да враћа, уноси, мења или брише податке из базе.
У оквиру *Business* слоја може се креирати једна класа у оквиру које ће се позвати инстанца класе из *Data* слоја и која ће кроз једну своју методу такође враћати исти резултат као што враћа она из *Data* слоја. Отприлике овако може изгледати ова класа:

```
public class StudentBusiness
{
    private StudentRepository studentRepository;

    public StudentBusiness() {
        studentRepository = new StudentRepository(); //kreiranje konekcije ka bazi
    }

    public List<Student> GetAllStudents()
    {
        return this. studentRepository.GetAllStudents(); // vraćanje liste studenata
    }
}
```

Због чега уопште постоји овај *Business* слој? Као што му сама реч каже, овде се може испрограмирати нека пословна логика. Нпр. у методи *GetAllStudents* можемо исфилтрирати листу студената да нпр. врати само оне којима име почиње са словом "N" или нешто слично томе (тад би можда имало смисла креирати неку другу методу, нпр. *GetStudentsByParam*). Наравно, могуће је ставити и неке параметре у ове методе на основу којих ће се вршити филтрирање или било каква друга *business* логика.
На крају, на *GUI* слоју ћемо заправо направити референцу ка *Business Layer*-у, и преко њега се повезивати на базу података и враћати податке. Овако може изгледати програмски кôд на некој *Web* форми где се купе подаци из базе података преко *Business* слоја а затим се на клик дугмета "пуни" *ListBox* контрола подацима:

```
private StudentBusiness businessStudent;

protected void Page_Load(object sender, EventArgs e)
{
    // na očitavanju stranice inicijalizacija business student klase
    this.businessStudent = new StudentBusiness();
}

protected void Button1_Click(object sender, EventArgs e)
{
    // na click dugmeta vraćanje liste studenata
    // i punjenje ListBox kontrole imenom studenata
    var students = this.businessStudent.GetAllStudents();
    foreach(Student s in students)
    {
        this.ListBox1.Items.Add(s.GetSetName);
    }
}
```