Spletna trgovina v ASP.NET in SQL Serverju z VWD

Andrej Arh

www.bodiprofi.si

munus 2

Uvajanje novih izobraževalnih programov v srednjem poklicnem in strokovnem izobraževanju s področja tehnike za obdobje 2008-2012.
Operacijo delno sofinancira Evropska unija iz Evropskega socialnega sklada in Ministrstvo za šolstvo in šport.

Splošne informacije gradiva

Izobraževalni program

Tehnik računalništva

Ime modula

Načrtovanje in razvoj spletnih aplikacij

Naslov učnih tem ali kompetenc, ki jih obravnava učno gradivo:

Povezovanje HTML, CSS, programiranja in podatkovne baze, priprava lastne predloge MasterPage in ustrezne podatkovne baze, prikaz podatkov na spletnem obrazcu, urejanje, brisanje in dodajanje novih zapisov v bazo preko spletnega obrazca, prenašanje podatkov med stranmi (Session), ustrezno obdelovanje podatkov z jezikom C#, skrb za varnost, določanje pravic posameznih uporabnikov ali uporabniških skupin, vgradnja predloge v lastno aplikacijo.

Povzetek

V gradivu je opisano: kako razviti in postaviti spletno trgovino, kako se zgradi administracijski del in uporabniški del (kupec), kako ustvariti potrebno bazo, kako izdelke prikazati na spletnem brskalniku, zgraditi košarico, naročilo, administracijo z izdelki, s spleta.

Ključne besede: Spletna trgovina, spletni obrazec, ASP.NET, CSS, HTML, C#, .NET, podatkovna baza, SQL, SqlDataSource, GridView, ListView, košarica, data class, FileUpload, varnost, objektno programiranje, Collections generic (List), Session, LINQ.

Avtor: Andrej Arh
Recenzent: Uroš Sterle
Lektor: Milena Ilič
Datum: Avgust 2012

Kazalo vsebine:

Kazalo slik:

1 Zahteve

1.1 Predznanje

Ogledali si bomo razvoj preproste spletne trgovine, za prodajo prenosnih računalnikov. Zaradi omejene obsežnosti literature bomo razvili preprosto verzijo, ki pa jo lahko razširimo v resnejšo aplikacijo. Za uspešno delo je potrebno sledeče predznanje:

- Poznavanje HTML,
- osnove oblikovanja s CSS,
- poznavanje in razumevanje podatkovnih baz in osnovno rokovanje s SQL (SELECT, UPDATE, INSERT, DELETE) stavki,
- objektno programiranje v programskem jeziku C#,
- osnovno poznavanje programskega orodja Visual Web Developer 2010 Express.

1.2 Priprava programske opreme

- Windows 7 Enterprise (Če je Home Edition, ne moremo inštalirati IIS strežnika),
- MS Visual Web Developer 2010 Express Edition (v nadaljevanju VWD),
- MS SQL Server 2008 Express,

Spletno trgovino bomo izdelali v tehnologiji ASP.NET, ki ga je razvilo podjetje Microsoft. Podatkovna baza bo SQL Server 2008 Express, ki je lahko implementirana v Visual Web Developer (v nadaljevanju VWD) ob inštalaciji, zato ga ni potrebno posebej naložiti.

Vso programsko opremo si lahko brezplačno prenesemo z Microsoftove spletne strani. Za VWD se moramo po 30 dneh še registrirati. Registracija je brezplačna.

2 Opis delovanja spletne trgovine

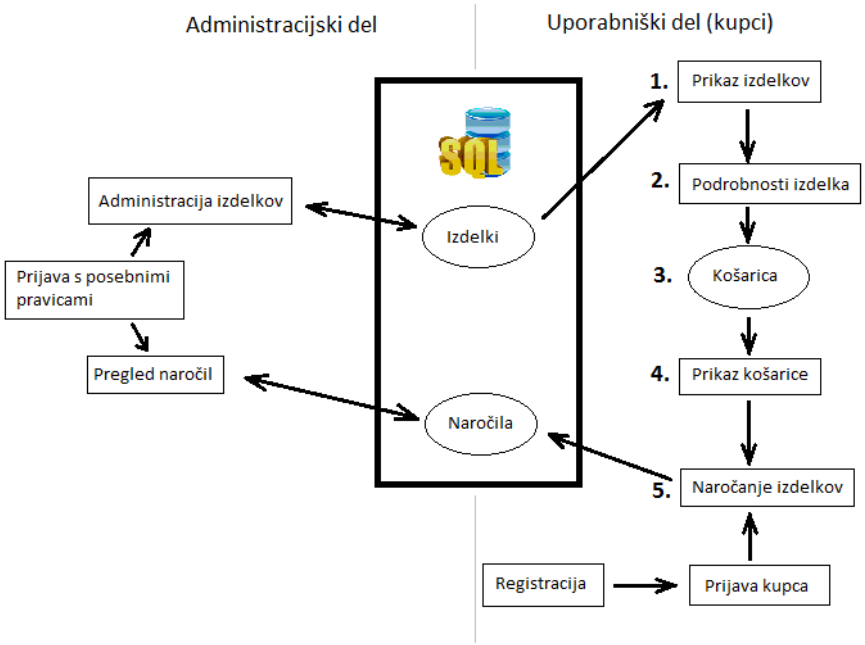
Pri razvoju spletne trgovine se v filozofijo prodajanja in psiholoških učinkih na kupca se ne bomo spuščali, ampak nas zanima predvsem tehnični del.

Na splošno bi lahko rekli, da je princip delovanja spletne trgovine podoben, kot v smo vajeni v fizični trgovini. Prodajalec polni police z izdelki, kupec pa jih jemlje s polic, daje v košarico in na koncu plača na blagajni.

Na spodnji sliki imamo prikazano blok shemo delovanja aplikacije, katero lahko razdelimo na dva dela: administracijski in uporabniški del.
V administracijski del spada vnos novih izdelkov, urejanje in brisanje izdelkov (ni priporočljivo) ter določanje aktivnosti samega produkta (ali se prikaže na uporabniški strani ali ne). V administracijski del spada tudi pregled naročenih izdelkov, ki so jih naročili kupci, ter delna možnost urejanja polja, ki označuje *že poslan / ni poslan*.

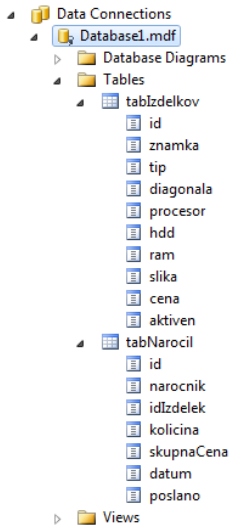
Uporabniški del (kupec) je sestavljen iz določenega zaporedja korakov.

1. Najprej iz baze iz tabele izdelkov prikažemo izdelke. Če kupec klikne na določen produkt,
2. se odpre stran, na kateri si lahko ogleda podrobnosti samega izdelka, katerega lahko doda v košarico, nato pa izbrane izdelke prikaže ali pa nadaljuje nakup.
3. V košarico se dodajajo izdelki, ki jih je uporabnik izbral iz prejšnjega koraka. Košarica je sestavljena iz določene podatkovne strukture. Osnovni atributi v košarici so izdelek, količina in cena.
4. Kupec lahko kadarkoli pogleda stanje košarice in iz nje lahko odstrani katerikoli izdelek. Če mu vsebina košarice ustreza, odda naročilo.
5. Ko kupec odda naročilo, mora biti v aplikaciji registriran in prijavljen z določenim uporabniškim računom. V tabelo naročila se shranijo podatki, ki so nujno potrebni za pošiljanje izdelkov kupcu: kupec, izdelek, količina, skupna cena in datum, nato pa se vsebina košarice izprazni. Ko je naročilo oddano, lahko na administracijski strani pregledajo izdelke in jih odpošljejo kupcu.



Slika 1: Shema delovanja spletne trgovine

Baza je srce aplikacije in mora biti načrtovana tako, da vanjo lahko shranimo primerne podatke o našem izdelku. Če bi npr. prodajali avtomobile, nas zagotovo zanimajo povsem drugi podatki, kot če bi prodajali računalnike. V našem projektu bomo prodajali računalnike, zato moramo pripraviti tudi ustrezno podatkovno strukturo izdelka.

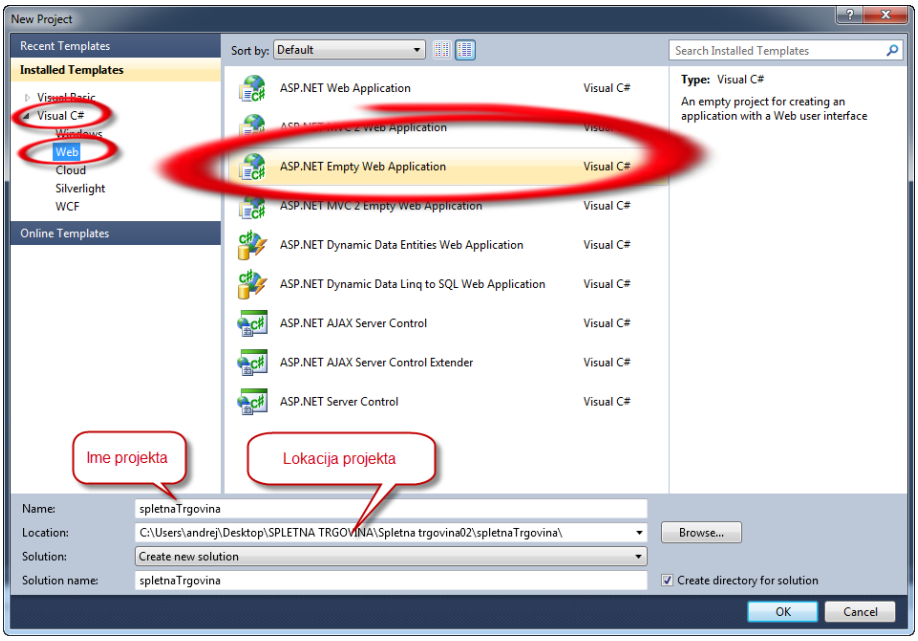


Slika 2: Podatkovna baza

3 Ogrodje

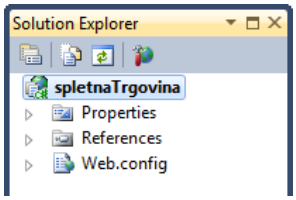
3.1 Nov projekt

Zaženimo programsko orodje VWD in ustvarimo nov projekt: **File/New project**.
Odpre se okno za ustvarjanje novega projekta. Paziti moramo, da na levi strani **izberemo jezik C#** in ne Visual Basic. Ko smo izbrali **Visual C#**, moramo na desni strani izbrati **ASP.NET Empty Web Application**. S tem smo izbrali čisto prazen projekt brez vseh datotek. V spodnjem delu bomo poimenovali naš projekt **spletnaTrgovina**, obenem pa določimo še lokacijo, kjer se bo nahajal naš projekt. To pot lahko spremenimo po želji. Kliknimo gumb **OK**.



Slika 3: Prvi projekt

Odpre se prazen projekt, v katerem so zaenkrat samo 3 datoteke. Na desni strani je Solution Explorer, v katerem so prikazane vse datoteke našega projekta.

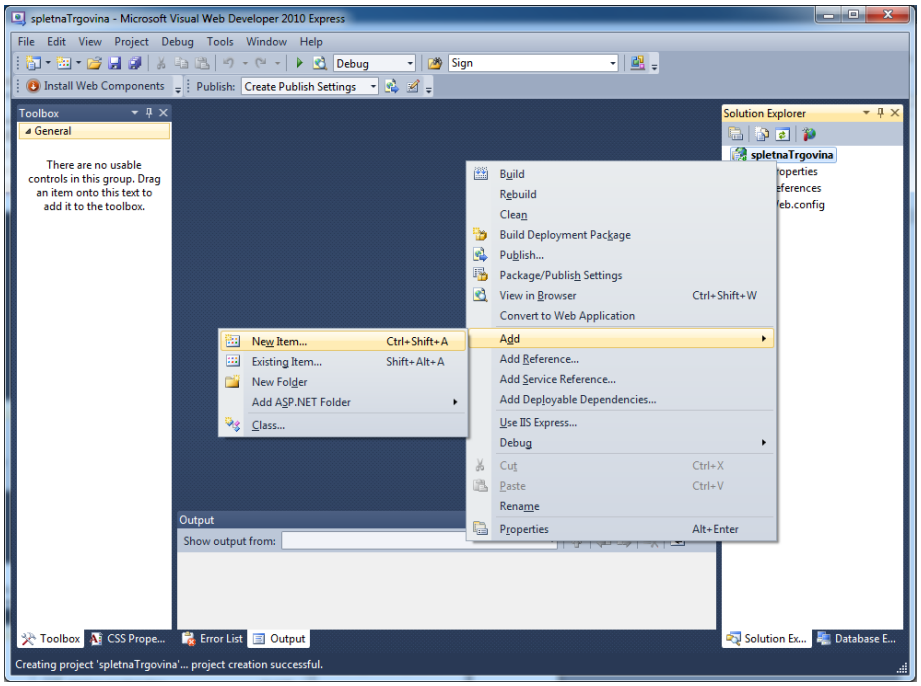


Slika 4: Solution Explorer

Če tega okna nimamo, ga lahko odpremo s kombinacijo tipk: **CTRL + ALT + L**
ali v menijski vrstici **View/Other Windows/Solution Explorer**.

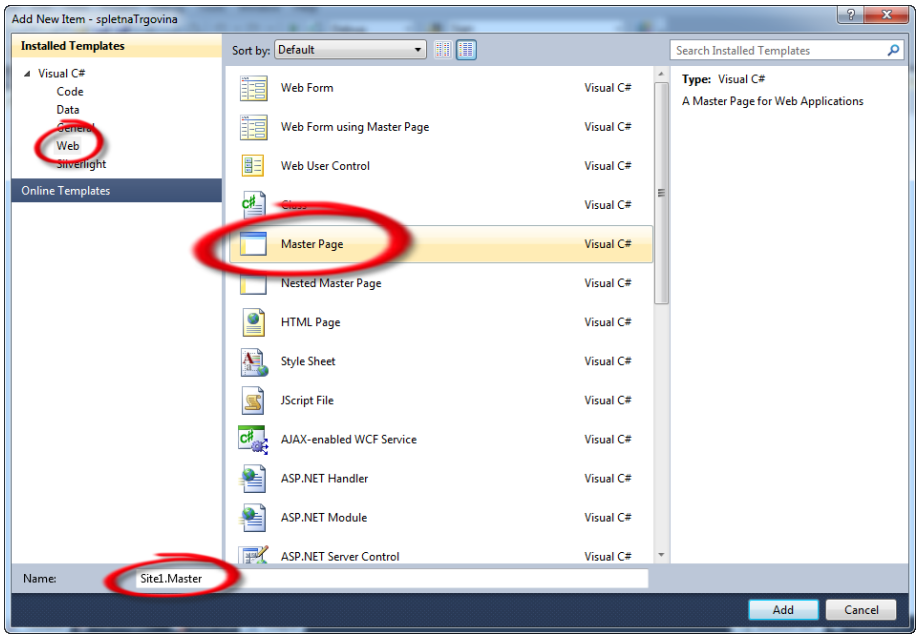
3.2 Predloga strani (Master Page)

Preden bomo kakorkoli začeli z našo aplikacijo, bomo najprej sestavili oz. zgradili predlogo (Template) naše aplikacije, ki jo bodo uporabljale vse podstrani (WebForms). Prednost te predloge je v tem, da jo lahko kadarkoli spremenimo, spremembe pa se bodo uveljavile na vseh podstraneh, ki uporabljajo to predlogo. S tem prihranimo veliko časa in nepotrebnega dela.
V naš projekt bomo dodali datoteko MasterPage. To naredimo tako, da v Solution Explorerju na ime projekta **spletnaTrgovina** desno kliknemo, nato kliknemo **Add/New Item …**



Slika 5: Nova datoteka

Odpre se okno, kjer lahko v naš projekt dodajamo datoteke različnega tipa.



Slika 6: Dodajanje datoteke MasterPage

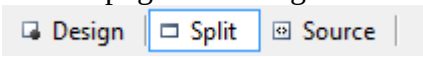
Prepričajmo se, da imamo na levi strani izbran **Visual C#**, zavihek **Web**. Na sredini okna pa izberemo **Master Page** ter pustimo privzeto ime **Site1.Master**. Kliknimo gumb **Add**.

V Solution Explorerju smo dobili novo datoteko Site1.Master, vsebina te datoteke pa je HTML koda, ki jo vidimo spodaj.

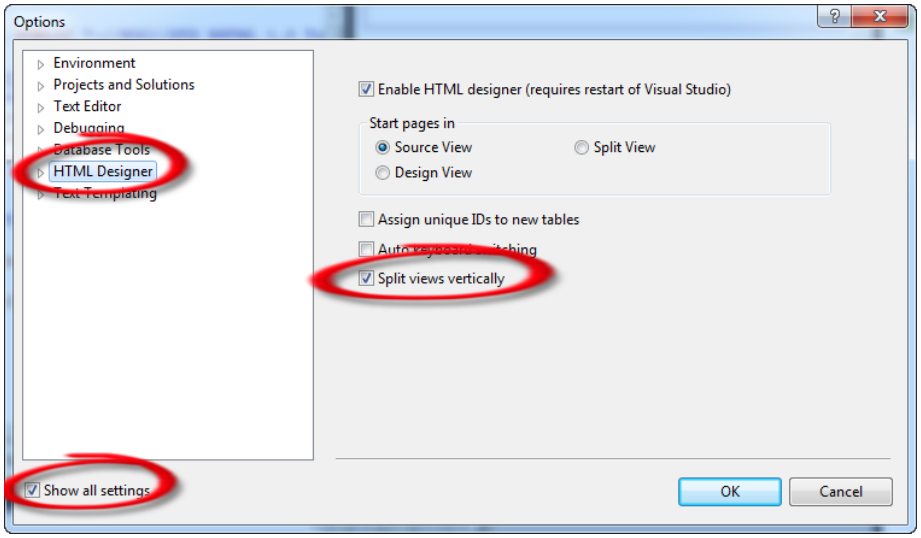
```
<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="Site1.master.cs"
Inherits="spletnaTrgovina.Site1" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <asp:ContentPlaceHolder ID="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">

        </asp:ContentPlaceHolder>
    </div>
    </form>
</body>
</html>
```

Pri tem opazimo, da so v njej vse že znane značke html, head, body, div in form. Form bo obrazec našega ogrodja, v njem pa je gradnik

```
<asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
</asp:ContentPlaceHolder>
```

Ta gradnik bo prikazoval vsebino vseh podstrani oz. obrazcev, ki bodo izhajali iz te predloge Site1.Master.
Poleg HTML kode lahko spremenimo pogled na design oz. razdelimo okno na Source/Design (kombinacija obeh pogledov). 

Če izberemo slednje, se okno horizontalno prepolovi na HTML kodo in izgled strani. Okno lahko razdelimo tudi vertikalno. V menijski vrstici izberimo **Tools/Options**.



Slika 7: Vertikalna razdelitev pogleda Design in Source

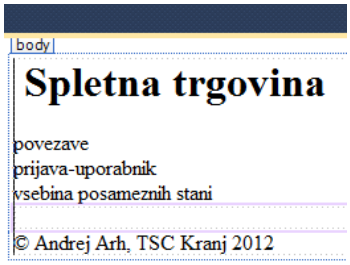
Preverimo, da je v levem spodnjem delu označena izbira **Show all settings**, nato pa v levem zgornjem oknu izberemo **HTML Designer**, na desni strani pa izberemo možnost **Split views vertically** in kliknemo OK.
Če želimo, da se te nastavitve uveljavijo, moramo VWD zapreti ter ponovno zagnati. Tako bomo imeli lahko HTML kodo in izgled strani v dveh vertikalno razdeljenih oknih, kar pripomore k lažjemu razvoju strani.

Ko smo še enkrat zagnali VWD, odprimo datoteko **Site1.Master** ter vsebino `<body>` zamenjajmo s spodnjo vsebino. Pri tem moramo paziti, da bodo besede **točno prepisane ali skopirane**. Pomembne so tudi velike začetnice.

```
<body>
     <form id="form1" runat="server">
     <div class="ogrodje">
        <div class="logo"><h1> Spletna trgovina</h1>
        </div>
        <div class="meni">
            <div class="meni2">
            povezave
            </div>
            <div class="login">
            prijava-uporabnik
            </div>
        </div>
        <div class="vsebina"> vsebina posameznih strani
                <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
```

```
                </asp:ContentPlaceHolder>
            </div>
            <div class="noga">
            &copy; Andrej Arh, TSC Kranj 2012
            </div>
        </div>
        </form>
</body>
```

Izgled zgornje HTML vsebine je prikazan na spodnji sliki.

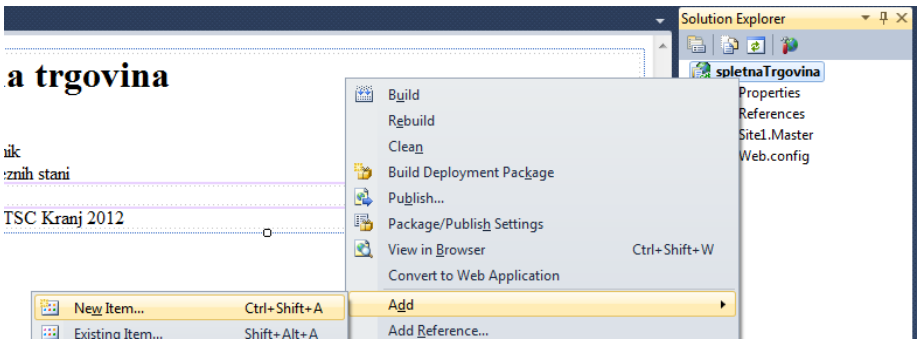


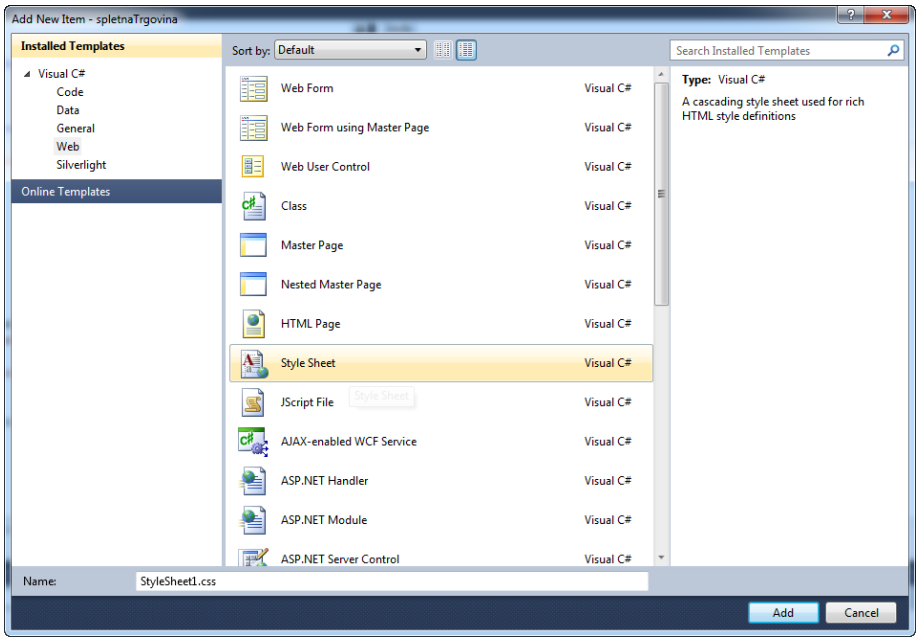
Slika 8: Izgled ogrodja brez CSS

3.3 Vključitev CSS v ogrodje

Na strani bomo uporabili preprost slog, ker oblikovanje sedaj ni pomembno, proti koncu literature pa bomo v projekt vgradili predlogo (Template), ki jo bomo dobili na spletu.

Dodajmo novo datoteko: desni klik na projekt v Solution Explorerju/**Add/ New Item**/, nato pa izberemo datoteko **Style Sheet**, ki je zaenkrat ne preimenujemo, zato naj bo poimenovana **StyleSheet1.css**.





Slika 9: Dodajanje datoteke Style Sheet

Z oblikovanjem strani se bomo ukvarjali kasneje, zato bomo sedaj uporabili preprosto obliko, ki je prilagojena tisti vsebini, ki smo jo napisali v Site1.Master.

Iz Solution Explorerja odprimo datoteko **StyleSheet1.css** in vanjo napišimo naslednjo vsebino:

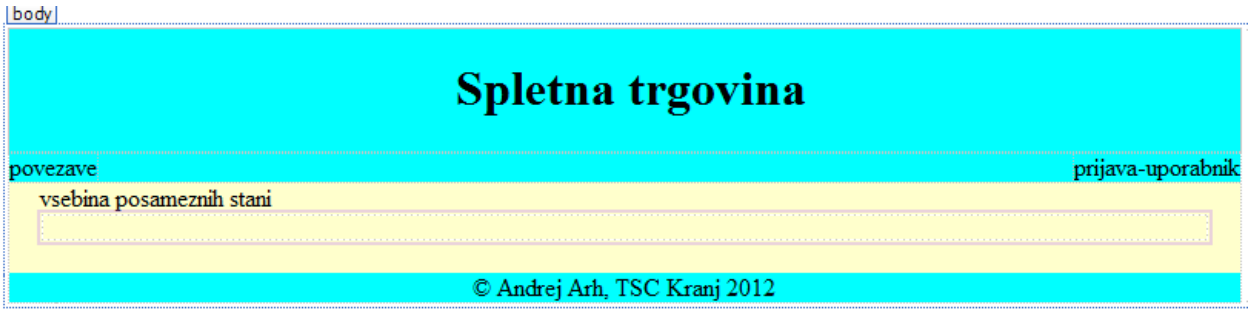
```
.ogrodje
{
    width: 800px;
}
.logo
{
    width:100%;
    height:80px;
    background-color:Aqua;
    text-align:center;
    vertical-align:middle;
}
.logo h1
{
    padding-top:20px;
}
.meni
{
    background-color: Aqua;
    float: left;
    width: 100%;
    text-align: left;
    padding: 0px;
}
.meni2
{
    float:left;
    text-align:left;
```

```
}
.login
{
    float:right;
    right:0px;
    text-align:right;
}
.vsebina
{
    background-color: #FFFFCC;
    padding:20px;
}
.noga
{
    background-color:Aqua;
    text-align:center;
}
```

Sedaj pa iz Solution Explorerja **z miško primimo datoteko StyleSheet1.css in jo z akcijo »drag & drop« izpustimo na naš Design pogled** ali pa v HTML v značko <head> napišemo naslednjo vrstico, ki vključuje css datoteko v html dokument.

```
<link href="StyleSheet1.css" rel="stylesheet" type="text/css" />
```

Ker so v HTML-ju poimenovani razredi odstavkov (`<div class="logo"> … </div>`)
tako, kot smo definirali v css datoteki, le-ta dobi obliko kot je prikazano na spodnji sliki, kasneje pa bomo v projekt implementirali predlogo, ki jo bomo povlekli s spleta.

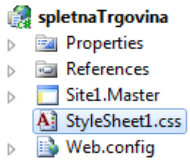


Slika 10: Izgled ogrodja s CSS

Z znanjem css oblikovanja lahko po želji predelamo zgornjo obliko.

Zgradili smo osnovno ogrodje naše spletne trgovine, posamezne strani pa se bodo prikazovale v gradniku ContentPlaceHolder. Nad njim na levi strani bodo povezave do posameznih podstrani, desno pa mesto za košarico, prijavo, odjavo in prikaz trenutnega uporabnika.

Če si pogledamo vsebino Solution Explorerja, bomo v njem opazili sledeče datoteke, kjer je:



Slika 11: Datoteke v projektu

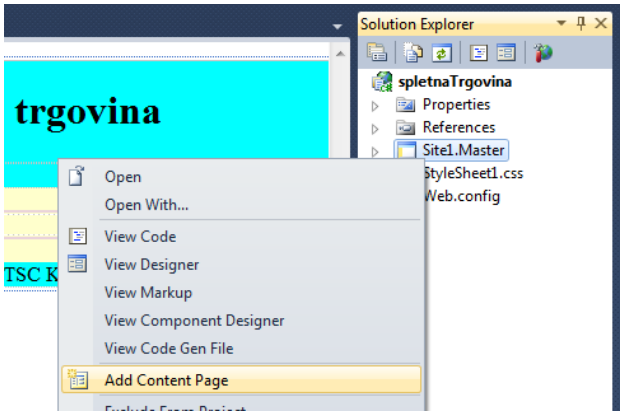
spletnaTrgovina – ime projekta,
Site1.Master – predloga (MasterPage), ki jo bodo v nadaljnje uporabljali vsi obrazci oz. podstrani,
StyleSheet1.css – datoteka, v kateri je zapisan slog oz. oblika strani,
Web.config – xml datoteka, v kateri so zapisani različni podatki o spletnem projektu.

4 Organizacija spletišča

V tem poglavju bomo organizirali spletišče. Posamezne podstani bomo postavili v različne mape ter postavili povezave do podstrani.

4.1 Spletni obrazci – podstrani

Sedaj pa dodajmo vse obrazce (WebForms), ki jih bomo uporabili v naši spletni trgovini. Če želimo, da bo obrazec uporabljal MasterPage predlogo, naredimo to na sledeč način. V Slolution Explorerju **desni klik na Site1.master/Add Content Page**.



Slika 12: Dodajanje novega obrazca WebForm - ContentPage

Vsebina obrazca WebForm1.aspx, ki smo ga ravno dodali, bo sedaj zapisana v gradniku ContentPlaceHolder1. Dovoljeno je spreminjanje samo te vsebine. Ostalo lahko spremenimo v Site1.Master.

Slika 13: Izgled prvega obrazca v brskalniku

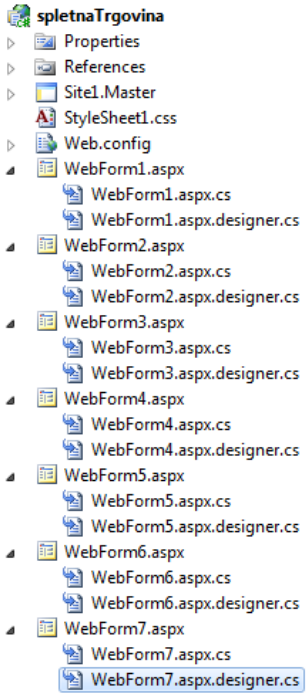
Obrazec ima sedaj obliko Site1.master, vsebino pa lahko pišemo samo v ContentPlaceHolder1. Zaenkrat napišimo samo besedilo **Pozdravljen svet**.

HTML koda tega dokumenta je sledeča:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site1.Master"
AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="spletnaTrgovina.WebForm1" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
runat="server">
    <p>
        Pozdravljen svet</p>
</asp:Content>
```

Pri tem gre omeniti nekatere lastnosti. `Language="C#"` pomeni, da bo kodiranje v ozadju z jezikom C#, `MasterPageFile="~/Site1.Master"` pomeni, da je za obliko izbran Site1.Master, in pa še `<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">` v tem gradniku bo vsebina posamezne strani.

Sedaj dodajmo še 6 obrazcev, kakršni so na spodnji sliki, na način **Add Content Page** tako kot prej.

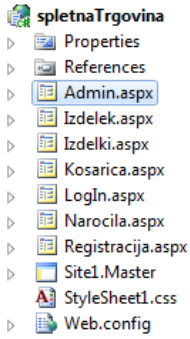


Slika 14: 7 spletnih obrazcev

Preimenujmo posamezne obrazce desni klik na **WebForm1.aspx/rename**. Preimenovanja naj bodo taka, kot je prikazano na spodnjem seznamu.

- WebForm1.aspx v Izdelki.aspx
- WebForm2.aspx v Izdelek.aspx

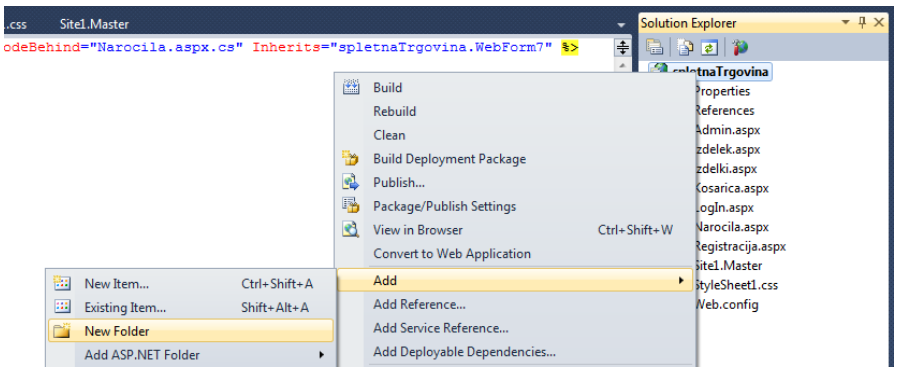
- WebForm3.aspx v Kosarica.aspx
- WebForm4.aspx v Admin.aspx
- WebForm5.aspx v LogIn.aspx
- WebForm6.aspx v Registracija.aspx
- WebForm7.aspx v Narocila.aspx



Slika 15: Poimenovanje obrazcev

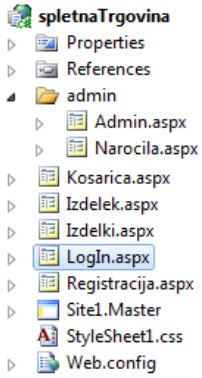
4.2 Mape

V naš projekt dodajmo novo mapo in jo poimenujmo **admin**, tako kot kaže spodnja slika.



Slika 16: Dodajanje nove mape v projekt

V mapo **admin** premaknimo datoteki **Admin.aspx** in **Narocila.aspx**, kot je prikazano na spodnji sliki. Datoteki smo premaknili mapo zato, da bomo kasneje lahko nastavljali različne pravice dostopa do datotek za posamezne skupine uporabnikov ali posameznega uporabnika.
Ko vse postavimo na svoje mesto, je končna struktura sledeča.



Slika 17: Struktura datotek v spletišču

4.3 Meni

Sedaj ko smo organizirali mape in datoteke, pa zgradimo še povezave do njih. Kakšen način povezovanja bomo uporabili, je izbira razvijalca. Lahko uporabimo način s povezavami čistega HTML `<a href="#">povezava</a>,` vendar tak način ni priporočljiv, še posebno v predlogi MasterPage. Če bomo to naredili na tak način, se povezava lahko izgubi, ko se bomo nahajali v kakšni drugi mapi ali podmapi, zato raje uporabimo gradnike, ki nam jih ponuja orodje VWD. V orodjarni ToolBox imamo na izbiro gradnik HyperLink ali pa pod zavihkom Navigation še Meni, TreeView in SiteMapPath.
SiteMapPath je sicer namenjen prikazovanju lokacije, v kateri mapi ali podmapi se nahajamo. Odločili smo se, da bomo uporabili gradnik **Menu** iz zavihka **Navigation**.

Iz Solution Explorerja odprimo datoteko **Site1.Master**.
Dodali bomo povezave v div, ki ima obliko razreda `class="meni2`, in prijavo uporabnika v div `class="login".`

```
<div class="meni">
    <div class="meni2">
    povezave
    </div>
    <div class="login">
    prijava-uporabnik
    </div>
</div>
```

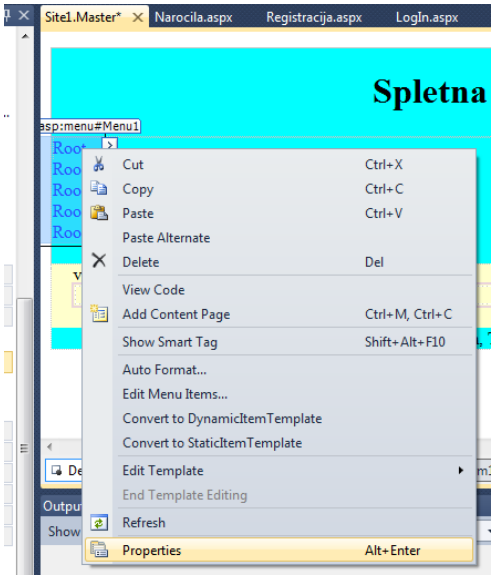
V **div.meni2** bomo izbrisali besedilo **povezave** in vanj dodali gradnik **Menu** ter enako v **div.login**. Pod naslovom sta sedaj dva **Menuja**. Levi bo za povezave na različne podstrani, desni pa za košarico.

Slika 18: Ogrodje z dodanima Menujema

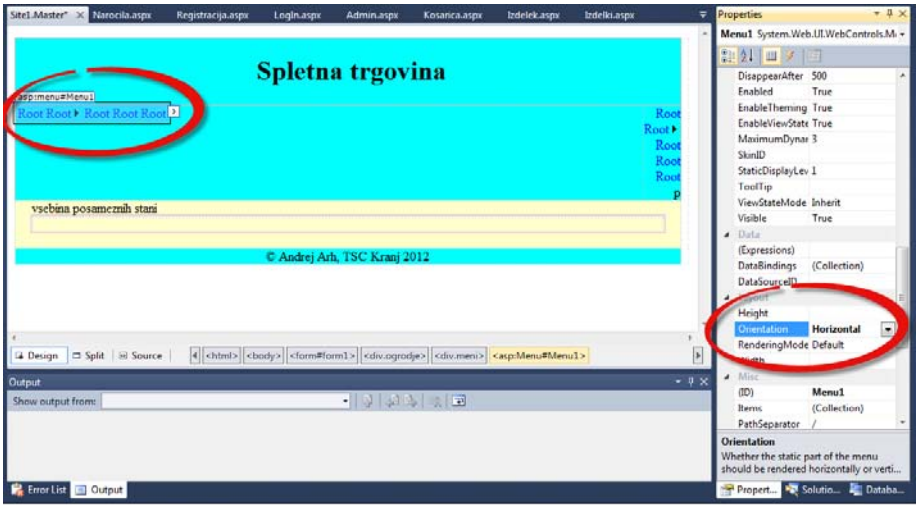
HTML koda Menujev:

```
<div class="meni">
    <div class="meni2">
        <asp:Menu ID="Menu1" runat="server">
        </asp:Menu>
    </div>
    <div class="login">
        <asp:Menu ID="Menu2" runat="server">
        </asp:Menu>
   </div>
</div>
```

Naš gradnik Menu ima povezave v vrsticah oz. vertikalno, kot vidimo na zgornji sliki. Želimo pa, da so povezave v eni vrstici (horizontalno) – ena zraven druge. Označimo levi meni ter **desni klik/properties**.

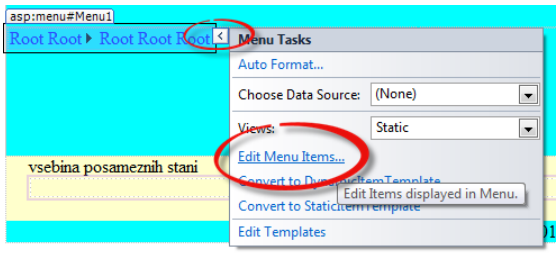


Če sedaj v desnem oknu Properties lastnost **Orientation** spremenimo **iz Vertical v Horizontal**, se bo naš meni prikazal v vrstici – ena povezava zraven druge.



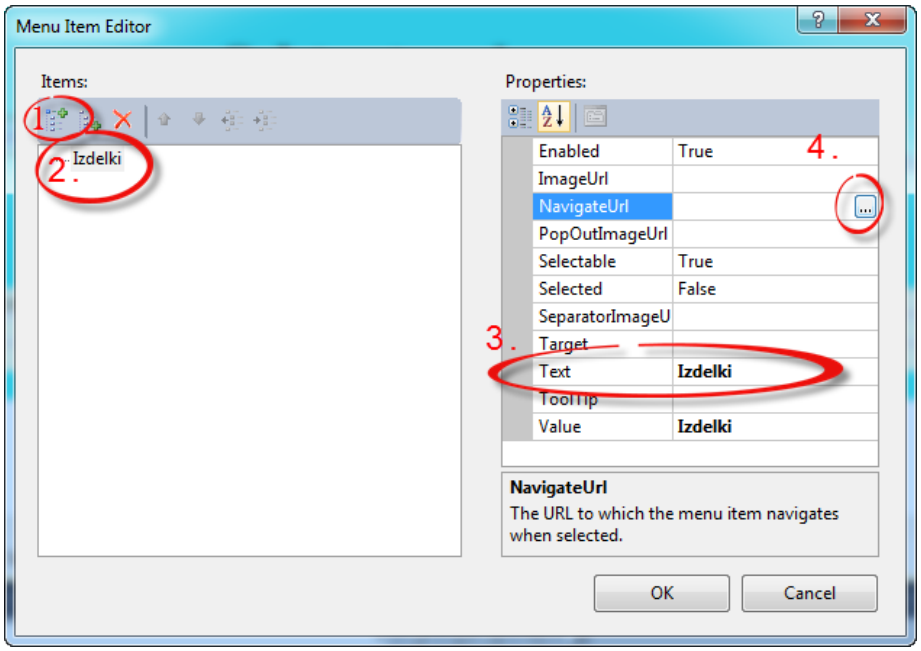
Slika 19: Nastavitev horizontalne orientacije Menuja

Sedaj pa uredimo Menu, da bomo imeli povezave med stranmi.
V pogledu Design označimo **Menu1** ter kliknimo na **zavihek Menu Task** ob desnem robu gradnika, nato pa kliknimo na povezavo **Edit Menu Items …**



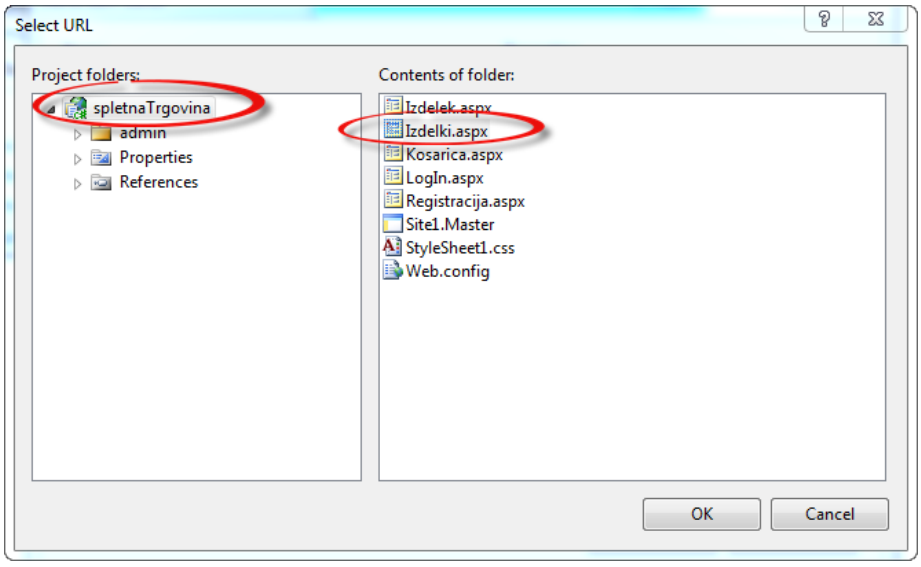
Slika 20: Menu Task

Odpre se okno za urejanje povezav v tem gradniku.



Slika 21: Nastavitev povezav v gradniku Menu

Na zgornji sliki pod točko 1 dodajamo povezave tega gradnika, pod točko 2 se prikaže posamezna povezava, ki jo moramo v oknu Properties še urediti. Na levi strani pod lastnost Text (točka 3) v katero vpišemo besedilo (vtipkajmo Izdelki) povezave, ki se bo prikazalo na spletni strani, ter pod točko 4 izberemo povezavo, na katero naj kaže. Ko odpremo to zadnjo točko, se nam pokaže novo okno, v katerem lahko izberemo datoteko, na katero želimo, da nas ob kliku preusmeri.

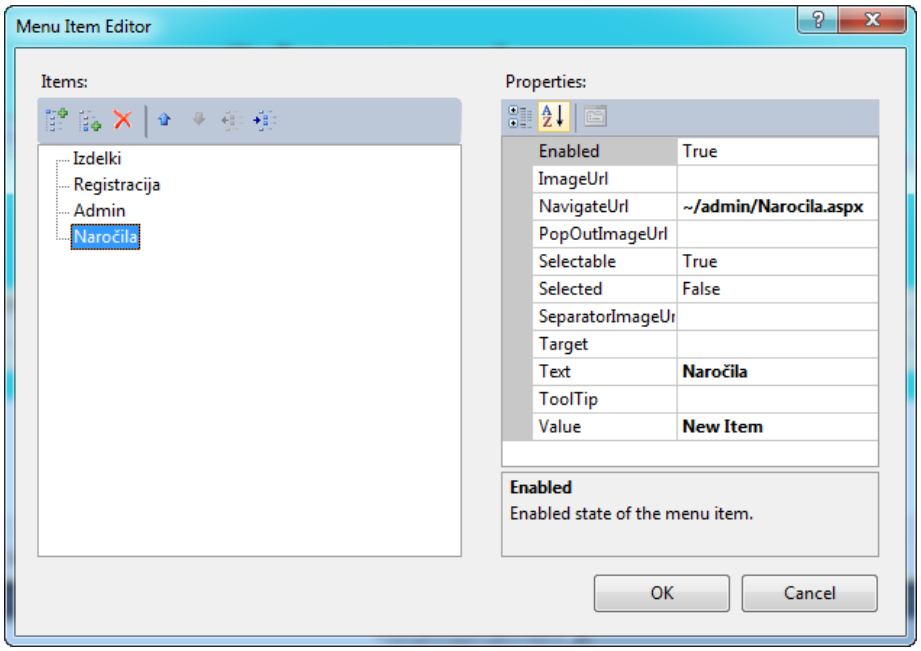


Slika 22: Določitev cilja povezavi

Na levi strani izberemo mapo, na desni strani pa je vsebina te mape. Ker datoteke Izdelki.aspx ni v nobeni podmapi, moramo na levi strani imeti označeno mapo spletnaTrgovina, na desni pa izberemo datoteko Izdelki.aspx, ter **OK**.
Sedaj pa dodajmo še tri povezave ter jim določimo ime in povezavo na stran.

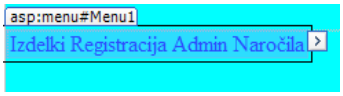
Povezavam smo določili cilje:

- Izdelki ~/Izdelki.aspx
- Registracija ~/Registracija.aspx
- Admin ~/admin/Admin.aspx
- Naročila ~/admin/Narocila.aspx



Slika 23: Konfigurirane povezave

Dobimo sledečo obliko menija:

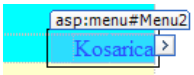


HTML koda Menuja:

```
<asp:Menu ID="Menu1" runat="server" Orientation="Horizontal">
    <Items>
        <asp:MenuItem NavigateUrl="~/Izdelki.aspx" Text="Izdelki"
             Value="Izdelki">
        </asp:MenuItem>
        <asp:MenuItem NavigateUrl="~/Registracija.aspx"
            Text="Registracija" Value="Registracija">
         </asp:MenuItem>
        <asp:MenuItem NavigateUrl="~/admin/Admin.aspx" Text="Admin"
            Value="Admin">
        </asp:MenuItem>
        <asp:MenuItem NavigateUrl="~/admin/Narocila.aspx"
            Text="Naročila"
           Value=" Naročila "></asp:MenuItem>
    </Items>
</asp:Menu>
```

Celoten meni je sestavljen iz **lastnosti** ter elementa **Items**, v katerem so gnezdene povezave **MenuItem**. Vsak element MenuItem ima lastnost **NavigateUrl**, ki hrani pot do datoteke, ki jo želimo odpreti, **Text** pa se bo prikazal na spletni strani.

Enako naredimo tudi za Menu2 ter ga povežimo z datoteko ~/Kosarica.aspx.

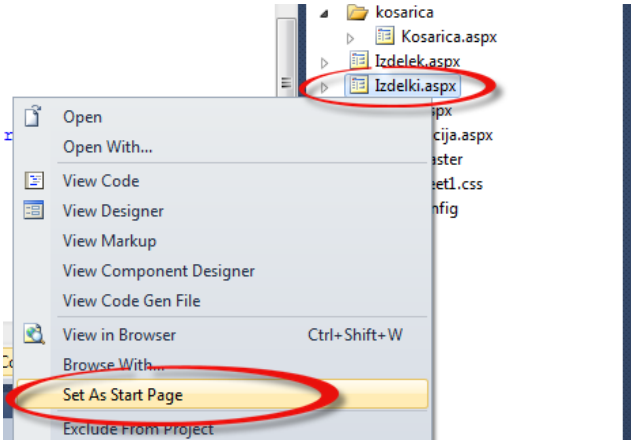


HTML:

```
<asp:Menu ID="Menu2" runat="server">
    <Items>
        <asp:MenuItem NavigateUrl="~/Kosarica.aspx"
                Text="Kosarica" Value="Kosarica"></asp:MenuItem>
    </Items>
</asp:Menu>
```

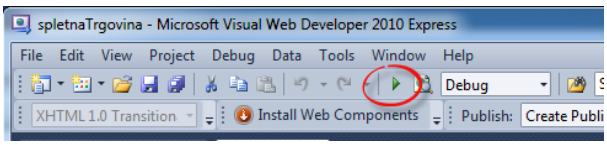
V tem primeru ima Menu2 samo eno povezavo, in sicer na stran, ki bo prikazovala vsebino košarice.

Poskrbimo še, da se bo ob zagonu projekta vedno kot prva stran odprla stran Izdelki.aspx. To pa zato, da se nam bo med razvojem projekta vedno odprla stran Izdelki.aspx. V Solution Explorerju desni klik na **Izdelki.aspx** in izberemo **Set as Start Page**.



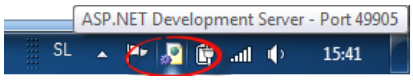
Slika 24: Nastavitev začetne strani

Poženimo projekt s klikom na **zeleno puščico** v orodni vrstici ali pa pritisnimo tipko **F5**.

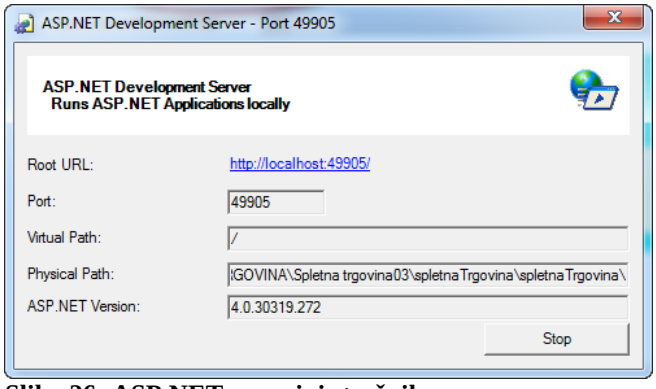


Slika 25: Zagon projekta (F5)

Ker ne bomo imeli čistih HTML strani, se za prikaz takih strani potrebuje spletni strežnik (v našem primeru IIS), vendar ima VWD že vgrajen razvojni strežnik, ki ga uporablja samo v času razvoja, kar nam precej olajša razvoj. Ko se stran prikaže v brskalniku, se v statusni vrstici pojavi ikona, označena na spodnji sliki.



Če 2x kliknemo nanjo, se nam prikaže status o razvojnem strežniku. V našem primeru je stran prikazana na localhost na portu 49905. Številka porta je naključna vrednost.

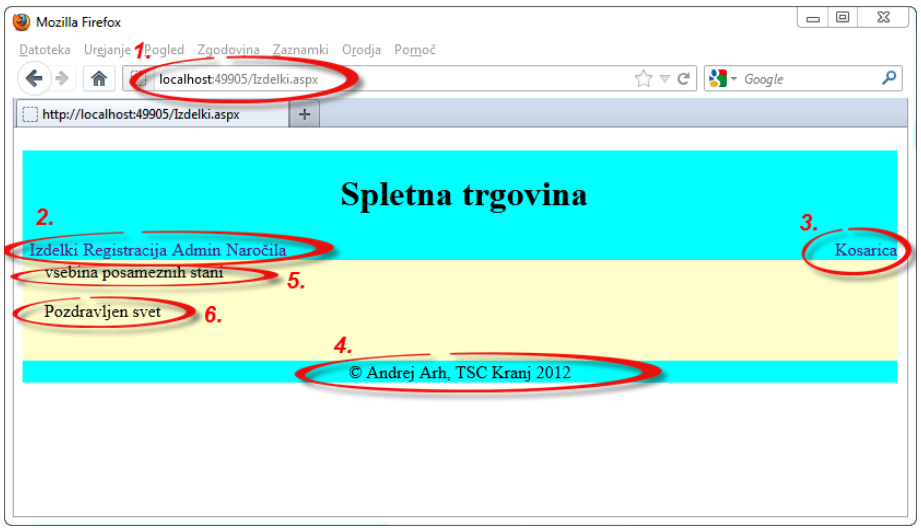


Slika 26: ASP.NET razvojni strežnik

Ti dve vrednosti sta tudi v URL vrstici našega brskalnika, le da je zraven dodano še ime datoteke, ki jo brskalnik trenutno prikazuje.

Prikazana stran je sestavljena iz več elementov, ki jih velja omeniti:

1. URL vrstica prikazuje **naslov:port/datoteka**. V našem primeru je naslov **localhost**, port je **49905**, datoteka, ki je odprta, pa je **Izdelki.aspx.**
2. Menijska vrstica, ki jo prikazuje Menu1, preko katere lahko dostopamo do posameznih strani v naši aplikaciji (V datoteki Site1.master – MasterPage).
3. Povezava do košarice.
4. Noga strani (v datoteki Site1.master – MasterPage).
5. Navaden tekst, ki je prikazan na vsaki strani (v datoteki Site1.master – MasterPage).
6. Vsebina posamezne strani ali datoteke. V našem primeru je ta tekst v datoteki Izdelki.aspx.



Slika 27: Razporeditev gradnikov po ogrodju

Iz datoteke Site1.Master izbrišite odvečno besedilo na točki 5 - **vsebina posamezne strani**. HTML koda datoteke Site1.Master:

```
<body>
     <form id="form1" runat="server">
     <div class="ogrodje">
        <div class="logo"><h1> Spletna trgovina</h1>
        </div>
        <div class="meni">
            <div class="meni2">
            <asp:Menu ID="Menu1" runat="server" Orientation="Horizontal">
                <Items>
                    <asp:MenuItem NavigateUrl="~/Izdelki.aspx" Text="Izdelki"
Value="Izdelki">
                    </asp:MenuItem>
                    <asp:MenuItem NavigateUrl="~/Registracija.aspx"
Text="Registracija"
                        Value="Registracija"></asp:MenuItem>
                    <asp:MenuItem NavigateUrl="~/admin/Admin.aspx" Text="Admin"
Value="Admin">
                    </asp:MenuItem>
                    <asp:MenuItem NavigateUrl="~/admin/Narocila.aspx"
Text="Naročila"
                        Value="New Item"></asp:MenuItem>
                </Items>
            </asp:Menu>
            </div>
            <div class="login">
                <asp:Menu ID="Menu2" runat="server">
                    <Items>
                        <asp:MenuItem NavigateUrl="~/Kosarica.aspx"
Text="Kosarica"
                            Value="Kosarica"></asp:MenuItem>
                    </Items>
                </asp:Menu>
            </div>
        </div>
        <div class="vsebina">
                <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">

                </asp:ContentPlaceHolder>
        </div>
        <div class="noga">
        &copy; Andrej Arh, TSC Kranj 2012
        </div>
    </div>
    </form>
</body>
```

5 Baza

Preden resno začnemo načrtovati našo aplikacijo, se moram najprej vprašati, kaj bomo v naši spletni trgovini prodajali. Ker nas pri prodaji računalnikov zanimajo povsem druge lastnosti kot pri avtomobilih ali kakšnem drugem izdelku, moramo za ta izdelek pripraviti tudi ustrezne tabele v bazi, da bomo vanjo lahko shranili tudi potrebne podatke o samem izdelku. Naš namen ni načrtovanje neke kompleksne podatkovne baze, ampak jo bomo zgradili ravno toliko, da bomo

lahko postavili preprosto spletno trgovino. V realnosti pa bi bilo potrebno v podatkovno bazo vložiti precej več znanja (relacije, procedure, triggerji, dogodki, zaščita itd.) SQL Serverja.

5.1 Tabele

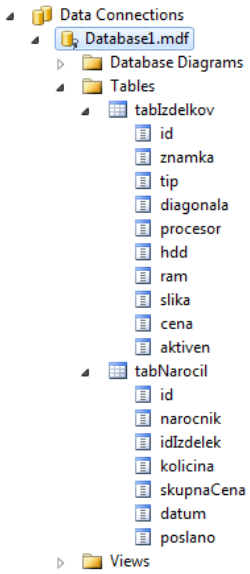
Preden bomo postavili prvo tabelo, moramo vedeti, katere lastnosti bodo zanimale kupca. Pri prenosnem računalniku bodo zagotovo igrali pomembno vlogo:

1. znamka
2. tip
3. diagonala zaslona [palci "]
4. hitrost procesorja [GHz]
5. velikost trdega diska [GB]
6. velikost rama [GB]
7. slika
8. cena [€]

Zraven pa bomo dodali še polje, ki bo identificiralo, če je izdelek še v prodaji (aktiven).

Za naročila kupcev pa bomo zopet ustvarili tabelo **tabNarocil**, v katero bomo zabeležili

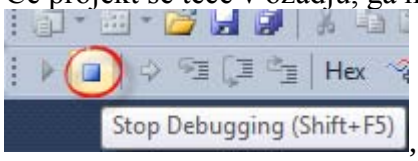
1. kdo je naročil
2. kateri izdelek
3. količino izdelkov
4. skupna cena teh izdelkov
5. datum in čas naročila
6. poslano / ni poslano. S tem poljem si bomo pomagali, da bo tisti, ki pošilja izdelke, videl samo ne poslane izdelke.



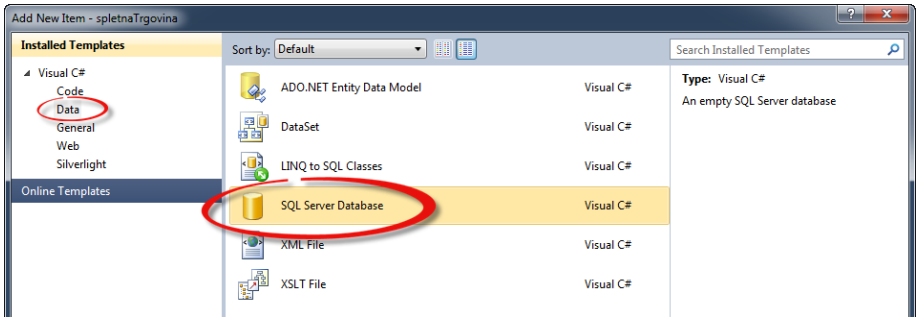
Slika 28: Struktura tabel v bazi

Naslednja naloga bo, da ustvarimo podatkovno bazo, v katero bomo shranili izdelke, ki jih bomo prodajali v naši spletni trgovini, in naročila, kjer se bodo hranili naročeni izdelki, ki jih bodo izbrali obiskovalci spletne trgovine.

Če projekt še teče v ozadju, ga moramo najprej ugasniti v orodni vrstici

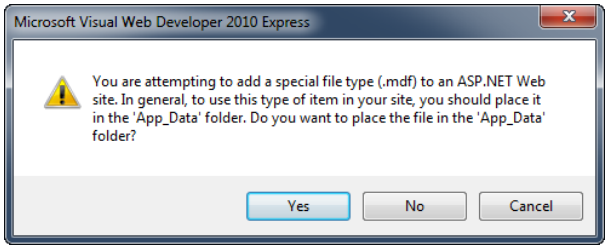


, da bomo lahko dodajali nove datoteke v projekt. V Solution Explorerju **desni klik na projekt/Add/New Item**. Odpre se okno, kjer bomo dodali podatkovno bazo. V zavihku **Data** izberemo **SQL Server Database**, ime pa pustimo privzeto – **Database1.mdf**, nato pa kliknimo gumb **Add**.



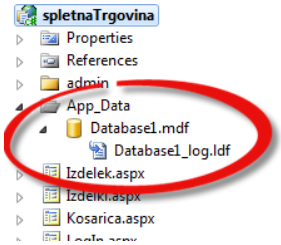
Slika 29: Dodajanje nove baze

Ko se nam pojavi okno, nas program vpraša, če postavi bazo v mapo **'App_Data'**, kjer so običajno shranjene datoteke s podatki. Kliknemo gumb **Yes**.



Slika 30: Obvestilo o kreiranju nove mape App_Data

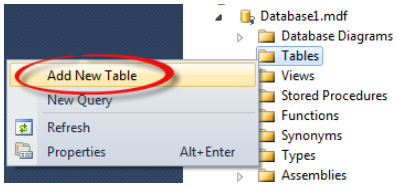
Če pogledamo v Solution Explorer opazimo, da se nam je v projekt dodala mapa, **App_Data** v njej pa je datoteka **Database1.mdf**, ki smo jo ravnokar ustvarili.



Slika 31: Baza v Solution Explorerju

Ker je baza Database1.mdf še prazna, moramo sedaj v njej ustvariti ustrezno podatkovno strukturo z ustreznimi tabelami, polji in podatkovnimi tipi, ki jih bomo potrebovali v naši spletni trgovini.
Dvokliknimo na datoteko **Database1.mdf**.

Odpre se nam zavihek Database Explorer, v katerem bomo definirali naše tabele podatkovne baze. Sedaj bomo dodali novo tabelo, v kateri bomo hranili podatke o izdelkih. **V Database Explorerju desni klik na Tables/Add New Table**.



Slika 32: Kreiranje tabel v bazi

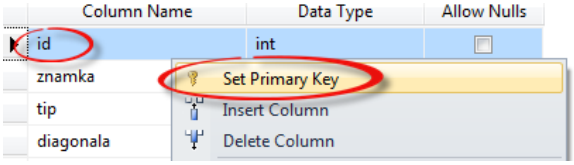
Odpre se nam grafični urejevalnik za definiranje tabele.
V posamezne vrstice bomo vpisali imena stolpcev v tabeli, zraven pa podatkovni tip.
Dodajmo nove stolpce z imeni in podatkovnimi tipi, kot kaže spodnja slika.

Column Name	Data Type	Allow Nulls
id	int	☐
znamka	varchar(50)	☐
tip	varchar(50)	☐
diagonala	float	☐
procesor	float	☐
hdd	float	☐
ram	float	☐
slika	varchar(200)	☑
cena	float	☐
aktiven	bit	☐

Slika 33: Definiranje tabele tabIzdelkov

5.1.1 Primarni ključ

Ker bomo vsakemu izdelku določili svojo številko id, bomo polju id določili primarni ključ. S tem smo zagotovili, da se to število v tabeli ne more ponoviti dvakrat oziroma vsak izdelek ima svojo šifro in nobena šifra ne more pripadati dvema izdelkoma.
Določimo primarni ključ polju id: **desni klik na polje id/Set Primary Key**.

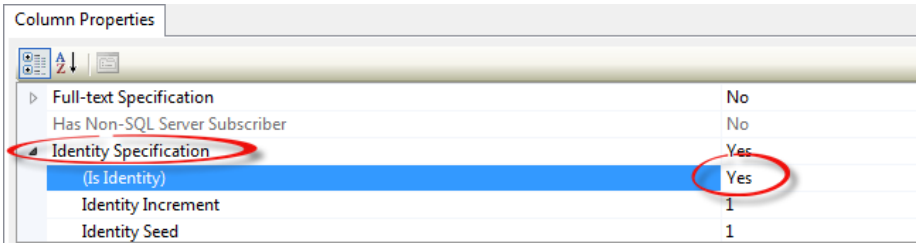


Slika 34: Nastavitev primarnega ključa

5.1.2 Samoštevilo (Autoincrement)

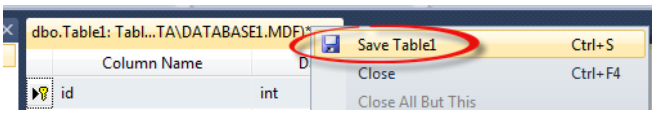
Samo povečevanje nam bo pomagalo, da se nam ne bo potrebno za vsak izdelek izmišljevati svoje kode, ampak baza sama poskrbi, da se vsak naslednji izdelek poveča za ena. To storimo tako, da označimo polje **id**, nato pa v zavihku **Column Properties** razširimo lastnost **Identity Specification** ter lastnost **(Is Identity)** spremenimo v **Yes**. Spodaj opazimo še lastnost Identity

Increment, kjer določimo, za koliko se bo povečalo število ob novem vnosu. V našem primeru naj ostane privzeto 1.



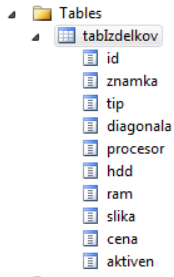
Slika 35: Nastavitev samoštevila (autoincrement)

Shranimo tabelo in jo poimenujmo **tabIzdelkov**. **Desni klik na zavihek tabele/Save Table1**.



Slika 36: Shranjevanje tabele

V Database Explorerju se sedaj prikaže ime tabele tabIzdelkov in njena struktura s pripadajočimi polji.



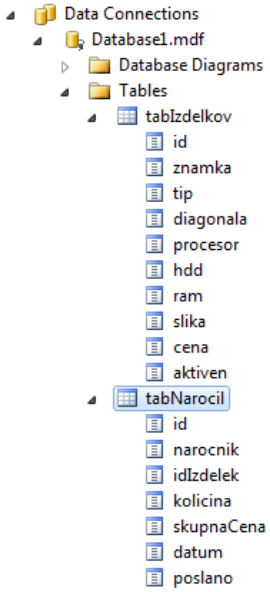
Slika 37: Tabela tabIzdelkov v Database Explorerju

Ustvarimo še tabelo za shranjevanje naročil: **desni klik na Tables/Add New Table** in naredimo sledečo podatkovno strukturo, kot je prikazano na sliki. Ne pozabimo na polje **id** zopet postaviti **primarni ključ** in nastaviti lastnost (**Is Identity** postavimo na **Yes**) tako kot v prejšnji tabeli in jo shranimo kot **tabNarocil**.

Column Name	Data Type	Allow Nulls
id	int	☐
narocnik	varchar(50)	☐
idIzdelek	int	☐
kolicina	float	☐
skupnaCena	float	☐
datum	datetime	☐
poslano	bit	☐

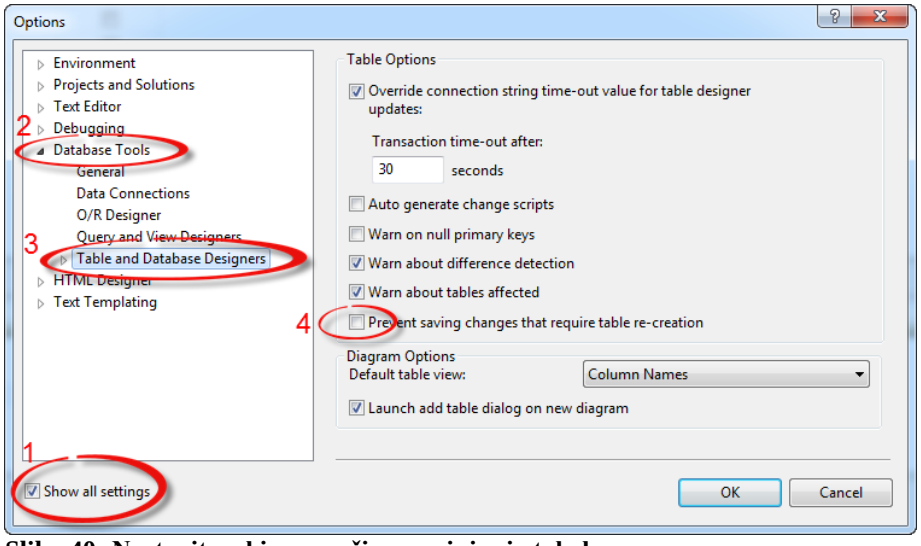
Slika 38: Definicija tabele tabNaročil

Imamo sledeči strukturi dveh tabel:



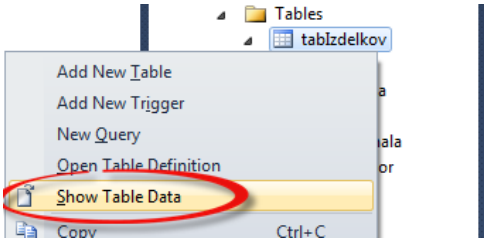
Slika 39: Tabeli tabIzdelkov in tabNarocil v Databse Explorerju

V primeru, da smo tabelo že shranili, pa bi radi v njej naredili kakšno spremembo, nam zaradi varnosti VWD tega ne dovoli. Če bi vseeno radi naredili kakšno spremembo, moramo v nastavitvah to omogočiti. V menijski vrstici kliknimo **Tools/Options/**. Odpre se okno z nastavitvami. Najprej moramo prikazati vse nastavitve (1), nato izberemo **Database Tools** (2), nato **Table and Database Designers** (3), potem pa moramo odstraniti kljukico iz nastavitve **Prevent saving changes that require table re-creation** (4). Če bomo to storili, bomo lahko spreminjali strukturo že shranjene tabele. OK.



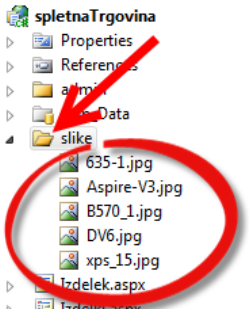
Slika 40: Nastavitev, ki omogoči spreminjanje tabel

Ker je naša tabela tabIzdelkov zaenkrat prazna, bomo vanjo ročno vpisali nekaj izdelkov. Desni klik v Database Explorerju, na **tabIzdelkov** kliknimo **Show Table Data**, nato pa se odpre tabela, v katero bomo ročno vpisali nekaj podatkov.



Slika 41: Prikaz vsebine tabele

S spleta naložimo slike 5 prenosnih računalnikov. Slikam dajmo kratka imena, kar bo olajšalo vpisovanje. V Solution Explorerju v projektu ustvarimo novo mapo **Add/New Folder** z imenom **slike** in vanjo z ukazom Copy-Paste prekopirajmo izbrane slike prenosnikov.



Slika 42: Slike izdelkov v Solution Explorerju

Imena slik bodo pomembna, ker bo v bazi v polju slika tudi to ime.
Vpišimo 5 podatkov, pri čemer enemu izdelku atribut **aktiven** dodelimo vrednost **False**. Tudi podatki bodo izmišljeni. Naj vas opozorim, da pri polju slika pazimo in zapišemo naslednjo obliko: **~/mapa/slika.končnica**. Pri tem pomeni ~ korenska mapa strežnika oz. relativna lokacija našega projekta, mapa/, v našem primeru se slike nahajajo v mapi **slike,** ter ime slike in njena ustrezna končnica. Kasneje bomo zgradili vnosni obrazec, s katerim bomo sliko izbrali z miško. Polja **id** ne vpisujemo, ker se zapisuje sam. Naj nas ne moti, če vmes kakšna številka manjka. To je posledica brisanja vrstic.

id	znamka	tip	diagonala	procesor	hdd	ram	slika	cena	aktiven
1	HP	635	17	1,2	200	2	~/slike/635-1.jpg	354	True
3	Acer	Aspire-V3	15	2,2	250	4	~/slike/Aspire-V3.jpg	150	True
4	Lenovo	B570	13	1	150	1	~/slike/B570_1.jpg	99,9	True
7	HP	DV6	17	2,1	520	4	~/slike/DV6.jpg	699	False
8	DELL	xps_15	17	1,8	250	2	~/slike/xps_15.jpg	450	True
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Slika 43: Ročni vnos podatkov v tabelo tabIzdelkov

6 Prikaz tabele iz baze na spletnem obrazcu

Ko smo uspešno vnesli 5 izdelkov v tabelo tabIzdelkov v bazi, moramo najti še način, da jih prikažemo na spletnem obrazcu.

Slika 44: Cilj prikazanih izdelkov

Odprimo datoteko **Izdelki.aspx**, ki jo najdemo v Solution Explorerju, ter izbrišimo besedilo **Pozdravljen svet**. Izgled HTML kode je sledeč:

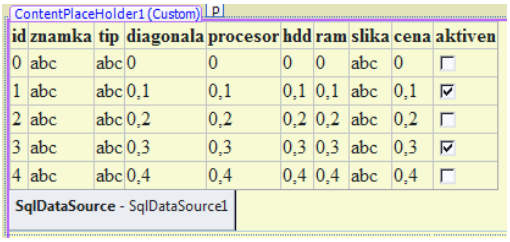
```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site1.Master"
AutoEventWireup="true" CodeBehind="Izdelki.aspx.cs"
Inherits="spletnaTrgovina.WebForm1" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
runat="server">
      TUKAJ MORA BITI VSEBINA SPLETNEGA OBRAZCA
</asp:Content>
```

Pri tem gre omeniti sledeče lastnosti:

1. `Title=""` – napis na zavihku v spletnem brskalniku
2. `Language="C#"` - določi se jezik, s katerim v ozadju lahko programiramo
3. `MasterPageFile="~/Site1.Master"` – obrazec privzame obliko iz datoteke Site1.Master
4. `CodeBehind="Izdelki.aspx.cs"` – programska koda (logika) se piše v datoteko Izdelki.aspx.cs
5. `<asp:Content ID="Content2"` - v tej znački je gnezdena in prikazana vsebina spletnega obrazca.
6. `ContentPlaceHolderID="ContentPlaceHolder1"` – vsebina spletnega obrazca se prikaže na tistem mestu, kjer je v Site1.Master gradnik z oznako `ContentPlaceHolderID= "ContentPlaceHolder1"`.
7. Torej imata Izdelki.aspx (tudi drugi obrazci) in Site1.Master isti ContentPlaceHolderID.

Preklopimo nazaj na Database Explorer in z miško kliknimo na tabelo **tabIzdelkov,** držimo z miško in jo izpustimo v obrazec **Izdelki.aspx**.

Kaj se je zgodilo? VWD nam je sam ustvaril nekaj gradnikov, ki so potrebni za prikaz podatkov iz baze. To sta gradnika GridView in SqlDataSource. SqlDataSource manipulira s podatki iz baze, GridView pa jih prikaže na spletnem obrazcu. Povezana sta tako, da ima gradnik GridView za lastnost DataSourceID izbran id gradnika SqlDataSource.



id	znamka	tip	diagonala	procesor	hdd	ram	slika	cena	aktiven
0	abc	abc	0	0	0	0	abc	0	☐
1	abc	abc	0,1	0,1	0,1	0,1	abc	0,1	☑
2	abc	abc	0,2	0,2	0,2	0,2	abc	0,2	☐
3	abc	abc	0,3	0,3	0,3	0,3	abc	0,3	☑
4	abc	abc	0,4	0,4	0,4	0,4	abc	0,4	☐

Slika 45: Gradnika Gridview in SqlDataSource

HTML datoteke Izdelki.aspx:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site1.Master"
AutoEventWireup="true" CodeBehind="Izdelki.aspx.cs"
Inherits="spletnaTrgovina.WebForm1" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
runat="server">
    <p>
        <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
            DataKeyNames="id" DataSourceID="SqlDataSource1"
            EmptyDataText="There are no data records to display.">
            <Columns>
                <asp:BoundField DataField="id" HeaderText="id" ReadOnly="True"
                    SortExpression="id" />
                <asp:BoundField DataField="znamka" HeaderText="znamka"
                    SortExpression="znamka" />
                <asp:BoundField DataField="tip" HeaderText="tip"
SortExpression="tip" />
                <asp:BoundField DataField="diagonala" HeaderText="diagonala"
                    SortExpression="diagonala" />
                <asp:BoundField DataField="procesor" HeaderText="procesor"
                    SortExpression="procesor" />
                <asp:BoundField DataField="hdd" HeaderText="hdd"
SortExpression="hdd" />
                <asp:BoundField DataField="ram" HeaderText="ram"
SortExpression="ram" />
                <asp:BoundField DataField="slika" HeaderText="slika"
SortExpression="slika" />
                <asp:BoundField DataField="cena" HeaderText="cena"
SortExpression="cena" />
                <asp:CheckBoxField DataField="aktiven" HeaderText="aktiven"
                    SortExpression="aktiven" />
            </Columns>
        </asp:GridView>
        <asp:SqlDataSource ID="SqlDataSource1" runat="server"
            ConnectionString="<%$ ConnectionStrings:Database1ConnectionString1 %>"
```

```
            DeleteCommand="DELETE FROM [tabIzdelkov] WHERE [id] = @id"
            InsertCommand="INSERT INTO [tabIzdelkov] ([znamka], [tip],
[diagonala], [procesor], [hdd], [ram], [slika], [cena], [aktiven]) VALUES
(@znamka, @tip, @diagonala, @procesor, @hdd, @ram, @slika, @cena, @aktiven)"
            ProviderName="<%$
ConnectionStrings:Database1ConnectionString1.ProviderName %>"
            SelectCommand="SELECT [id], [znamka], [tip], [diagonala], [procesor],
[hdd], [ram], [slika], [cena], [aktiven] FROM [tabIzdelkov]"
            UpdateCommand="UPDATE [tabIzdelkov] SET [znamka] = @znamka, [tip] =
@tip, [diagonala] = @diagonala, [procesor] = @procesor, [hdd] = @hdd, [ram] =
@ram, [slika] = @slika, [cena] = @cena, [aktiven] = @aktiven WHERE [id] = @id">
            <DeleteParameters>
                <asp:Parameter Name="id" Type="Int32" />
            </DeleteParameters>
            <InsertParameters>
                <asp:Parameter Name="znamka" Type="String" />
                <asp:Parameter Name="tip" Type="String" />
                <asp:Parameter Name="diagonala" Type="Double" />
                <asp:Parameter Name="procesor" Type="Double" />
                <asp:Parameter Name="hdd" Type="Double" />
                <asp:Parameter Name="ram" Type="Double" />
                <asp:Parameter Name="slika" Type="String" />
                <asp:Parameter Name="cena" Type="Double" />
                <asp:Parameter Name="aktiven" Type="Boolean" />
            </InsertParameters>
            <UpdateParameters>
                <asp:Parameter Name="znamka" Type="String" />
                <asp:Parameter Name="tip" Type="String" />
                <asp:Parameter Name="diagonala" Type="Double" />
                <asp:Parameter Name="procesor" Type="Double" />
                <asp:Parameter Name="hdd" Type="Double" />
                <asp:Parameter Name="ram" Type="Double" />
                <asp:Parameter Name="slika" Type="String" />
                <asp:Parameter Name="cena" Type="Double" />
                <asp:Parameter Name="aktiven" Type="Boolean" />
                <asp:Parameter Name="id" Type="Int32" />
            </UpdateParameters>
        </asp:SqlDataSource>
</p>
    <p>
         </p>
</asp:Content>
```

6.1 SqlDataSource

Gradnik SqlDataSource je nekakšna povezava z bazo in manipulira s podatki iz nje. Kratko povedano, izvaja SELECT, INSERT, UPDATE in DELETE ukaze nad tabelo v bazi. Če si podrobneje ogledamo HTML kodo tega gradnika, gre omeniti:

1. `ID="SqlDataSource1"` – ID gradnika.
2. `ConnectionString="<%$ ConnectionStrings:Database1ConnectionString1 %>"` – ConnectionString definira povezavo do baze, ki si jo lahko ogledamo v datoteki Web.config v Solution Explorerju. Del vsebine te datoteke, v kateri je definiran ConnectionString, je sledeč:

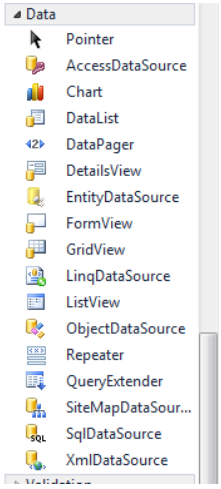
```
<connectionStrings>
  <add name="Database1ConnectionString1" connectionString="Data
Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\Database1.mdf;Integrated
Security=True;User Instance=True" providerName="System.Data.SqlClient" />
```

```
</connectionStrings>
```

3. `DeleteCommand, InsertCommand, SelectCommand in UpdateCommand.` Če si zgoraj v kodi podrobno ogledamo, kakšne so vrednosti teh lastnosti, ugotovimo, da so to osnovni SQL ukazi za brisanje, vstavljanje, prikazovanje in urejanje podatkov v tabeli v bazi.
4. `<DeleteParameters>, <InsertParameters>, <UpdateParameters> in <SelectParameters>` (zadnjega v tem primeru ni) gnezdijo parametre, prikazane v spodnji točki.
5. `<asp:Parameter Name="znamka" Type="String" />` - ta parameter bo hranil vrednost za polje **znamka** v bazi. V lastnosti Name je ime polja v bazi, Type pa je podatkovni tip v bazi. Le-ta je tipa **String**, kar je v SQL Serverju ekvivalentno tipu **varchar.** Vrednost parametra v npr. UpdateCommand se uporabi tako: `SET [znamka] = @znamka,` pri čemer je `@znamka` vrednost parametra.

6.2 Gradniki za prikaz podatkov

Če prikazujemo podatke na spletnem obrazcu, zagotovo potrebujemo vir podatkov (DataSource). Če si ogledamo spodnjo sliko, opazimo, da nimamo na izbiro samo SqlDataSource, ampak tudi druge vire, npr: AccessDataSource, če za podatkovno bazo izberemo Access, LinqDataSource, ObjectDataSource … Torej se lahko povežemo na različne podatkovne vire. Kako bomo podatke prikazali na spletnem obrazcu, pa je odvisno od tega, kaj želimo. Če naredimo to tako kot v zgornjem primeru (kliknimo na tabelo v Database Explorerju, jo z miško primemo in jo izpustimo na spletni obrazec), se ustvari ustrezni DataSource, podatki pa so prikazani v gradniku GridView. Tipično za ta gradnik je, da ima različne zapise prikazane v posameznih vrsticah, podatke pa razvrščene po kolonah.



Slika 46: Zavihek Data v orodjarni Toolbox

Za prikaz podatkov lahko uporabimo tudi druge gradnike. Iz orodjarne ToolBox iz zavihka Data lahko uporabimo DetailsView (prikaz samo enega izdelka naenkrat), FormView, DataList, ListView in druge; iz zavihka Standard pa lahko CheckBoxList, DropDownList, ListBox, RadioButtonList itd. Vsak ima svoje prednosti in slabosti ter svoj namen, zato običajno izberemo tistega, ki nam najbolj ustreza.

6.3 GridView

Gradnik GridView je namenjen prikazovanju podatkov. Lahko mu pripnemo različne podatkovne izvore, tako kot smo jih že omenili v prejšnji točki. Omogoča nam tudi označevanje, urejanje in brisanje podatkov v bazi, če ima SqlDataSource update in delete komande. Torej lahko tudi preko tega gradnika urejamo, brišemo in označujemo podatke.

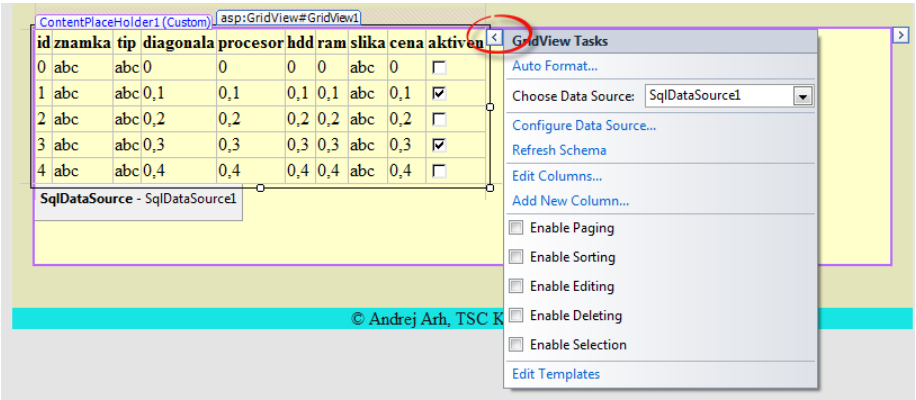
Če poženemo projekt (F5), opazimo, da se na spletnem obrazcu sedaj prikaže ravno taka vsebina, kot je zapisana v tabeli v bazi.

id	znamka	tip	diagonala	procesor	hdd	ram	slika	cena	aktiven
1	HP	635	17	1,2	200	2	~/slike/635-1.jpg	354	☑
3	Acer	Aspire-V3	15	2,2	250	4	~/slike/Aspire-V3.jpg	150	☑
4	Lenovo	B570	13	1	150	1	~/slike/B570_1.jpg	99,9	☑
7	HP	DV6	17	2,1	520	4	~/slike/DV6.jpg	699	☐
8	DELL	xps_15	17	1,8	250	2	~/slike/xps_15.jpg	450	☑

Slika 47: V brskalniku prikazan neoblikovan prikaz podatkov z GridView

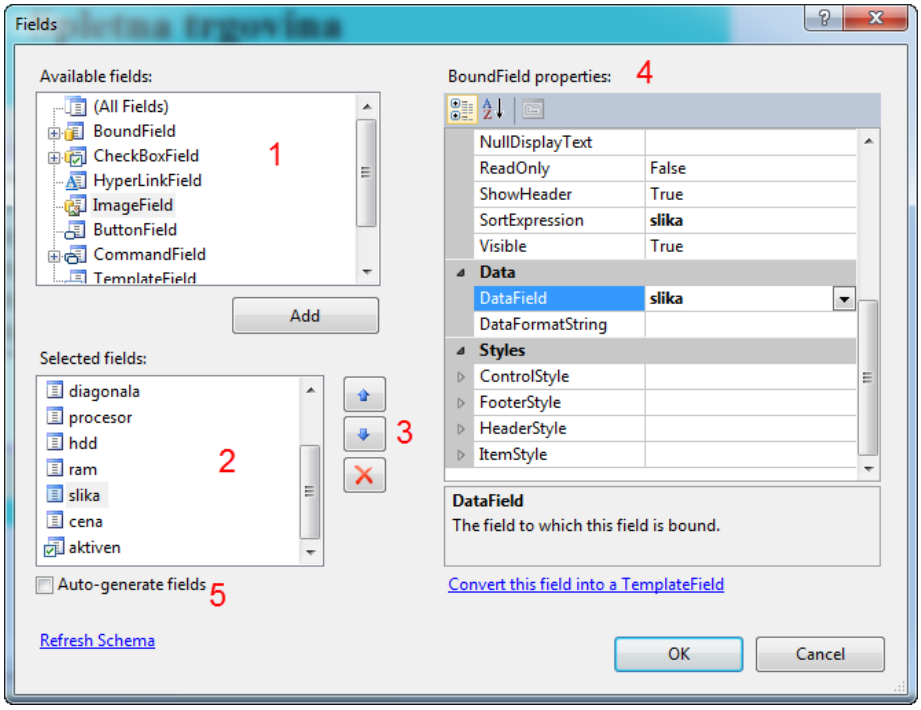
Zakaj se ne prikaže slika? Ker je v tabeli tabIzdelkov polje slika definirano kot besedilo, zato je tudi tu prikazal vneseno besedilo. To besedilo je pot do slike, zato lahko namesto polja z besedilom vstavimo polje s sliko.
Kliknimo na zavihek na gradniku GridView, kot kaže spodnja slika. Odpre se okno z različnimi nastavitvami.

1. **Auto Format …** Izbira je namenjena oblikovanju tega gradnika. Če izberemo katerokoli možnost zraven, opazimo, da se oblika gradnika GridView spreminja. Če nam to oblikovanje ni všeč, ga lahko tudi odstranimo z izbiro **Remove Formating.**
2. **Choose Data Source:** Tu iz spustnega seznama izberemo podatkovni vir. V našem primeru je to gradnik SqlDataSource, ki ima id=SqlDataSource1.
3. **Configure Data Source …** Ob tej izbiri posegamo v gradnik SqlDataSource ter ga s čarovnikom lahko tudi ustrezno spremenimo. Več o tem bomo povedali v nadaljevanju.
4. **Refresh Schema:** Ta možnost nam osveži gradnik GridView, kadar smo naredili kakšno spremembo v SqlDataSource, in prilagodi kolone le-temu.
5. **Edit Columns …** Urejamo kolone, ki so definirane v GridView (dodajamo, brišemo …).
6. Izbira **Enable Paging:** Če jo označimo, se pod gradnikom pojavi številčenje. V primeru, da imamo prikazanih zelo veliko vrstic, jih s tem enostavno omejimo, npr: 20 vrstic na stran.
7. **Enable Sorting:** Če jo označimo, lahko uporabnik sortira zapise po določeni koloni.
8. **Enable Editing:** V gradniku GridView se pojavi gumb z napisom Edit, s katerim lahko urejamo izbrani zapis. Če kliknemo Cancel, se sprememba razveljavi, če pa kliknemo Update, se spremembe zapišejo v bazo.
9. **Enable Deleting:** V gradniku GridView se pojavi še en gumb, s katerim lahko direktno iz baze izbrišemo zapis.
10. **Enable Selection:** Tudi tu se pojavi gumb z napisom Select, s katerim lahko sprožimo določeno akcijo, če kliknemo nanj. Npr. dodamo izdelek v košarico.



Slika 48: Zavihek GridView Task

Uredimo kolone, da bomo namesto besedila slike tudi dejansko videli sliko.
Tako kot na zgornji sliki, kliknimo na možnost **Edit Columns …**

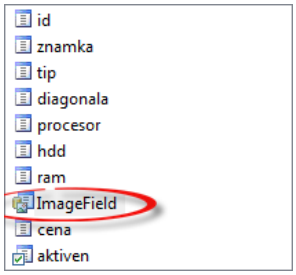


Slika 49: Nastavitev in oblikovanje gradnika GridView

Odpre se okno, v katerem lahko urejamo GridView. Naj razložimo posamezna okna:

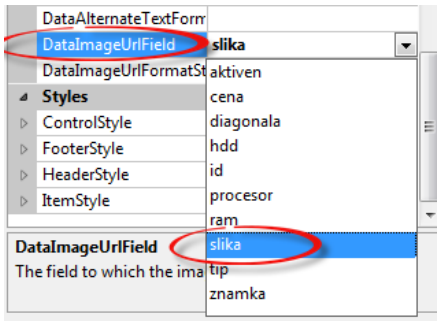
1. Polja oz. kolone, ki jih lahko dodamo v gradnik GridView.
2. Izbrane kolone v gradniku GridView.
3. Gumbi, s katerimi lahko določamo vrstni red med posameznimi kolonami in gumb za brisanje kolone.
4. Nastavitve izbrane kolone. Tu gre omeniti označeno polje DataField, v katerem določimo, kateri podatek iz baze se bo prikazoval v tej koloni (vir).
5. Auto-generate filed: ta opcija nam ponuja, da se vsa polja zgenerirajo avtomatsko glede na to, kakšna je struktura podatkov v datasource.

Sedaj iz okna Selected fields (točka 2) izbrišimo polje **slika** ter iz okna Available fields (točka 1) označimo **ImageField** ter kliknimo gumb **Add** in ga postavimo na mesto, kjer je bilo prej polje slika.



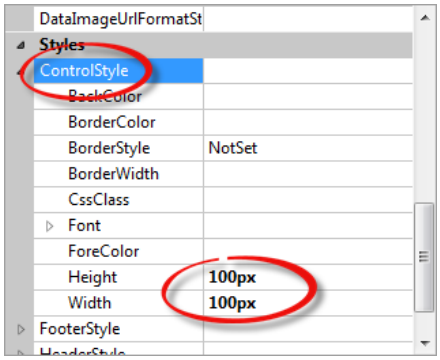
Slika 50: Polje ImageField

Če si pogledamo nastavitve tega polja, so malo drugačne kot v polju z besedilom, saj gre za sliko in ne besedilo. Poiščimo lastnost DataImageUrlField, ter ji določimo vir **slika**. To je ime izbranega polja v poizvedbi iz baze. Če imate slučajno spustni seznam prazen, **ročno vpišite slika**.



Slika 51: Vir polja ImageField (DataImageUrl)

Slika se bo prikazala v naravni velikosti, zato ji določimo še dimenzije: višina 100px in širina 100px.
Razširimo zavihek **ControlStyle** ter določimo lastnosti **Height** in **Width** na vrednost **100px**.



Slika 52: Nastavitev širine slike

Če sedaj poženemo projekt (F5), bomo dobili prikazane še slike.

id	znamka	tip	diagonala	procesor	hdd	ram		cena	aktiven
1	HP	635	17	1,2	200	2		354	☑
3	Acer	Aspire-V3	15	2,2	250	4		150	☑
4	Lenovo	B570	13	1	150	1		99,9	☑
7	HP	DV6	17	2,1	520	4		699	☐
8	DELL	xps_15	17	1,8	250	2		450	☑

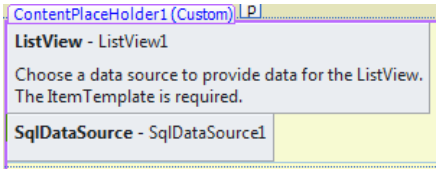
Slika 53: Prikaz podatkov s slikami v GridView

HTML koda gradnika GridView:

```
        <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
            DataKeyNames="id" DataSourceID="SqlDataSource1"
            EmptyDataText="There are no data records to display.">
            <Columns>
                <asp:BoundField DataField="id" HeaderText="id" ReadOnly="True"
                    SortExpression="id" />
                <asp:BoundField DataField="znamka" HeaderText="znamka"
                    SortExpression="znamka" />
                <asp:BoundField DataField="tip" HeaderText="tip"
SortExpression="tip" />
                <asp:BoundField DataField="diagonala" HeaderText="diagonala"
                    SortExpression="diagonala" />
                <asp:BoundField DataField="procesor" HeaderText="procesor"
                    SortExpression="procesor" />
                <asp:BoundField DataField="hdd" HeaderText="hdd"
SortExpression="hdd" />
                <asp:BoundField DataField="ram" HeaderText="ram"
SortExpression="ram" />
                <asp:ImageField DataImageUrlField="slika">
                    <ControlStyle Height="100px" Width="100px" />
                </asp:ImageField>
                <asp:BoundField DataField="cena" HeaderText="cena"
SortExpression="cena" />
                <asp:CheckBoxField DataField="aktiven" HeaderText="aktiven"
                    SortExpression="aktiven" />
            </Columns>
        </asp:GridView>
```

6.4 ListView

Gradnik ListView je precej podoben gradniku GridView, le da ta bazira na principu template oz. predloge za prikaz, označevanje, urejanje in vstavljanje novih podatkov. V principu zahteva malo več dela, vendar pa je precej bolj prilagodljiv kot GridView. Poleg tega zapise iz baze lahko prikažemo tudi drugače.
Pri našem prikazu izdelkov ne bomo prikazovali vseh podatkov o izdelku, ampak samo najpomembnejše: znamko, model, sliko in ceno. Če želi obiskovalec strani izvedeti več o izdelku, si le-to lahko ogleda na drugi strani ob kliku na podrobnosti.
Odprimo datoteko **Izdelki.aspx** in izbrišimo gradnika GridView in SqlDataSource.
V orodjarni ToolBox pod zavihkom **Data** izberimo gradnik ListView, ki ga z miško dvokliknemo ali pa kliknemo in spustimo na spletni obrazec, nato pa isto naredimo še z gradnikom SqlDataSource.

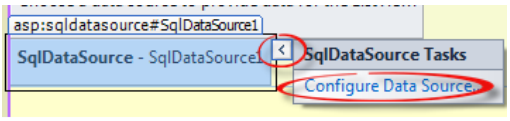


Slika 54: Nekonfigurirana gradnika ListView in SqlDataSource

HTML nekonfiguriranih gradnikov ListView in SqlDataSource:

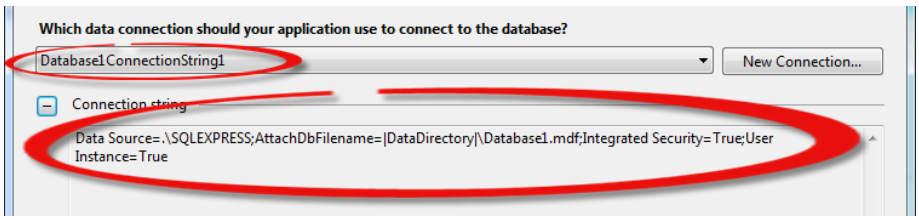
```
<asp:ListView ID="ListView1" runat="server">
        </asp:ListView>
        <asp:SqlDataSource ID="SqlDataSource1" runat="server"></asp:SqlDataSource>
```

Oba gradnika imata definiran samo ID. Zato moramo najprej definirati SqlDataSource, nato pa ga povezati z gradnikom ListView.
Kliknimo na zavihek na **SqlDataSource Task**, tako, kot je prikazano na spodnji sliki, in izberimo možnost **Configure Data Source …**



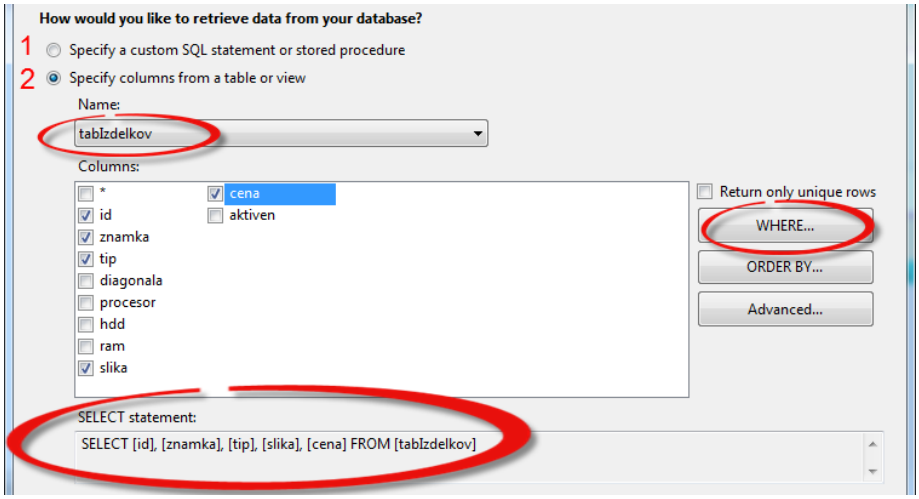
Slika 55: SqlDataSource Task

Odpre se čarovnik za konfiguriranje vira SqlDataSource. Najprej moramo izbrati ConnectionString. Ker imamo samo eno bazo, imamo trenutno tudi samo eno izbiro. Lahko si jo ogledamo tudi v datoteki Web.config. Kliknimo **Next!**



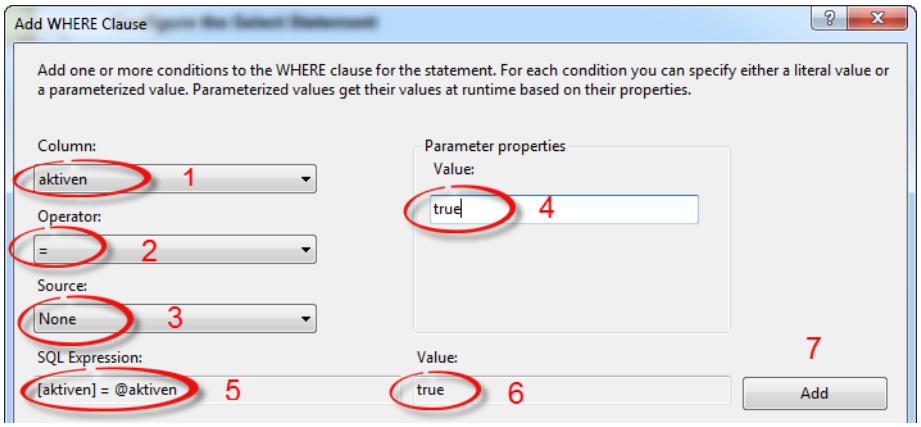
Slika 56: SqlDataSource in ConnectionString

V naslednjem koraku definiramo poizvedbo iz baze. Če želimo kakšno bolj zapleteno poizvedbo z več tabelami ali shranjeno proceduro, izberemo možnost (1). Če imamo preprosto poizvedbo, kot je naša, bomo izbrali možnost (2).
Spodaj pod Name izberemo tabelo v bazi, iz katere bi radi podatke. **Izberimo tabIzdelkov**. V oknu Columns se prikažejo vsa polja, ki so v tej tabeli. **Označimo samo tista polja, kot so označena na spodnji sliki**. Ko bomo izbirali polja, bomo na dnu pod SELECT statment videli, kako se generira SQL izraz za poizvedbo. Ko smo izbrali polja kot na spodnji sliki, na desni strani kliknimo gumb WHERE.



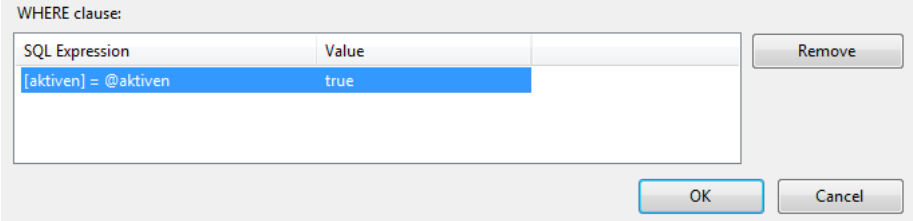
Slika 57: SqlDataSource in izbira tabele in polj tabele

Na tem koraku bomo naredili pogoj zato, da bomo na spletnem obrazcu videli samo aktivne izdelke. Enostavno povedano: samo tiste, ki so na zalogi. V točki (1) izberemo polje, nad katerim želimo izvesti preverjanje. Izberimo polje **aktiven**. V točki (2) izberemo operator za preverjanje, označen naj bo **enakost** =. V točki (3) za Source izberemo **None**, nato pa v točki (4) določimo privzeto vrednost parametra – v primeru, da parametra ne bomo spreminjali, bo imel vrednost **true**. V točki (5) vidimo SQL izraz WHERE pogoja, v točki (6) pa privzeto vrednost parametra. Kliknimo na gumb Add.



Slika 58: SqlDataSource in nastavitev pogoja WHERE

Ko smo kliknili na gumb Add, sta se v oknu **WHERE clause** pojavila parameter in njegova privzeta vrednost. Po potrebi lahko ustvarimo več parametrov. Kliknimo gumb OK.

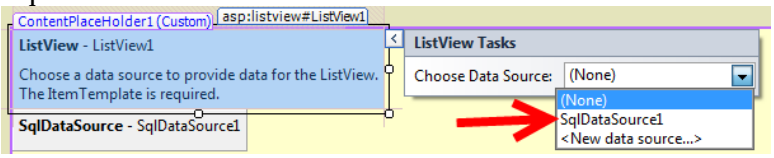


Slika 59: SqlDataSource in SelectParameter

V oknu SQL statment sedaj dobimo dopolnjen SQL izraz. **SELECT [id], [znamka], [tip], [slika], [cena] FROM [tabIzdelkov] WHERE ([aktiven] = @aktiven).** SQL izrazu smo dodali dodaten pogoj s parametrom **@aktiven.** Kliknimo **Next**. Prišli smo še do zadnjega koraka **Test Query**, kjer lahko ob kliku na gumb **Test Query** preverimo, ali smo pravilno zapisali poizvedbo. Kliknimo gumb Finish.
HTML koda gradnika SqlDataSource:

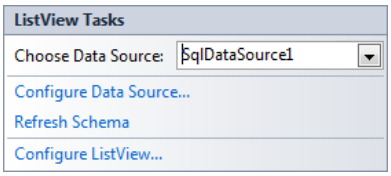
```
  <asp:SqlDataSource ID="SqlDataSource1" runat="server"
      ConnectionString="<%$ ConnectionStrings:Database1ConnectionString1 %>"
      SelectCommand="SELECT [id], [znamka], [tip], [slika], [cena] FROM
[tabIzdelkov] WHERE ([aktiven] = @aktiven)">
      <SelectParameters>
         <asp:Parameter DefaultValue="true" Name="aktiven" Type="Boolean" />
      </SelectParameters>
  </asp:SqlDataSource>
```

S čarovnikom smo gradniku SqlDataSource določili `ConnectionString`, nato smo določili SELECT stavek, ki je zapisan v lastnosti `SelectCommand`, poleg tega pa smo določili še parameter z imenom **aktiven**, ki je tipa Bool, privzeto vrednost pa ima **true**.
Ko smo konfigurirali SqlDataSource, bomo še gradniku ListView izbrali vir. Klik na zavihek **ListView Task in izberemo SqlDataSource1**. S tem smo povezali gradnik ListView in SqlDataSource.



Slika 60: Vir gradnika ListView

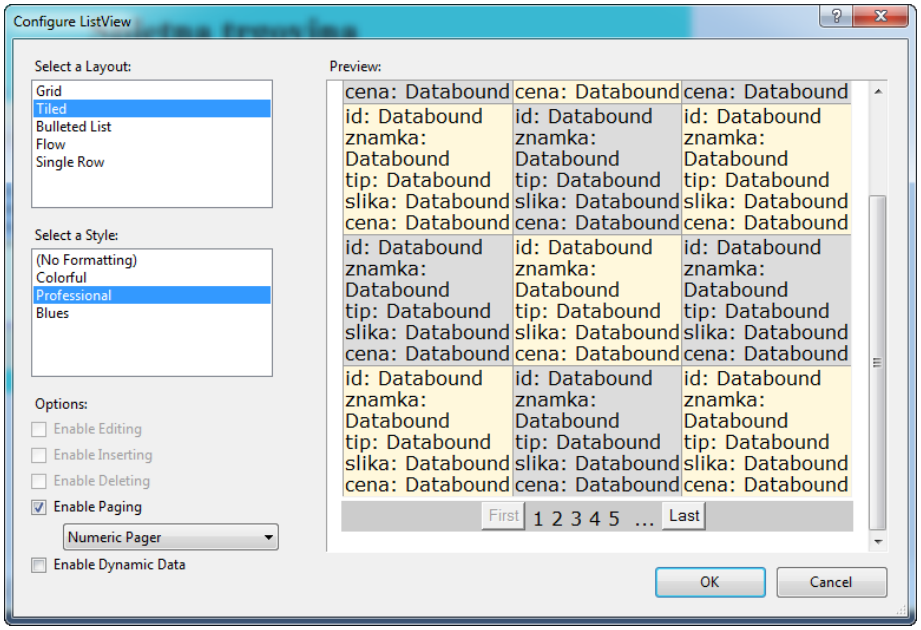
V zavihku **ListView Tasks** se nam ponudijo nove možnosti.



Slika 61: ListView Task

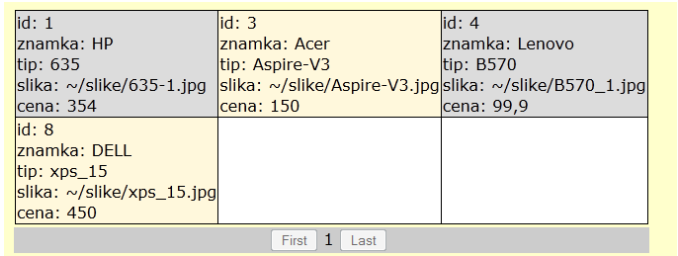
Kliknimo na možnost **Configure ListView …**
Odpre se okno, kjer določimo način prikazovanja podatkov in obliko. V oknu **Select a Layout** imamo več možnosti prikazovanja podatkov. Tu bi omenili samo možnost **Grid**, kar pomeni, da se podatki prikazujejo na tak način kot v gradniku GridView. Mi se bomo odločili za drugo

možnost, in sicer **Tiled**. Izdelki bodo prikazani v skupinah, vsaka skupina v svojem oknu. Koliko oken bo v vrstici, pa bomo določili v HTML kodi. V oknu **Select a Style** lahko izberemo tudi obliko (barve ozadja, obrobe …). Izberimo **Professional**. Če bi v SqlDataSource določili tudi edit, insert in delete ukaze, bi lahko pod zavihkom Options izbrali tudi te možnosti. Ker jih nismo, jih nimamo. Označimo pa možnost **Enable Paging** ter s spustnega seznama izberimo **Numeric Pager**. S to nastavitvijo določimo, da se na eni strani prikaže samo določeno število zapisov (izdelkov), pod izdelki pa se prikažejo navigacijski gumbi za krmarjenje po skupinah zapisov. Kliknimo OK.



Slika 62: Nastavitev izgleda gradnika ListView

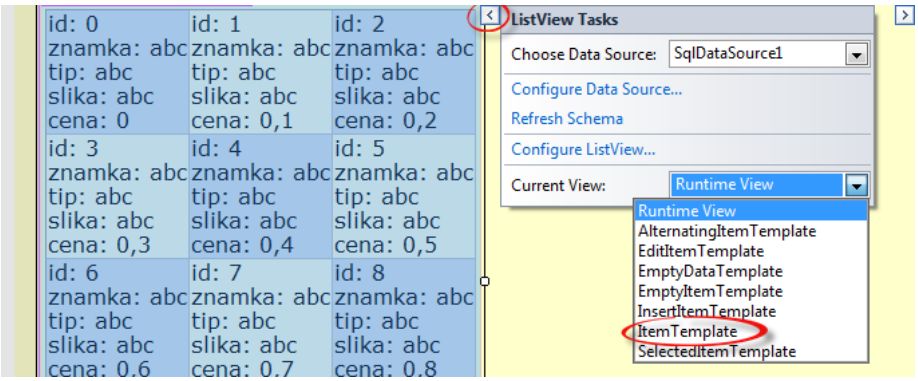
Če zaženemo projekt (F5), se prikažejo štirje izdelki.



Slika 63: Prikaz izdelkov v gradniku ListView v brskalniku

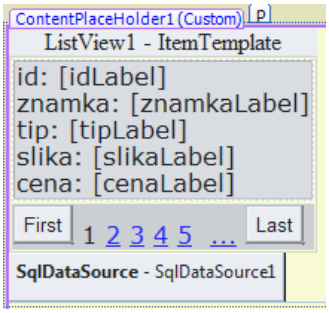
6.4.1 Template predloge (Item, AleternatingItem, Select, Insert, Edit, Update, Group)

Sedaj pa uredimo predlogo, ki določa, kako bodo izdelki prikazani v posamezni celici. Kliknimo na zavihek v ListView, kot prikazuje spodnja slika. Trenutno je izbran pogled (Current View) **Runtime View**. Le-ta prikazuje, v kakšni obliki bodo vsi izdelki, prikazani na spletni strani, mi pa izberimo **Item Template**.



Slika 64: ListView in ItemTemplate

Kot opazimo na spodnji sliki, ima sedaj gradnik ListView malo drugačno obliko. Pogled **Item Template** nam omogoča pogled posamezne celice oz. posameznega izdelka, v njem pa določimo, kako bodo razporejeni in prikazani posamezni podatki izdelka.



Slika 65: ItemTemplate

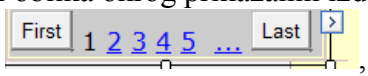
Če si sedaj ogledamo HTML kodo tega gradnika, bomo opazili, da nekaterih delov gradnika ListView sploh ne bomo potrebovali, zato jih bomo odstranili iz HTML kode.
Odstranimo torej sledeče značke in njihovo vsebino:

- `<AlternatingItemTemplate> … </AlternatingItemTemplate>` namen je isti kot pri Item Template, le da ta prikazuje sode celice, Item Template pa lihe. Če tega odstranimo, se bo prikazoval samo Item Template. V našem primeru ne bo več celic z rumenim ozadjem.
- `<EditItemTemplate> … </EditItemTemplate>` Ta nam omogoča urejanje izdelkov v bazi, vendar le, če imamo tudi v SqlDataSource definiran UPDATE izraz.
- `<InsertItemTemplate> … </InsertItemTemplate>` Podobno kot v zgornji točki, le da tu določimo obliko za vstavljanje novih izdelkov. Določen mora biti tudi INSERT izraz.
- `<SelectedItemTemplate> … </SelectedItemTemplate>` Ta del je namenjen obliki v primeru, da kliknemo na gumb, kjer je lastnost `CommandName="select".` V tem delu bi lahko samo v izbranem okencu prikazali podrobnosti posameznega izdelka. O tem malo več kasneje.

V gradniku ListView nam torej ostanejo sledeče značke:

- `<ItemTemplate>` Tu je HTML koda, ki se uporabi za obliko posamezne celice oz. izdelka. Ob tem gre omeniti `'<%# Eval("id") %>' />.` Znački `<%# … %>` označujeta začetek in konec skripta, ki črpa podatke iz SqlDataSourca, `Eval("id")` pa določa, iz

katerega polja se bo črpal podatek. V našem primeru se bo prikazala vrednost podatka `"id"`.

- `<LayoutTemplate>` Tu se določi oblika okrog prikazanih izdelkov. V njem je lahko vgnezden tudi `<asp:DataPager` 

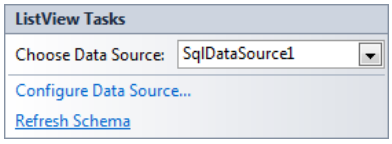
, ki omogoča krmarjenje po straneh zapisov. Tu lahko spremenimo vrednost lastnosti `PageSize="12".` Trenutno je nastavljeno, da bo na posamezni strani prikazanih največ 12 izdelkov.
- `<EmptyDataTemplate>` Določa obliko, če v zapisu ni podatkov. V našem primeru je tekst `No data was returned.`
- `<EmptyItemTemplate>` če v vrsti sploh ni več zapisa.
- `<GroupTemplate>` Nastavimo lahko lastnosti za celo skupino izdelkov.

Omeniti gre še eno lastnost v gradniku ListView in to je **`GroupItemCount="3"`**. S tem nastavimo, koliko zapisov oz. izdelkov bo v skupini oz. **koliko jih bo v posamezni vrstici**. Če to vrednost spremenimo v 4, bodo v vrstici prikazani štirje izdelki.
HTML gradnika ListView:

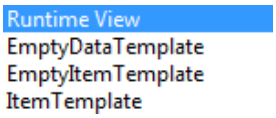
```
        <asp:ListView ID="ListView1" runat="server" DataKeyNames="id"
            DataSourceID="SqlDataSource1" GroupItemCount="3">
            <EmptyDataTemplate>
                <table runat="server"
                    style="background-color: #FFFFFF;border-collapse:
collapse;border-color: #999999;border-style:none;border-width:1px;">
                    <tr>
                        <td>
                            No data was returned.</td>
                    </tr>
                </table>
            </EmptyDataTemplate>
            <EmptyItemTemplate>
<td runat="server" />
            </EmptyItemTemplate>
            <GroupTemplate>
                <tr ID="itemPlaceholderContainer" runat="server">
                    <td ID="itemPlaceholder" runat="server">
                    a
                    </td>
                </tr>
            </GroupTemplate>
            <ItemTemplate>
                <td runat="server" style="background-color:#DCDCDC;color:
#000000;">
                    id:
                    <asp:Label ID="idLabel" runat="server" Text='<%# Eval("id")
%>' />
                    <br />znamka:
                    <asp:Label ID="znamkaLabel" runat="server" Text='<%#
Eval("znamka") %>' />
                    <br />tip:
                    <asp:Label ID="tipLabel" runat="server" Text='<%# Eval("tip")
%>' />
                    <br />slika:
                    <asp:Label ID="slikaLabel" runat="server" Text='<%#
Eval("slika") %>' />
                    <br />cena:
                    <asp:Label ID="cenaLabel" runat="server" Text='<%#
Eval("cena") %>' />
```

```
                    <br />
                </td>
            </ItemTemplate>
            <LayoutTemplate>
                <table runat="server">
                    <tr runat="server">
                        <td runat="server">
                            <table ID="groupPlaceholderContainer" runat="server"
border="1"
                                style="background-color: #FFFFFF;border-collapse:
collapse;border-color: #999999;border-style:none;border-width:1px;font-family:
Verdana, Arial, Helvetica, sans-serif;">
                                <tr ID="groupPlaceholder" runat="server">
                                </tr>
                            </table>
                        </td>
                    </tr>
                    <tr runat="server">
                        <td runat="server"
                            style="text-align: center;background-color:
#CCCCCC;font-family: Verdana, Arial, Helvetica, sans-serif;color: #000000;">
                            <asp:DataPager ID="DataPager1" runat="server"
PageSize="12">
                                <Fields>
                                    <asp:NextPreviousPagerField
ButtonType="Button" ShowFirstPageButton="True"
                                        ShowNextPageButton="False"
ShowPreviousPageButton="False" />
                                    <asp:NumericPagerField />
                                    <asp:NextPreviousPagerField
ButtonType="Button" ShowLastPageButton="True"
                                        ShowNextPageButton="False"
ShowPreviousPageButton="False" />
                                </Fields>
                            </asp:DataPager>
                        </td>
                    </tr>
                </table>
            </LayoutTemplate>
        </asp:ListView>
```

Če se po spremenjeni HTML kodi ne prikažejo več možnosti pogleda **Current View**,

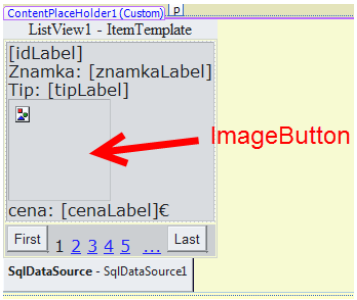


moramo najprej vse skupaj shraniti **Save All**, nato pa na zavihku **ListView Task** klikniti na **Refresh Schema**. Pojavi se okno **Refresh Data Source Schema**, na katerem pa kliknimo **OK**. Nato še kliknimo **NO**. Spet shranimo **Save All,** nato pa **zavihek ListView Tasks zapremo in še enkrat odpremo**. Zopet se nam ponudi možnost izbiranja **Current View**. Če si ogledamo možnosti, ki se ponudijo, opazimo, da ni več tistih, ki smo jih prej v HTML kodi izbrisali.



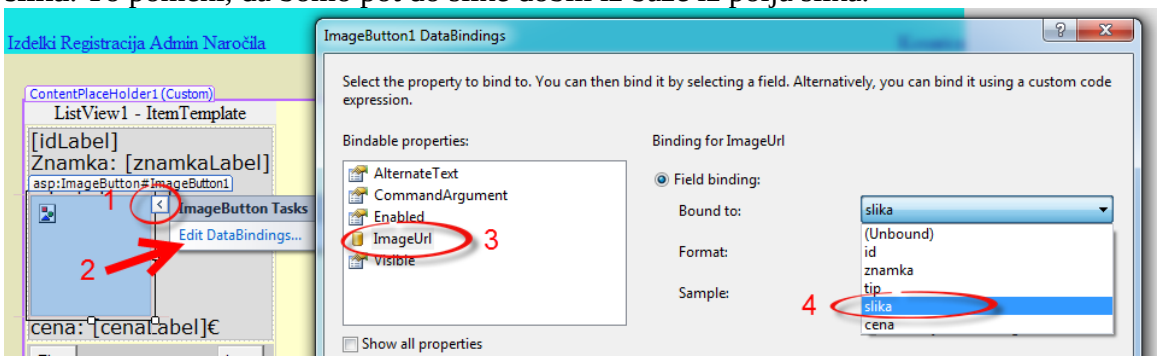
Slika 66: Možnosti pogleda Current View

Izberimo Item Template.
Določili bomo znamko, tip, sliko in ceno. Id bomo potrebovali, vendar ga ne želimo prikazati, zato ga bomo pomanjšali na 0px. Tako bomo vrednost id izdelka programsko prebrali, ne bo pa vidna kupcem.



Slika 67: Dodajanje slike v ItemTemplate

1. Najprej izbrišimo besedilo ID, izbrišimo sliko ter pri ceni za [cenaLabel] dodajmo znak **€**. Nato pomanjšamo besedilo za id. Desni klik na **[idLabel]** / **properties**, nato v oknu Properties razširimo zavihek **Font** ter v lastnost **Size** vpišimo vrednost **0pt**. S tem smo poskrbeli, da kupcu skrijemo id izdelka.
2. Dodajmo gradnik ImageButton iz orodjarne ToolBox. Primimo desni spodnji kot tega gradnika in ga poljubno raztegnimo. Desni klik na ta gradnik / properties in iz okna Properties spremenimo vrednosti lastnosti **Width** in **Height** oba na **100px**. S tem smo zagotovili, da bo imela slika vedno velikost 100px * 100px. Sledimo korakom spodnje slike. Gradniku ImageButton kliknimo na zavihek **ImageButton Tasks (1)**, nato kliknimo na možnost **Edit DataBindings ... (2).** Odpre se okno, v katerem bomo lastnosti **ImageUrl (3)** določili pot do slike. Pod točko **(4)** bomo izbrali **Field binding:, Bount to: slika**. To pomeni, da bomo pot do slike dobili iz baze iz polja slika.



Slika 68: Vir slike v ItemTemplate

3. Preklopimo na pogled Source (HTML) in gradniku ImageButton dodajmo novo lastnost, in sicer `CommandName="select"` tako, kot je na spodnji kodi odebeljeno in podčrtano.

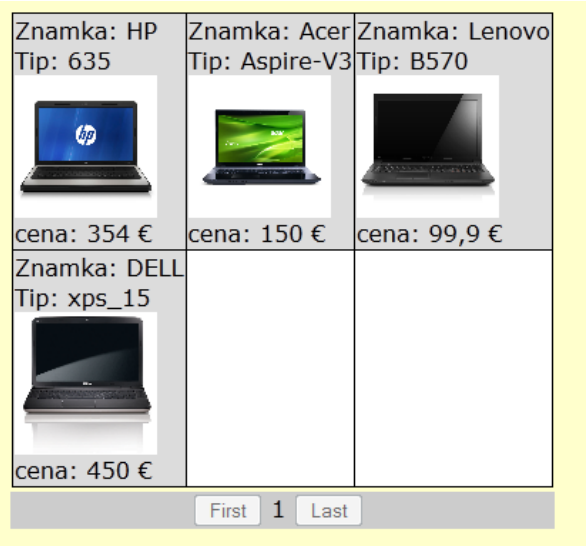
HTML ItemTemplate:

```
<ItemTemplate>
```

```
        <td runat="server" style="background-color:#DCDCDC;color: #000000;">
            <asp:Label ID="idLabel" runat="server" Text='<%# Eval("id") %>'
              Font-Size="0pt" />
            <br />
            Znamka:
            <asp:Label ID="znamkaLabel" runat="server" Text='<%# Eval("znamka")
%>' />
            <br />
            Tip:
            <asp:Label ID="tipLabel" runat="server" Text='<%# Eval("tip") %>' />
            <br />
            <asp:ImageButton ID="ImageButton1" runat="server" Height="100px"
            ImageUrl='<%# Eval("slika") %>' Width="100px" CommandName="select"
/>
            <br />cena:
            <asp:Label ID="cenaLabel" runat="server" Text='<%# Eval("cena")
%>'></asp:Label>
            €<br />
        </td>
</ItemTemplate>
```

Možnosti CommandName so: select, insert, update in delete. S tem, ko smo izbrali select, bomo gradniku ListView spremenili lastnost SelectedIndex, kar bomo s pridom izkoristili, da bomo prebrali **id** izdelka iz želene celice.

Zaženimo projekt (F5) v brskalniku, pa dobimo naslednji rezultat.

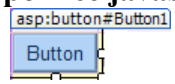


Slika 69: Prikaz izdelkov s sliko z ListView v brskalniku

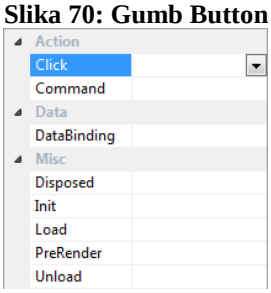
7 Dogodki

Dogodki so obvezen del uporabniku prijazne spletne aplikacije. Namenjeni so temu, da uporabnik čim lažje upravlja z obrazci, še posebej pa so v pomoč razvijalcu spletne aplikacije. Vsak gradnik ima svoje možnosti in namen, zato so tudi dogodki, ki jih nudijo gradniki, specifični glede na namen. Poglejmo si enega najpogosteje uporabljenih gradnikov (gumb – Button). Če iz orodne vrstice Toolbox z miško na obrazec nesemo gradnik Button, mu lahko v

oknu Properties nastavljamo različne lastnosti. Da bi videli, kakšne dogodke nudi ta gradnik, na vrhu okna Properties kliknimo  na gumb, na katerem je narisana rumena strela. Če si spodaj ogledamo dogodke, ki jih nudi ta gradnik, gre omeniti najbolj uporabljenega oz. najbolj primernega za svoj namen. To je dogodek **Click**. Če ga bomo definirali in sprogramirali v datoteki .aspx.cs, se bo vsakič, ko bomo kliknili na ta gumb, izvedla koda, ki smo jo zapisali v .aspx.cs v metodi **Button1_Click**. **Dejansko metodo Button1_Click pokliče javascript na dogodek onclick,** ki je definiran v HTML kodi v gradniku Button.

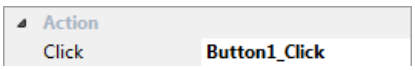


Slika 70: Gumb Button



Slika 71: Dogodek Click

Če v oknu **Properties/Events** sedaj dvakrat kliknemo na dogodek **Click**, se zraven pojavi ime **Button1_Click**.



Slika 72: Dogodek Clikc in Metoda Button1_Click

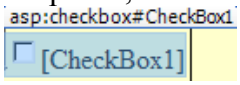
V HTML kodi gumba opazimo javascript dogodek **onclick**, ki pokliče metodo **Button1_Click**:

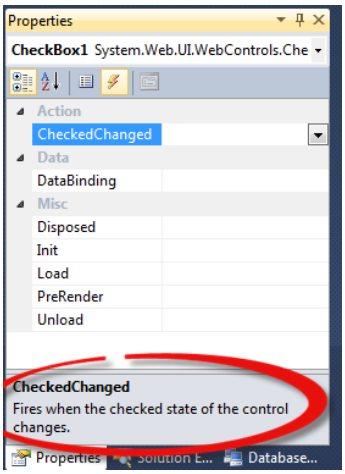
```
<asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="Button" />
```

Avtomatsko se odpre datoteka .aspx.cs, v kateri lahko definiramo, kaj se bo zgodilo, ko se sproži ta dogodek:

```
protected void Button1_Click(object sender, EventArgs e)
{
      //koda
}
```

Če si pogledamo npr. CheckBox, vidimo, da je njegov glavni dogodek **oncheckedchanged**, ki pokliče metodo **CheckedChanged**. Na spodnji sliki imamo označeno razlago, ki nam pove, da se bo sprožil, če bomo gradniku spremenili stanje.



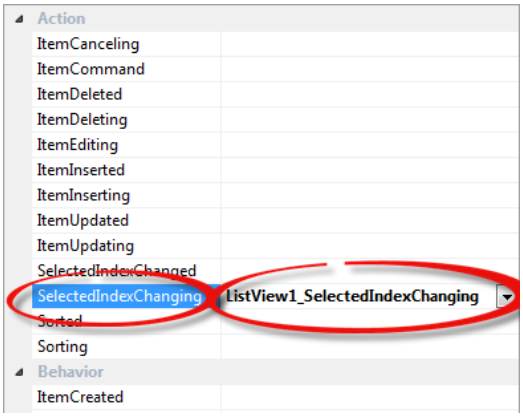


Slika 73: Dogodek CheckedChanged

Tako lahko pregledamo vse dogodke gradnikov, ki jih nudijo. **Izbrišimo gradnika Button in CheckBox** ter v datoteki .aspx.cs **izbrišimo metodo Button1_Click**.

Vrnimo se nazaj na našo trgovino v **Izdelki.aspx**.
Poskrbimo, da se bo ob kliku na sliko izdelka odprla nova stran, v kateri bodo vse podrobnosti, ki jih hranimo v bazi o izbranem izdelku.
Gradniku ListView določimo pogled (**Current View**) v **RunTimeView.** Desni klik na ta gradnik in **properties**. Odpre se nam **okno Properties**, v katerem lahko nastavljamo različne lastnosti tega gradnika. Ker pa bi radi na klik prikazali podrobnosti izdelka, moramo okno preklopiti na možnost **Events** (Dogodki). Gumb za preklop se nahaja na vrhu okna Properties in ima narisano nekakšno rumeno strelo, tako kot vidite na spodnji sliki.

Kliknimo na ta gumb. Prikaže se okno, v katerem lahko aktiviramo različne dogodke, ter v ozadju na sprožen dogodek naredimo tudi ustrezno akcijo.
Pazimo! Na voljo imamo dva skoraj isto poimenovana dogodka, in sicer **SelectedIndexChanged** in **SelectedIndexChanging**. Zelo moramo biti pozorni, da **izberemo drugega (SelectedIndexChanging)**, nato pa dvakrat kliknimo nanj. Zraven napisa se pojavi generično ime metode **ListView1_SelectedIndexChanging.**



Slika 74: Dogodek SelectedIndexChanging in metoda ListView1_SelectedIndexChanging

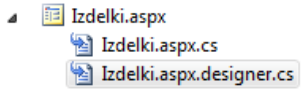
Javascript dogodek `onselectedindexchanging` bo sprožil metodo `ListView1_SelectedIndexChanging`.
HTML v glavi gradnika ListView:

```
<asp:ListView ID="ListView1" runat="server" DataKeyNames="id"
    DataSourceID="SqlDataSource1" GroupItemCount="3"
    onselectedindexchanging="ListView1_SelectedIndexChanging">
```

V datoteki Izdelki.aspx.cs pa je pojavi koda:

```
//vključene knjižnice
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
//Ime projekta
namespace spletnaTrgovina
{
    //ime obrazca
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        //izvede se vsakič, ko se stran naloži
        }

        protected void ListView1_SelectedIndexChanging(object
sender, ListViewSelectEventArgs e)
        {
 //izvede se, ko z miško kliknemo na sliko v ListView.
        }
    }
}
```

Trenutno odprto datoteko Izdelki.aspx.cs lahko najdemo tudi v Solution Explorerju pod zavihkom Izdelki.aspx.



V Izdelki.aspx.cs je torej zapisana koda C#, v datoteki Izdelki.aspx.designer.cs pa so definirani vsi gradniki, ki smo jih dodali iz orodjarne na obrazec.

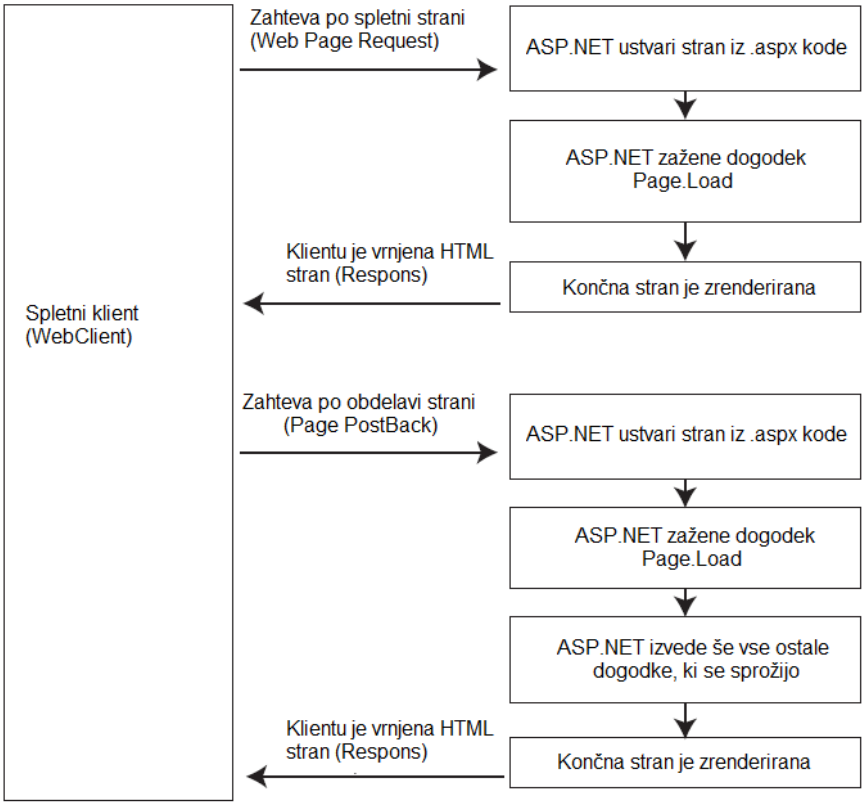
Sedaj bomo v metodo `ListView1_SelectedIndexChanging` zapisali kodo, ki bo prebrala **id** izbranega izdelka, jo dodali v Session, nato pa ga na obrazcu Izdelek.aspx prikazali z vsemi podrobnostmi, ki so zapisane v bazi.

8 Prenos podatkov

Kadar bomo prenašali podatke ter uporabljali spremenljivke, si moramo najprej pogledati, kaj se dogaja z njimi, kakšna je njihova življenjska doba in kako se odvijajo koraki ob zahtevi in ponovni zahtevi določene strani, nato pa še upravljanje s Session-om.

8.1 Življenjski cikel strani (Page Life Cycle)

Poglejmo, kako si sledi tok dogodkov, ko pride do zahteve po strani (Page Request) ali po ponovni obdelavi strani (Page PostBack npr: Button_Click,…).



Slika 75: Življenjski cikel strani

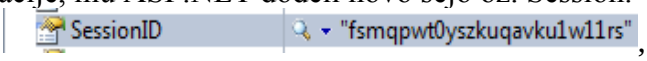
Razlaga zaporedja akcij, če pride do ponovnega zahtevka strani (PostBack):

- Na klientovi strani Javascript pokliče funkcijo (dogodek)_doPostBack (Button.Click, Checkbox.CheckedChanged …).
- ASP.NET re-kreira stran iz datoteke .aspx.
- Kontrole se napolnijo z informacijami iz stanja ViewState.
- Sproži se dogodek Page.Load.
- Sprožijo se vsi ostali dogodki (Button.Click, CheckBox.CheckedChenged, ListView.SelectedIndexChanged …).
- Sproži se dogodek Page.PreRender.
- Nova stanja (vrednosti) kontrol se shranijo v ViewState.
- Zgrajena (renderirana) je HTML stran (le-ta se ne da več spreminjati).

- Sproži se dogodek Page.Unload.
- HTML se pošlje klientu.
- Stran je izbrisana iz spomina (tudi spremenljivke).

Če si dobro ogledamo tok dogodkov, lahko iz tega sklepamo, da se vse nestatične spremenljivke med staro stranjo in postBack stranjo uničijo, zato se izgubijo tudi vse vrednosti, ki so shranjene v njih. Da se stare vrednosti gradnikov ohranijo, se gradnikom vse vrednosti in spremembe shranijo v stanje ViewState. Z drugimi besedami, ASP.NET zagotovi, da se vse vrednosti gradnikov prenašajo preko stanja ViewState. Lahko bi pošiljali tudi vrednosti spremenljivk preko stanja ViewState, vendar to stanje ni namenjeno temu, zato bomo uporabili Session.

8.2 Session

Vsakič, ko klient dostopa do te spletne aplikacije, mu ASP.NET dodeli novo sejo oz. Session. Vsak Session ima svojo unikatno kodo, npr: 

, in traja, dokler uporabnik ne zapre brskalnika ali ne obišče neke druge spletne strani ali po daljšem obdobju svoje neaktivnosti. Strežnik v tem zadnjem primeru ne ve, kaj se z uporabnikom dogaja, zato po cca. 20 min. neaktivnosti klienta sam prekine Session. Takrat so izgubljene tudi vse vrednosti, ki so bile shranjene v Session.
Session je pomemben tudi za shranjevanje in prenašanje vrednosti spremenljivk in objektov. Vanjo lahko zapišemo tudi kakšno poizvedbo iz baze (dataset) zato, da je po nepotrebnem ne obremenjujemo z enimi in istimi poizvedbami. Če lahko vrednosti shranjujemo v Session, potem iz njega lahko tudi beremo.

Primer shranjevanja preproste celoštevilske spremenljivke:

```
int x = 5; //Spremenljivki x dodelimo vrednost 5.
Session["st"] = x; //V stanju Session se ustvari nova spremenljivka "st", ki dobi
vrednost spremenljivke x, torej 5.
```

Branje vrednosti iz stanja Session:

```
int y = (int)Session["st"]; //Iz stanja Session iz spremenljivke "st" pridobimo
vrednost ter jo explicitno (casting) pretvorimo v celo število (int)
```

Nekaj primerov nepravilne uporabe:

```
string z = (string)Session["st"]; //Napaka - števila ne moremo explicitno
pretvoriti v string.
int k = (int)Session["st1"]; //Napaka, spremenljivka "st1" v Session-u ne obstaja,
zato ima vrednost NULL, te vrednosti pa ne moremo pretvoriti v noben drug
podatkovni tip, zato program med izvajanjem javi napako.
```

Za spodnji primer imamo definirano vsebino razreda Oseba:

```
class Oseba
{
    public string ime { get; set; }
    public int letnik { get; set; }
}
```

Primer shranjevanja tabelarične spremenljivke List< >:

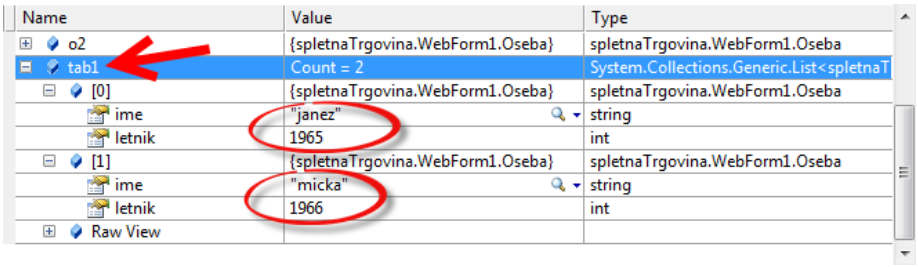
```
List<Oseba> tab = new List<Oseba>();//Definirano novo zbirko (collections generic)
tipa Oseba.
```

```
Oseba o1 = new Oseba(); //Ustvarimo nov objekt tipa oseba.
o1.ime = "janez"; //Objektu o1 dodelimo vrednosti.
o1.letnik = 1965;
tab.Add(o1); //Objekt o1 z vrednostmi dodamo zbirki tab.

Oseba o2 = new Oseba();//Nov objekt tipa oseba.
o2.ime = "micka"; //Dodelimo nove podatke.
o2.letnik = 1966;
tab.Add(o2); //Objekt o2 z novimi vrednostmi dodamo zbirki.

Session["x"] = tab; //V Session v spremenljivki "x" zapišemo zbirko s podatki tipa
oseba.
```

Primer branja zbirke List< > iz Sessiona - lahko na isti strani ali pa na kakšni drugi strani v projektu.

```
List<Oseba> tab1 = (List<Oseba>)Session["x"]; //Ustvarimo zbirko tipa oseba ter
vrednost "x" iz Sessiona explicitno pretvorimo v zbirko tipa Oseba.
```

Po pretečeni kodi ima spremenljivka tab1 shranjena dva objekta.



Slika 76: Stanje spremenljivk v zbirki List

Zapisovanje in branje takih tabelaričnih spremenljivk v Session bomo kasneje uporabili za košarico.

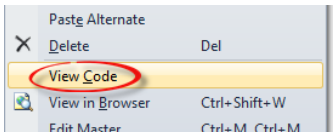
Vrnimo se nazaj k naši datoteki **Izdelki.aspx.cs**. Napišimo sledečo kodo v metodo **ListView1_SelectedIndexChanging,** tako kot je prikazano spodaj.

```
protected void ListView1_SelectedIndexChanging(object sender,
ListViewSelectEventArgs e)
{
//Iz gradnika ListView dobimo zaporedno številko celice, v katero smo kliknili.
     int indx = e.NewSelectedIndex;
//Iz celice, v katero smo kliknili, preberemo vrednost id izdelka. DataKeys je
definiran tudi v glavi gradnika ListView v katerem je določeno, da se črpa iz
polja id.
     string id = ListView1.DataKeys[indx].Value.ToString();
//id izdelka shranimo v Session v spremenljivko "id". To ime spremenljivke si
moramo zapomniti, ker jo bomo potrebovali tam, kjer jo bomo želeli prebrati.
     Session["id"] = id;
//preusmerimo stran na Izdelek.aspx
     Response.Redirect("Izdelek.aspx");
}
```

Odprimo datoteko **Izdelek.aspx** ter vanjo postavimo gradnik **Label**.

HTML: `<asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>`

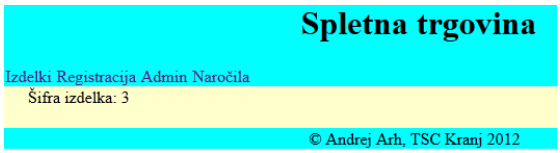
Desni klik na obrazec in kliknimo možnost **View Code**.



Odpre se nam datoteka Izdelek.aspx.cs, v kateri bomo v metodo **Page_Load** zapisali naslednjo kodo. Na tej strani se bo torej izpisalo **ID izdelka: #**.

```
//Dogodek Page.Load se izvede vsakič, ko se stran naloži v brskalniku
protected void Page_Load(object sender, EventArgs e)
{
    //iz Session preberemo vrednost spremenljivke "id"
    string idIzdelka = Session["id"].ToString();
    //v labelo v lastnost Text zapišemo besedilo:
    Label1.Text = "ID izdelka: " + idIzdelka;
}
```

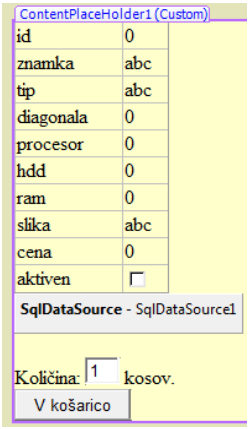
Če sedaj poženemo projekt (F5), nato pa kliknemo na sliko enega izdelka, se nam odpre stran Izdelek.aspx, v njej pa je izpisana šifra izbranega izdelka.



Slika 77: Prikaz id izbranega izdelka v brskalniku

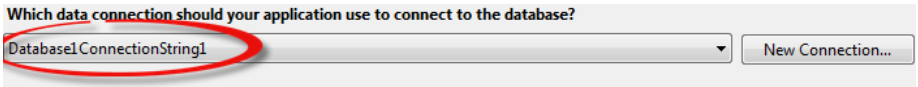
Naša naloga je, da sedaj izpišemo vse podrobnosti izbranega izdelka. Pri tem bomo uporabili nov gradnik **DetailsView** (dobimo ga v orodjarni pod zavihkom **DATA**, ki lahko naenkrat prikazuje samo en izdelek. Poleg tega bomo morali zgraditi nov **DataSource**, ki bo izpisal samo določen izdelek, na koncu pa še gumb, ki bo lahko izbrani izdelek dodal v košarico.

Najprej **izbrišimo gradnik Label** ter v datoteki **Izdelek.aspx.cs izbrišimo kodo**, ki je zapisana v metodi Page_Load() {**BRIŠI**}. V datoteki **Izdelek.aspx** na obrazec **dodajmo** gradnike **DetailsView, SqlDataSource, TextBox in Button**. **Gradniku DetailsView določimo DataSource, in sicer SqlDataSource1**. Gradniku **Button in TextBox** spremenimo Text. Desni klik na **Button/properties**, v oknu **Properties** pa v lastnost **Text** zapišimo besedilo **V košarico**, TextBoxu pa **1**. Okrog TextBoxa dodajmo še besedilo **Količina:** in **kosov**, nato pa zmanjšajte širino TextBoxa z miško na cca. **20px**.



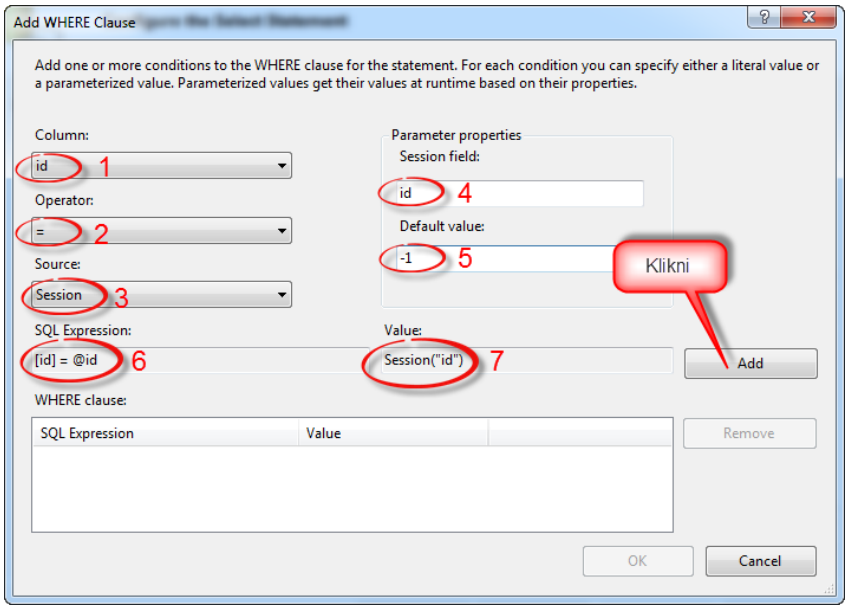
Slika 78: Razporeditev gradnikov DetailsView, TextBox in Button.

Definirajmo še poizvedbo, in sicer na podoben način kot v poglavju, kjer smo opisovali gradnik ListView.
Odprimo zavihek **SqlDataSource Task** na gradniku SqlDataSource in izberimo možnost **Configure DataSource**. Odpre se nam čarovnik, v katerem bomo najprej določili ConnectionString.



Slika 79: SqlDataSource in ConnectionString

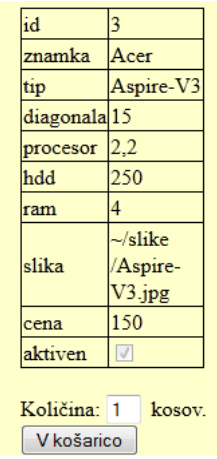
Kliknimo gumb **Next**. Ker bi radi prikazali vsa polja o določenem izdelku, bomo pustili privzeto nastavitev, in sicer je označena samo *****, SQL izraz pa mora biti sledeč:
SELECT * FROM [tabIzdelkov].
Določimo še pogoj. Izpisali bomo samo en izdelek, ki ima določen **id**. Kliknimo gumb **WHERE**, kjer se nam odpre okno za nastavljanje pogoja. Pri gradniku ListView smo izbrali SELECT parameter, ki ima privzeto vrednost TRUE, v tem primeru bomo **vrednost prebrali kar iz Session-a**, zato da ne bo potrebno programsko dodeljevati vrednosti parametra.
Izpolnimo vrednosti tako, kot kaže spodnja slika.



Slika 80: Nastavitev WHERE pogoja s Session parametrom

1. Izberemo kolono, ki jo bomo uporabili v preverjanju pogoja. Poiščemo se samo en izdelek z določenim **id**.
2. Operator naj bo enačaj.
3. Source določa, iz kje se bo pridobila vrednost parametra. Izberimo **Session**.
4. Določimo, iz katere spremenljivke v Sessionu se bo pridobila vrednost parametra (**id**).
5. Default value: določimo privzeto vrednost za primer, da nimamo izbranega nobenega izdelka. -1 pa zato, ker se ID-ji začnejo od 0 naprej. Torej noben izdelek ne more imeti id-ja z vrednostjo -1.
6. Izpiše se izraz pogoja.
7. Prikaže se vir, iz katerega se bo pridobila vrednost parametra.

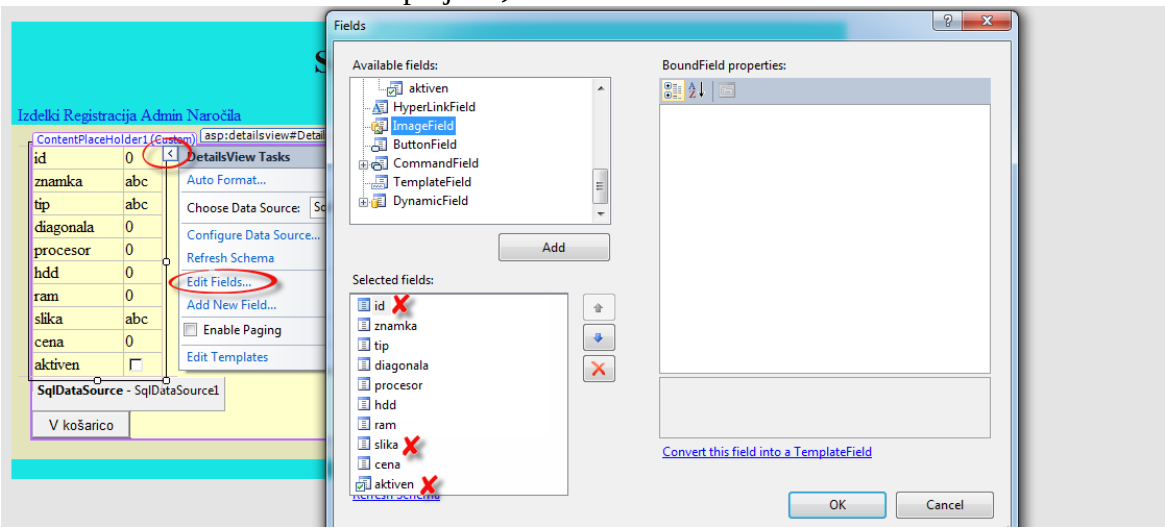
Kliknimo gumb **Add**, nato pa še **OK**. Zapre se okno za definiranje pogoja, nato kliknimo **Next**, nato pa še **Finish**. S tem smo definirali SqlDataSource, da bo prikazal vse vrednosti za izbrani izdelek.
Zaženimo projekt, kliknimo na en izdelek. V brskalniku se prikažejo podrobnosti izbranega izdelka.



Slika 81: Neoblikovan prikaz izdelka z gradnikom DetailsView v brskalniku

Kot smo že navajeni, moramo polje slika zamenjati z gradnikom Image, ki nam bo prikazoval sliko izdelka.

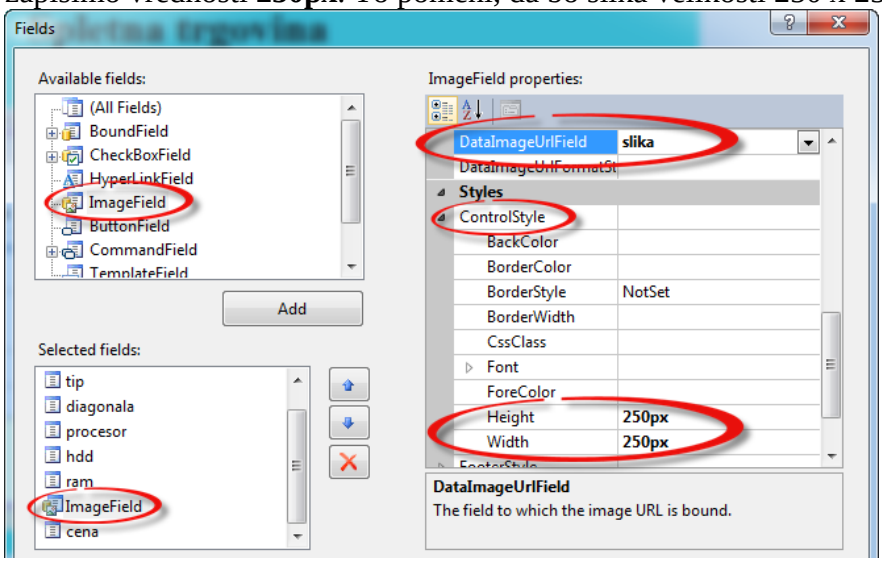
1. Kliknimo na zavihek **DetailsView Task**.
2. Izberimo možnost **Edit Fields**.
3. Odpre se okno za urejanje gradnika.
4. Iz okna Selected fields: **izbriši** polja **id, slika in aktiven**.



Slika 82: Nastavitev in oblikovanje polj v gradniku DetailsView

5. Iz okna Aveliable fields izberimo **ImageField** in kliknimo **Add**.
6. V oknu Selected fields premaknimo **ImageField** polje na predzanje mesto, tako kot vidimo na spodnji sliki **(ZELO POMEMBEN JE VRSTNI RED POLJ, KER BOMO POTEM PREKO INDEXA VRSTICE BRALI VREDNOSTI (glej spodnjo sliko). Če zamenjamo npr. sliko in ceno, nam po spodnji kodi program ne bo deloval, ker bi pred zamenjavo imela slika index 6, po zamenjavi pa bi imela slika index 7. Torej bi prebrali napačen podatek.**).
7. Na desni strani v oknu ImageField properties-lastnosti **DataImageUrlField** izberimo vir **slika**.

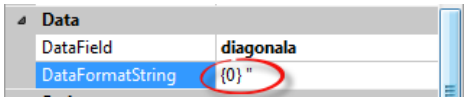
8. V zavihku Styles izberimo podzavihek **ControlStyle** ter v lastnosti **Height** in **Width** zapišimo vrednosti **250px**. To pomeni, da bo slika velikosti 250 x 250px.



Slika 83: Nastavitev lastnosti slike (ImageField)

Poskrbimo še, da se bodo vrednostim v gradniku DetailsView poleg števila zapisale tudi enote. Diagonala bo v **PALCIH** **''**, procesor v **GHz**, velikost diska HDD v **GB**, ram v **GB** in cena v **€**.

1. Kliknimo na polje **diagonala** v oknu Selected fields.
2. V oknu **BoundField properties** poiščemo zavihek **Data** in v lastnost **DataFormatString** zapišemo vrednost **{0}** **''**, tako kot kaže spodnja slika. Tako smo formatirali izpis za polje diagonala. Izpis bo npr: **17** **''**.
   Več o formatiranju stringov si lahko pogledamo na naslednjih straneh:
   http://www.csharp-examples.net/string-format-double/
   http://idunno.org/archive/2004/07/14/122.aspx
   http://www.howtogeek.com/?post_type=post&p=123

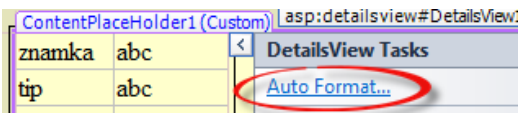


Slika 84: Formatiranje izpisa

Izberimo še ostala polja in jim v lastnost DataFormatString zapišimo naslednje vrednosti:
- procesor: **{0} GHz**
- hdd: **{0} GB**
- ram: **{0} GB**
- cena: **{0} €**

3. Kliknimo OK.

Da bomo dobili malo drugačen izgled, kot smo vajeni, kliknimo na zavihek na gradniku DetailsView ter izberimo možnost **AutoFormat …**



Slika 85: Auto Format

Odpre se okno z že vnaprej določenimi oblikami. Recimo, da bomo izbrali **Sand & Sky**. Kliknimo OK. Dobili smo eno od oblikovanih predlog, ki jih ponuja VWD. Če zaženemo projekt (F5) ter kliknemo na en izdelek, dobimo sledeče:

Slika 86: Oblikovan prikaz izdelka z DetailsView v brskalniku

Sedaj imamo prikazano tudi sliko velikosti 250 x 250 px ter vse pripadajoče fizikalne enote HTML gradnikov DetailsView in SqlDataSource:

```
    <asp:DetailsView ID="DetailsView1" runat="server" AutoGenerateRows="False"
        BackColor="LightGoldenrodYellow" BorderColor="Tan" BorderWidth="1px"
        CellPadding="2" DataKeyNames="id" DataSourceID="SqlDataSource1"
        ForeColor="Black" GridLines="None" Height="50px" Width="125px">
        <AlternatingRowStyle BackColor="PaleGoldenrod" />
        <EditRowStyle BackColor="DarkSlateBlue" ForeColor="GhostWhite" />
        <Fields>
            <asp:BoundField DataField="znamka" HeaderText="znamka"
                SortExpression="znamka" />
            <asp:BoundField DataField="tip" HeaderText="tip" SortExpression="tip"
/>
            <asp:BoundField DataField="diagonala" DataFormatString="{0} &amp;quot;"
                HeaderText="diagonala" SortExpression="diagonala" />
            <asp:BoundField DataField="procesor" DataFormatString="{0} GHz"
                HeaderText="procesor" SortExpression="procesor" />
            <asp:BoundField DataField="hdd" DataFormatString="{0} GB"
HeaderText="hdd"
                SortExpression="hdd" />
            <asp:BoundField DataField="ram" DataFormatString="{0} GB"
HeaderText="ram"
```

```
                SortExpression="ram" />
            <asp:ImageField DataImageUrlField="slika">
                <ControlStyle Height="250px" Width="250px" />
            </asp:ImageField>
            <asp:BoundField DataField="cena" DataFormatString="{0} €"
HeaderText="cena"
                SortExpression="cena" />
        </Fields>
        <FooterStyle BackColor="Tan" />
        <HeaderStyle BackColor="Tan" Font-Bold="True" />
        <PagerStyle BackColor="PaleGoldenrod" ForeColor="DarkSlateBlue"
            HorizontalAlign="Center" />
    </asp:DetailsView>
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
        ConnectionString="<%$ ConnectionStrings:Database1ConnectionString1 %>"
        SelectCommand="SELECT * FROM [tabIzdelkov] WHERE ([id] = @id)">
        <SelectParameters>
            <asp:SessionParameter DefaultValue="-1" Name="id" SessionField="id"
                Type="Int32" />
        </SelectParameters>
    </asp:SqlDataSource>
<br />
    Količina:
    <asp:TextBox ID="TextBox1" runat="server" Width="20px">1</asp:TextBox>
 kosov.<br />
    <asp:Button ID="Button1" runat="server" Text="V košarico" />
```

Pri SqlDataSource sedaj opazimo, da imamo SessionParameter, ki pa ima podobno vlogo kot Parameter. SessionParameter mora imeti definirano lastnost SessionField, ki določa, iz katere spremenljivke v Sessionu se bo črpala vrednost.
Zraven imamo še gradnika TextBox, ki bo služil za vnos količine in Button, ki bo ob kliku sprožil dodajanje novega izdelka v košarico.

9 Košarica

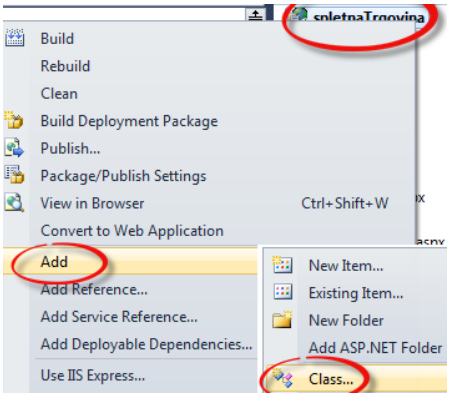
Ste že kdaj bili v trgovini in ste kupovali različne izdelke, pred blagajno pa ste se spomnili, da ste vzeli napačen izdelek, zato ste ga morali vzeti iz nje? Kaj pa, če ga ne bi mogli več vzeti ven? Bi morali še enkrat od začetka ter ponovno nabrati želene izdelke?
Podobno kot v fizični trgovini imamo tudi v spletni trgovini nakupovalno košarico. V njej so ravno tako shranjeni izdelki, ki smo jih izbrali. Seveda na izgled ni plastična in nima ročaja, ima pa prostor za izdelke in njegove atribute. Ne nosimo je v roki, ampak bo na voljo v Sessionu.
Premislimo torej, kaj bomo shranili v košarico oz. kakšne podatke o izdelku bomo potrebovali v košarici.

a) id izdelka
b) znamka
c) tip (model)
d) slika
e) cena
f) količina
g) cena skupaj

Vse, kar potrebujemo, je ustrezna podatkovna struktura za posamezni izdelek. Praktično bomo torej košarico naredili tako, da bomo definirali nov razred **kosarica**, v katerem bomo definirali

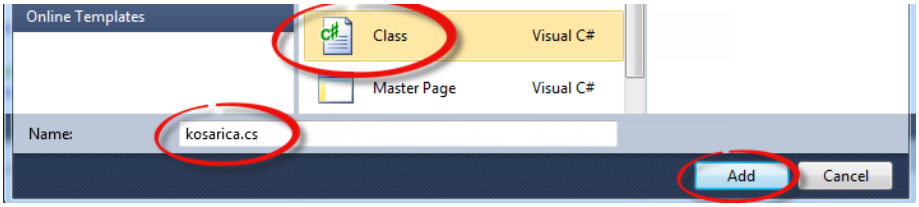
lastnosti izdelka (kot je našteto zgoraj). Cela košarica se bo hranila v Sessionu, definirana pa bo iz generične zbirke (collections generic - **List<kosarica>**) in bo tipa **kosarica** (razred, ki ga bomo še definirali). Vsak zapis v tej zbirki bo imel vse zgoraj naštete podatke: od id izdelka do skupne cene.

V projektu (Solution Explorer) ustvarimo nov **razred** (podatkovni razred – Data Class) z imenom **kosarica**. V projektu pa zato, da bo dosegljiv v vseh datotekah projekta, kjer koli ga bomo potrebovali. V Solution Explorerju desni klik na projekt in **Add/Class …**



Slika 87: Dodajanje razreda košarica 1

Odpre se okno, v katerem izberemo datoteko, ki jo bomo dodali projektu. Izberimo **Class** in jo poimenujmo **kosarica.cs** ter kliknimo **Add**.



Slika 88: Dodajanje razreda kosarica 2

Odpre se datoteka kosarica.cs, ki ima naslednjo vsebino:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace spletnaTrgovina
{
    public class kosarica
    {
    }
}
```

V naslednjem koraku moramo razredu kosarica definirati posamezna polja in lastnosti. Lastnosti pa zato, da lahko avtomatično prikazujemo podatke iz zbirke List<…>.

Oglejmo si še nekaj podrobnosti o poljih in lastnostih v razredu (class).
V spodnji kodi smo definirali privatno spremenljivko id ter javno lastnost ID, ki omogoča branje in pisanje privatnega polja id.

```
private int id;//privatno polje id
public int ID //javna lastnost ID, ki omogoča branje in pisanje v polje id
{
    get { return id; } //vrne vrednost polja id
    set { id = value; } //zapiše vrednost v polje id
}
```

Poznamo pa tudi krajši zapis, ki ima popolnoma enak učinek kot zgornja koda. Temu se reče avtomatična lastnost (automatic properties) in ravno tako ustvari privatno polje, ki ima javno lastnost, ki iz tega privatnega polja bere in vanj piše.
Koda avtomatične lastnosti: `public int id { get; set; }`
Kot opazimo, je ta način precej krajši, zato bomo vsa polja in njihove lastnosti definirali na ta drugi način.

Koda košarice:

```
    public class kosarica
    {
//Konstruktor, ki ob ustvarjanju novega objekta vse vrednosti postavi na 0, 0.0
ali "".
        public kosarica() { }

//Preobložen (overloaded) konstruktor, ki ga ni potrebno definirati!!!
//Pokliče se, če ob ustvarjanju novega objekta podamo tudi vrednosti parametrov,
ta pa nato dodeli vsem poljem te vrednosti.
        public kosarica(int id, string znamka, string tip, string slika, double
cena, double kolicina)
        {
            this.id = id; //privatno polje id dobi vrednost parametra id
            this.znamka = znamka;
            this.tip = tip;
            this.slika = slika;
            this.cena = cena;
            this.kolicina = kolicina;
        }
//Definirajmo še privatna polja in javne lastnosti.
        public int id { get; set; }
        public string znamka { get; set; }
        public string tip { get; set; }
        public string slika { get; set; }
        public double cena { get; set; }
        public double kolicina { get; set; }
//Lastnost cenaSkupaj pa omogoča samo branje, torej je read-only, ker nima set
metode.
        public double cenaSkupaj
        {
//Vrednost pa lahko izračunamo kar iz vnešenih podatkov: količina * cena.
            get { return kolicina * cena; }
        }
    }
```

Shranimo datoteko **kosarica.cs**.

Odprimo datoteko Izdelek.aspx ter dvakrat kliknimo na gumb **V kosarico**.
HTML koda gradnikov TextBox in Button v datoteki Izdelek.aspx pod gradnikom GridView:

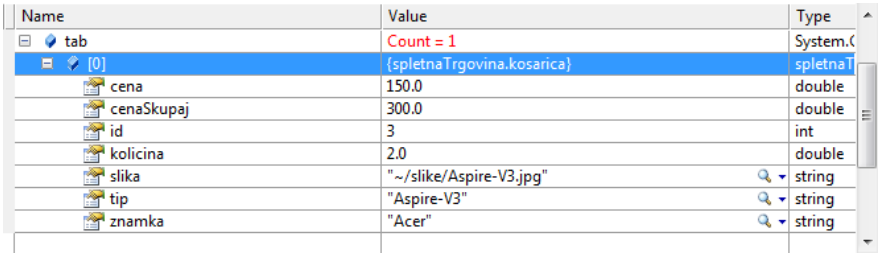
```
    Količina:
    <asp:TextBox ID="TextBox1" runat="server" Width="20px">1</asp:TextBox>
 kosov.<br />
    <asp:Button ID="Button1" runat="server" Text="V košarico"
        onclick="Button1_Click" />
```

V gradniku Button je sedaj definiran dogodek **onclick**, ki sproži metodo **Button1_Click**.
Odpre se datoteka **Izdelek.aspx.cs**, v kateri bomo napisali kodo, da se bo ob kliku na gumb izdelek dodal v košarico. Preverili bomo, če košarica v Sessionu že obstaja (če ne, jo ustvarimo), ustvarili nov objekt tipa košarica, iz Sessiona prebrali vsebino košarice ter v objekt tipa košarica zapisali te podatke. V naslednjem koraku bomo ta objekt dodali zbirki tipa kosarica in jo zopet shranili nazaj v Session.
V datoteki **Izdelek.aspx.cs** definirajmo metodo **Button1_Click**.

```
protected void Button1_Click(object sender, EventArgs e)
{
 //Ustvarimo novo zbirko tab tipa kosarica.
    List<kosarica> tab = new List<kosarica>();
 //Če spremenljivka kos (košarica) v Sessionu obstaja, vrednost zapišemo v zbirko
tab.
    if (Session["kos"] != null)
    {
//Explicitno (cast) pretvorimo spremenljivko kos iz Sessiona v tip List<kosarica>.
//Če je v spremenljivki kos zapisan kakšen drug podatkovni tip, bo program med
izvajanjem javil napako.
       tab = (List<kosarica>)Session["kos"];
    }
//Nov objekt k tipa košarica
    kosarica k = new kosarica();
//Iz Sessiona iz spremenljivke "id" bomo pridobili id izbranega izdelka
    k.id = Convert.ToInt32(Session["id"].ToString());
//Ostale podatke bomo prebrali iz gradnika DetailsView iz določenih vrstic druge
celice.
    k.znamka = DetailsView1.Rows[0].Cells[1].Text;
    k.tip = DetailsView1.Rows[1].Cells[1].Text;
//Url slike v 7. vrstici. V tej celici s Controls[0] povemo, da želimo prvi
element v tej celici. Določimo, da ima lastnosti tipa Image, le-temu pa lahko
preberemo lastnost ImageUrl.
    k.slika = ((Image)DetailsView1.Rows[6].Cells[1].Controls[0]).ImageUrl;
//Ker smo ceni dodali še enoto €, tu ne gre za čisto število tipa int ali double
ali katerega od drugih številskih podatkovnih tipov, zato z njim ne moremo
računati ali ga spremeniti v kakšen številski podatkovni tip. Ker pa vemo, da je
med številko in enoto € vmes znak presledek,
    string celicaCena = DetailsView1.Rows[7].Cells[1].Text;
//bomo ta string razdelili s presledkom z metodo Split. Kot rezultat dobimo
tabelo, ki ima v vsakem polju del besedila pred presledkom in za presledkom. Torej
bo v prvi celici številska vrednost, v drugi pa vedno znak €.
    string[] tabCena = celicaCena.Split(' ');
//Iz te tabele preberemo vrednost in jo shranimo v spremenljivko cena.
    string cena = tabCena[0];
//Ceno pretvorimo v ustrezen podatkovni tip in jo shranimo v objekt k.
    k.cena = Convert.ToDouble(cena);
//Iz gradnika TextBox preberemo vrednost za količino.
```

```
    k.kolicina = Convert.ToDouble(TextBox1.Text);
//objekt k dodamo zbirki tab
    tab.Add(k);
//v Session v spremenljivko kos shranimo novo vsebino košarice.
    Session["kos"] = tab;
//Preusmerimo nas nazaj na stran Izdelki.aspx.
    Response.Redirect("Izdelki.aspx");
}
```

Če sedaj na vrstico Response.Redirect (…) nastavimo BrakePoint, bomo lahko videli vrednosti spremenljivk in objektov naše košarice med izvajanjem kode. Tako bomo lahko preverili, če smo izbrali pravilne podatke. Zaženimo projekt, izberimo izdelek, določimo količino, nato kliknimo gumb Button. Ko program med izvajanjem pride do točke BrakePoint, v oknu Local Variables lahko vidimo vrednosti vseh lokalnih spremenljivk. Tako lahko preverimo, ali smo iz pravih celic prebrali prave vrednosti.



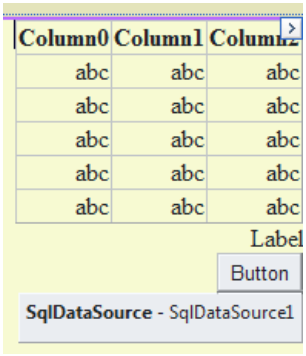
Slika 89: Prikaz stanja spremenljivk v košarici

9.1 Urejanje košarice

Preden oddamo naročilo, moramo dati kupcu možnost, da pregleda izbrane izdelke, jih uredi ali po potrebi tudi izbriše. Ker imamo košarico shranjeno v Sessionu, jo lahko prikažemo kjerkoli v projektu, da jo lahko ustrezno uredimo.
Odprimo datoteko **Kosarica.aspx**.
Ker želimo imeti zaradi urejenosti izpisa vse tabele in gradnike na tej strani poravnane na desno, bomo v Content dodali značko div in lastnost align="right", tako kot vidimo v spodnji kodi.

```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
runat="server">
<div align="right">

</div>
</asp:Content>
```

V nadaljevanju bomo vse gradnike ugnezdili v to značko. Dodajmo gradnike: **GridView, Label, Button in SqlDataSource**. V gradniku GridView bomo prikazali vsebino kupčeve košarice, v gradniku Label bo skupna cena, ob kliku na gumb bomo naročili izdelke in jih s pomočjo SqlDataSource zapisali v bazo. Torej SqlDataSource sedaj ne bo povezan z nobeno tabelo ali gradnikom, ampak mu bomo programsko dodelili vrednosti insert parametrov, nato pa izvedli insert metodo.



Slika 90: Grandik GridView za prikaz košarice

HTML datoteke Kosarica.aspx:

```
<div align="right">
    <asp:GridView ID="GridView1" runat="server">
    </asp:GridView>
    <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
    <br />
    <asp:Button ID="Button1" runat="server" Text="Button" />
    <asp:SqlDataSource ID="SqlDataSource1" runat="server">
    </asp:SqlDataSource>
    <br />
</div>
```

Kadar bo kupec hotel videti vsebino košarice, bo kliknil na povezavo do datoteke Kosarica.aspx. Ko se bo stran začela nalagati (Page_Load), se bo vsebina košarice prepisala iz Sessiona v zbirko List<kosarica>, nato pa ta zbirka določila gradniku GridView za vir oz. DataSource, v gradnik Label pa zapisala skupna cena vseh izdelkov v košarici.
Odprimo datoteko **Kosarica.aspx.cs** ali pa desni klik na obrazec/**View Code**..

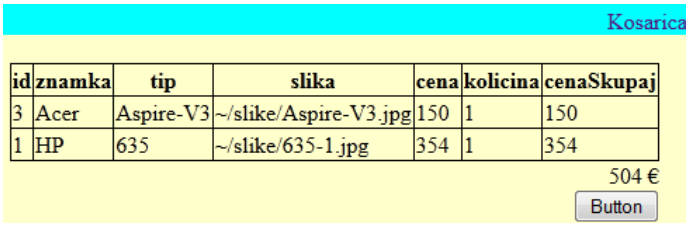
```
protected void Page_Load(object sender, EventArgs e)
{
//Ustvarimo zbirko tab tipa kosarica, vanjo pa bomo zapisali vrednosti iz Session-
a.
    List<kosarica> tab = new List<kosarica>();
    if (Session["kos"] != null)
    {
         tab = (List<kosarica>)Session["kos"];
    }
//Gradniku GridView določimo vir, ki pa je kar Zbirka tipa List<kosarica>.
    GridView1.DataSource = tab;
//Podatke povežemo v gradniku GridView iz programsko določenega vira tab.
    GridView1.DataBind();

//Izračunamo vsoto cene vseh izdelkov skupaj. V tem primeru smo uporabili Lambda
izraze (Lambda Expressions) iz knjižnice System.Linq.
    Label1.Text = tab.Sum(y => y.cenaSkupaj) + " €";
}
```

Za izračun skupne cene bi lahko uporabili tudi klasični način s for ali foreach zanko na sledeči način:

```
double vsota = 0;
for (int i = 0; i < tab.Count; i++)
```

```
{
    double cena =tab[i].cenaSkupaj;
    vsota = vsota + cena;
}
Label1.Text = vsota + " €";
```

Več o poizvedovalnem jeziku LINQ in Lambda Expressions si lahko ogledamo na spletu:
http://www.switchonthecode.com/tutorials/csharp-tutorial-the-lambda-operator
http://code.msdn.microsoft.com/101-LINQ-Samples-3fb9811b
Ključne besede v internetnih iskalnikih: LINQ, Lambda Expressions.
Če zaženemo projekt (F5), dodamo 2 izdelka v košarico, dobimo sledeče:

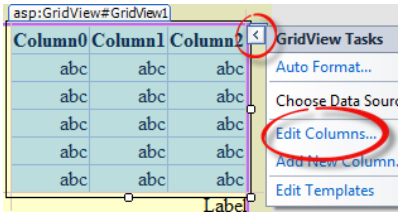


Slika 91: Prikaz košarice v brskalniku

Podatke lahko pustimo tako, kot se prikažejo, vendar zopet namesto lokacije slike želimo prikazati sliko, zato bomo najprej poskrbeli za to. Omeniti gre tudi, da zato, ker smo v **razredu kosarica** definirali **javne lastnosti**, povezane s polji, lahko tudi avtomatsko prikažemo vse te podatke. Če bi v razredu košarica definirali samo javna polja, tega ne bi mogli storiti na tak avtomatski način.

Naslednja zadevo bo, da bomo odstranili polje **id**, **prikazali sliko** in **dodali gumb Delete**, ki bo omogočal brisanje posameznih izdelkov iz košarice.

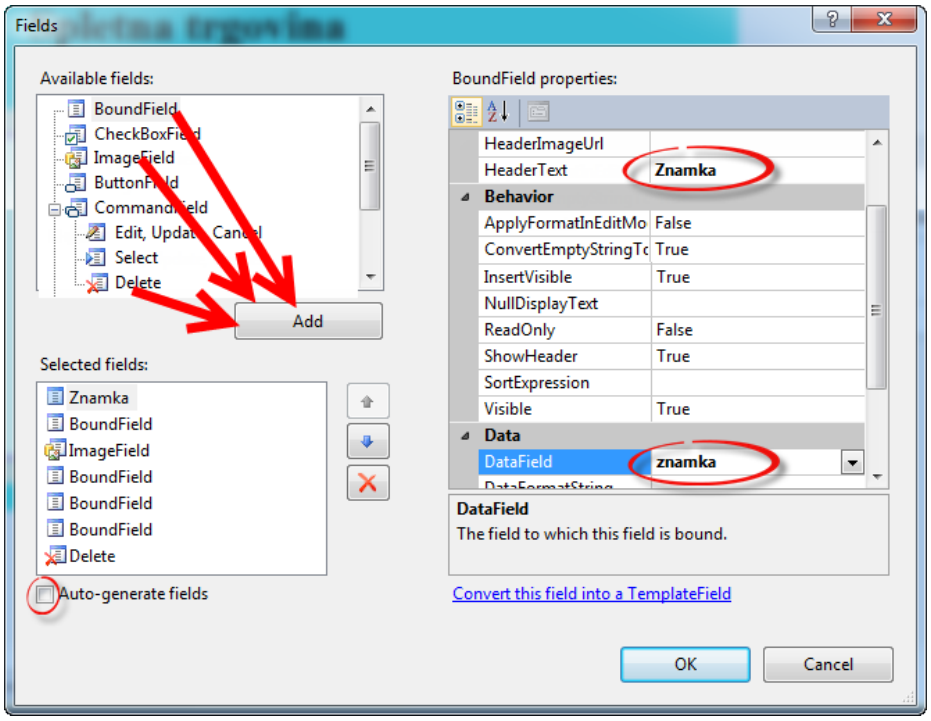
Na gradniku GridView kliknimo na zavihek **GridView task**, nato pa izberimo **Edit Columns**.



Slika 92: Urejanje in oblikovanje stolpcev košarice 1

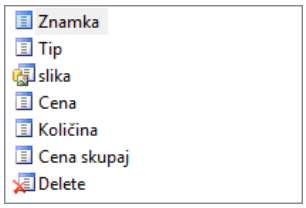
Najprej bomo **izključili** opcijo **Auto-generate fields** tako, kot vidimo levo spodaj na spodnji sliki.
Dodajmo 5 polj tipa **BoundField**, enega **ImageField** ter iz zavihka CommandField dodajmo polje **Delete**. Vrstni red naj bo tak, kot je prikazano na spodnji sliki. Ker smo gradniku GridView programsko določili vir (DataSource), sedaj v polju DataField nimamo spustnega seznama, v katerem bi lahko izbirali, iz katerega polja naj črpa podatke, ampak moramo sami napisati imena. Napisana morajo biti **točno tako, kot so definirana imena lastnosti v razredu kosarica.**
Najprej iz okna Selected Fields označimo prvo polje **BoundField** ter mu v oknu **BoundField properties** določimo vrednost lastnosti **DataField,** in sicer **znamka** (tako kot smo napisali ime

lastnosti v razredu kosarica). Določimo tudi lastnost **HeaderText,** in sicer **Znamka**. To bo napis, ki se bo prikazal na vrhu posamezne kolone nad podatki.
Na isti način sedaj določimo še naslednje polje tip izdelka oz. model (**DataField = tip, HeaderText = Tip**).



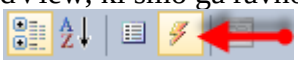
Slika 93: Urejanje in oblikovanje stolpcev košarice2

Polju ImageField določimo podatkovni izvor in velikost prikazane slike (**DataImageUrlField = slika, ControlStyle/Width = 100px, ControlStyle/Heigth = 100px**).
Polju cena določimo (**DataField = cena, DataFormatString = {0} €, HeaderText = Cena**).
Polju količina določimo (**DataField = kolicina, HeaderText = Količina**).
Polju skupnaCena določimo (**DataField = cenaSkupaj, DataFormatString = {0} €, HeaderText = Cena skupaj**).
Če si sedaj ogledamo okno Selected fields, so se vsa polja preimenovala tako, kot smo določili lastnosti HeaderText.



Slika 94: Stolpci košarice

Kliknimo OK.
Določimo še, da se bo izdelek izbrisal iz košarice, ko bomo kliknili gumb Delete v zadnjem polju.

Označimo naš gradnik GridView, ki smo ga ravnokar urejali, in na vrhu okna **Properties** kliknimo na gumb **Events**  ter dvokliknimo na dogodek **RowDeleting**. Proženje dogodkov že poznamo. Javascript dogodek **onrowdeleting** bo sprožil metodo **GridView1_RowDeleting** ob kliku na gumb Button.
HTML koda Gradnika GridView:

```
<div align="right">
    <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
        onrowdeleting="GridView1_RowDeleting">
        <Columns>
            <asp:BoundField DataField="znamka" HeaderText="Znamka" />
            <asp:BoundField DataField="tip" HeaderText="Tip" />
            <asp:ImageField DataImageUrlField="slika">
                <ControlStyle Height="100px" Width="100px" />
            </asp:ImageField>
            <asp:BoundField DataField="cena" DataFormatString="{0} €"
HeaderText="Cena" />
            <asp:BoundField DataField="kolicina" HeaderText="Količina" />
            <asp:BoundField DataField="cenaSkupaj" DataFormatString="{0} €"
                HeaderText="Cena skupaj" >
            </asp:BoundField>
            <asp:CommandField ShowDeleteButton="True" />
        </Columns>
    </asp:GridView>
    <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
    <br />
    <asp:Button ID="Button1" runat="server" Text="Button" />
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"></asp:SqlDataSource>
    <br />
</div>
```

Definirajmo še metodo **GridView1_RowDeleting**. Iz zbirke bomo izbrisali vrstico z določenim indexom, ki ga bomo pridobili iz lastnosti argumenta **e.RowIndex**.

```
protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
    //Preberemo, na kateri vrstici smo kliknili Delete.
    int ind = e.RowIndex;
    List<kosarica> tab = new List<kosarica>();
    if (Session["kos"] != null)
    {
         tab = (List<kosarica>)Session["kos"];
         //Iz zbirke tab odstranimo vrstico oz. izdelek, ki smo ga izbrali.
         tab.RemoveAt(ind);
         GridView1.DataSource = tab;
         GridView1.DataBind();
    }
    //Pokličemo metodo Page_Load, da osvežimo stran.
    Page_Load(sender, e);
}
```

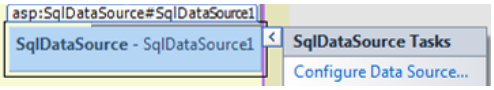
Če želimo malo lepšo obliko gradnika GridView, kliknimo na zavihek **GridView Task** (levo zgoraj), izberimo **Auto Format …** ter eno od želenih oblik, npr. Sand&Sky.

Zaženimo projekt (F5), izberimo nekaj izdelkov ter preizkusimo funkcionalnost gumba Delete. Obenem opazimo tudi, da se pod gradnikom GridView spreminja tudi končna skupna cena, ko brišemo izdelke iz košarice. Ta se namreč izpiše v dogodku Page.Load.

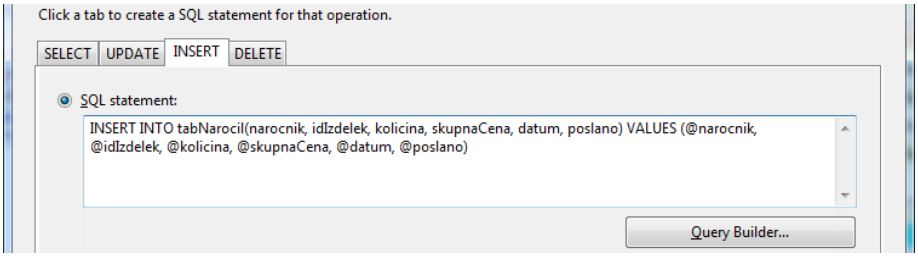
Slika 95: Prikaz oblikovane košarice v brskalniku

9.2 Naročanje izdelkov

Naslednja funkcija naše košarice bo shranjevanje izbranih izdelkov v bazo. Za to bomo uporabili gradnik SqlDataSource, ki mu bomo definirali ustrezen SQL **INSERT** izraz.
Označimo gumb **Button** in mu spremenimo napis. Lastnost **Text = Oddaj naročilo**.
Kliknimo na zavihek SqlDataSource Task in izberimo **Configure Data Source …**



Odpre se čarovnik. Prvi korak je, da določimo **Connection String.** Izberimo **Database1ConnectionString1** ter kliknimo **Next**.
V naslednjem koraku Configure the Select Statment izberimo možnost **Specify a custom statment or stored procedure** ter kliknimo **Next**. Odpre se okno, v katerem lahko napišemo vse izraze: SELECT, UPDATE, INSERT in DELETE. Ker mora biti SELECT izraz vedno definiran, pod zavihkom SELECT zapišimo izraz: **SELECT * FROM tabNarocil**. Sedaj kliknimo na zavihek INSERT ter mu napišimo vrednost:
INSERT INTO tabNarocil(narocnik, idIzdelek, kolicina, skupnaCena, datum, poslano) VALUES (@narocnik, @idIzdelek, @kolicina, @skupnaCena, @datum, @poslano).
Znak @ pomeni, da gre za parameter, ki bo ravnotako definiran v SqlDataSource.



Slika 96: SqlDataSource in INSERT izraz

Lahko pa si pomagamo tudi s čarovnikom za ustvarjanje izrazov (Query Builder …).
Kliknimo **Next**, nato pa **Finish**.
HTML gradnika SqlDataSource:

```
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
        ConnectionString="<%$ ConnectionStrings:Database1ConnectionString1 %>"
        InsertCommand="INSERT INTO tabNarocil(narocnik, idIzdelek, kolicina,
skupnaCena, datum, poslano) VALUES (@narocnik, @idIzdelek, @kolicina, @skupnaCena,
@datum, @poslano)"
        SelectCommand="SELECT * FROM tabNarocil">
        <InsertParameters>
            <asp:Parameter Name="narocnik" />
            <asp:Parameter Name="idIzdelek" />
            <asp:Parameter Name="kolicina" />
            <asp:Parameter Name="skupnaCena" />
            <asp:Parameter Name="datum" />
            <asp:Parameter Name="poslano" />
        </InsertParameters>
    </asp:SqlDataSource>
```

Ko smo definirali INSERT izraz, bomo napisali kodo, da se bodo ob kliku na gumb ustrezni podatki vnesli v bazo.
Dvokliknimo na gumb **Button**.
Odpre se nam datoteka **Kosarica.aspx.cs**, v kateri bomo definirali metodo **Button1_Click**.

```
protected void Button1_Click(object sender, EventArgs e)
{
    //Ustvarimo novo zbirko tab, ki je tipa List<kosarica>.
    List<kosarica> tab = new List<kosarica>();
    //Preverimo, če ni prazna.
    if (Session["kos"] != null)
    {
        //Potem prepišemo košarico iz Sessiona v zbirko tab.
        tab = (List<kosarica>)Session["kos"];
        //Shranimo trenutni datum in čas. Lahko se zgodi, da moramo datum in čas
zapisati v točno določeni obliki.
        string dat = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");
        //Zanka, ki se izvede tolikokrat, kolikor je naročil v zbirki tab.
        for (int i = 0; i < tab.Count; i++)
        {
            //Vnesemo Insert parametre v SqlDataSource.
            //V naročnika shranimo prijavljeno uporabniško ime.
            SqlDataSource1.InsertParameters["narocnik"].DefaultValue =
User.Identity.Name;
            SqlDataSource1.InsertParameters["idIzdelek"].DefaultValue =
tab[i].id.ToString();
            //Kadar bomo podajali parametre, je decimalno ločilo znak pika. V C#
pa pri pretvarjanju v string dobimo decimalno ločilo vejico, zato moramo v
številu vejico zamenjati s piko, drugače dobimo SqlExeption.
            SqlDataSource1.InsertParameters["kolicina"].DefaultValue =
tab[i].kolicina.ToString().Replace(',','.');
            SqlDataSource1.InsertParameters["skupnaCena"].DefaultValue =
tab[i].cenaSkupaj.ToString().Replace(',', '.');
            SqlDataSource1.InsertParameters["datum"].DefaultValue = dat;
            SqlDataSource1.InsertParameters["poslano"].DefaultValue = "FALSE";
            //Zapis se shrani v bazo.
```

```
            SqlDataSource1.Insert();
        }
        //Izpraznimo zbirko tab.
        tab = new List<kosarica>();
        //Spremenljivko "kos" v Sessionu postavimo na vrednost null, kar pomeni,
da ta ne obstaja več
        Session["kos"] = null;
    }
    //Osvežimo stran s klicem metode Page_Load()
    Page_Load(sender, e);
}
```

Zaženimo projekt z (F5), nekaj izdelkov dodajmo v košarico ter preizkusimo gumb Delete. Če brišemo izdelke, se ustrezno zmanjšuje tudi končna skupna cena. Oddajmo naročilo in v bazi poglejmo, ali so se vrednosti vnesle v tabelo tabIzdelkov v bazi.

Slika 97: Oddajanje naročila v košarici v brskalniku

Ko oddamo naročilo, se vsi izdelki izbrišejo, skupna cena pa se postavi na 0 €, ker se košarica izprazni. Poglejmo si še zapis v bazi. V oknu Database Explorer desni klik na tabelo tabNarocil/**Show Table Data**.

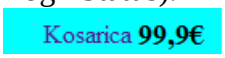
	id	narocnik	idIzdelek	kolicina	skupnaCena	datum	poslano
▶	16	MYPC\andrej	4	2	199,8	6.8.2012 14:12:22	False
	17	MYPC\andrej	8	3	1350	6.8.2012 14:12:22	False
	18	MYPC\andrej	3	1	150	6.8.2012 14:12:22	False
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Slika 98: Zapis v tabeli tabNarocil v bazi

Na zgornji sliki opazimo, da so vsi podatki, ki smo jih vnesli pravilno zapisani. Zaenkrat je edina neznanka naročnik, ki jo pridobimo iz vrstice: **`User.Identity.Name;`**. Trenutno deluje aplikacija tako, da vzame za uporabnika tistega, ki je prijavljen v sistem Windows. Torej je v mojem primeru Ime računalnika **MYPC**, uporabnik pa **andrej**. Kasneje bomo pri varnosti nastavili, da bo uporabnik tisti, ki se bo prijavil v prijavnem obrazcu.

Naredimo še, da se bo ob dodajanju novega izdelka v košarico zraven povezave košarica izpisala skupna cena, ki je trenutno nabrana v košarici. Tako bomo na kateri koli strani vedno videli, za kakšno vrednost imamo nabranih izdelkov.

Odprimo datoteko **Site1.Master** ter zraven gradnika **Menu2** dodajmo gradnik **Label** ter mu določimo **Font/Size = Large, Font/Bold = True** in **Text = 0 €.** V značko **div.login** bomo vstavili **tabelo z eno vrstico in štirimi celicami** (2 bomo potrebovali še kasneje za Login in LoginStatus).



HTML koda div.login:

```
            <div class="login">
            <table>
            <tr>
                <td>
                <asp:Menu ID="Menu2" runat="server">
                    <Items>
                        <asp:MenuItem NavigateUrl="~/Kosarica.aspx"
Text="Košarica" Value="Košarica"></asp:MenuItem>
                    </Items>
                </asp:Menu>
                </td>
                <td>
                <asp:Label ID="Label1" runat="server" Text="0 €" Font-Bold="True"
Font-Size="Large"></asp:Label>
                </td>
                <td></td>
                <td></td>
            </tr>
            </table>
            </div>
```

Sedaj bomo poskrbeli, da se vsakič na zadnjem koraku, preden se dokončno izgradi HTML koda na strežniku, izračuna skupna cena in se zapiše v gradnik Label.
Kadar imamo MasterPage in ContentPage, se dogodki prožijo po tem vrstnem redu:

1. Page_Load v MasterPage
2. Vsi drugi dogodki, kot so Button_Click … v MasterPage
3. Page_Load v ContentPage (posamezna podstran ali obrazec)
4. Vsi drugi dogodki, kot so Button_Click … v ContentPage
5. Page_PreRender v ContentPage
6. Page_PreRender v MasterPage

Če bi spodnjo kodo zapisali v Page_Load, bi se cena osvežila šele naslednjič, ko bi osvežili stran. Vedno en korak nazaj. Zato jo moramo izvesti čisto na koncu, ko se izdelek že doda v košarico. V košarico pa se doda šele v dogodku Button_Click iz ContentPage-a.
Odprimo datoteko Site1.Master.cs. Metoda Page_Load je že privzeto napisana, zraven pa napišimo še metodo **Page_PreRedner**, ki izračuna in izpiše skupno ceno v gradnik Label.

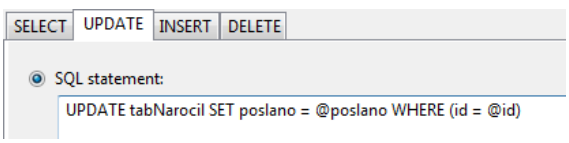
```
protected void Page_Load(object sender, EventArgs e)
{

}

protected void Page_PreRender(object sender, EventArgs e)
{
    Label1.Text = "0 €";
    List<kosarica> tab = new List<kosarica>();
    if (Session["kos"] != null)
    {
         tab = (List<kosarica>)Session["kos"];
         double skupaj = tab.Sum(y => y.cenaSkupaj);
         Label1.Text = skupaj + " €";
    }
}
```

Zaženimo projekt in v košarico dodajajmo izdelke. Ob tem opazimo, da se zraven povezave košarica spreminja tudi cena, ki je trenutno v košarici.

10 Pregled naročenih izdelkov

Sedaj, ko imamo shranjene naročene izdelke v bazi, moramo še poskrbeti, da lahko nekdo pregleduje naročene izdelke in jih pošilja kupcu. Pošiljatelja zanimajo samo ne poslana naročila, zato bomo z WHERE pogojem že takoj izločili poslana.
V Solution Explorerju iz mape **admin** odprimo datoteko **Narocila.aspx**. Na obrazec dodajmo gradnika **GridView** in **SqlDataSource**. Najprej bomo nastavili SqlDataSource, in sicer SELECT in UPDATE izraz.
Kliknimo na zavihek **SqlDataSource Task** in izberimo **Configure Data Source …**
Izberimo **ConnectionString,** in sicer **Database1ConnectionString1** in **Next.**
V koraku **Specify a Select statment** izberimo **Specify a custom SQL statment or stored procedure** in kliknimo **Next.**
Izbran naj bo zavihek SELECT ter vanj zapišimo:
SELECT tabNarocil.id, tabNarocil.narocnik, tabNarocil.idIzdelek, tabIzdelkov.znamka, tabIzdelkov.tip, tabNarocil.kolicina, tabNarocil.skupnaCena, tabNarocil.datum, tabNarocil.poslano FROM tabIzdelkov INNER JOIN tabNarocil ON tabIzdelkov.id = tabNarocil.idIzdelek WHERE (tabNarocil.poslano = 'false')

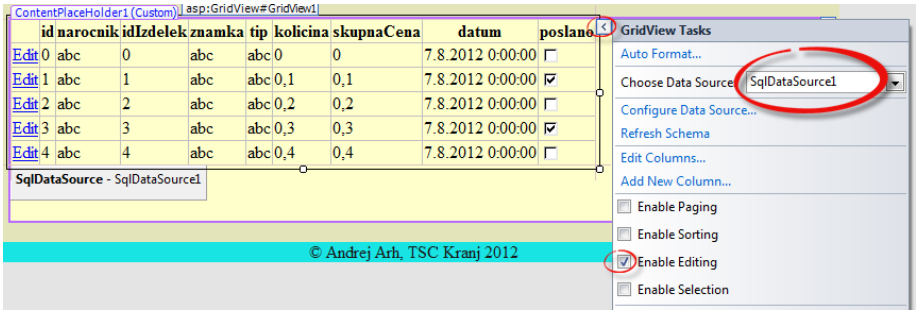
Izberimo zavihek UPDATE in vanj zapišimo:
UPDATE tabNarocil SET poslano = @poslano WHERE (id = @id)



Slika 99: SqlDataSource in UPDATE izraz

S tem UPDATE izrazom spremenimo lahko samo polje **poslano** za določen **id** in nič drugega.
Kliknimo **Next**, nato **Finish**.

Kliknimo na zavihek **GridView Task**, določimo lastnost **Choose Data Source = SqlDataSource,** označimo možnost **Enable Editing**, v gradniku GridView pa se kot prva kolona prikaže gumb Edit.



Slika 100: GridView in Edit gumb

Ker bomo zraven cene rabili še znak **€** in pri količini znak **x**, kliknimo na **Edit Columns …** Polju **kolicina** spremenimo lastnost **DataFormatString = x {0}**, polju skupnaCena pa **DataFormatString = {0} €**. Kliknimo **OK**.
Lahko mu spremenimo še obliko **Auto Format …** in npr: **Rainy Day**.

Zaženimo projekt (F5) in kliknimo na povezavo naročila.

	id	narocnik	idIzdelek	znamka	tip	kolicina	skupnaCena	datum	poslano
Edit	16	MYPC\andrej	4	Lenovo	B570	x 2	199,8 €	6.8.2012 14:12:22	☐
Edit	17	MYPC\andrej	8	DELL	xps_15	x 3	1350 €	6.8.2012 14:12:22	☐
Edit	18	MYPC\andrej	3	Acer	Aspire-V3	x 1	150 €	6.8.2012 14:12:22	☐

Slika 101: Prikaz naročil v brskalniku

V brskalniku se nam pojavi tabela naročil. Če kliknemo gumb **Edit**, se nam ponudi možnost, da **uredimo polja**. Ker smo SqlDataSouce pri UPDATE izrazu definirali, da se posodobi samo polje poslano, za vse ostale spremembe nima učinka.
Recimo, da želimo drugi izdelek označiti kot poslanega, kliknimo Edit, nato z miško označimo kljukico v koloni poslano ter kliknemo Update na levi stran.

	id	narocnik	idIzdelek	znamka	tip	kolicina	skupnaCena	datum	poslano
Edit	16	MYPC\andrej	4	Lenovo	B570	x 2	199,8 €	6.8.2012 14:12:22	☐
Update Cancel	17	MYPC\andrej	8	DELL	xps_15	3	1350	6.8.2012 14:12:22	☑
Edit	18	MYPC\andrej	3	Acer	Aspire-V3	x 1	150 €	6.8.2012 14:12:22	☐

Slika 102: Prikaz urejanja naročil (poslano)

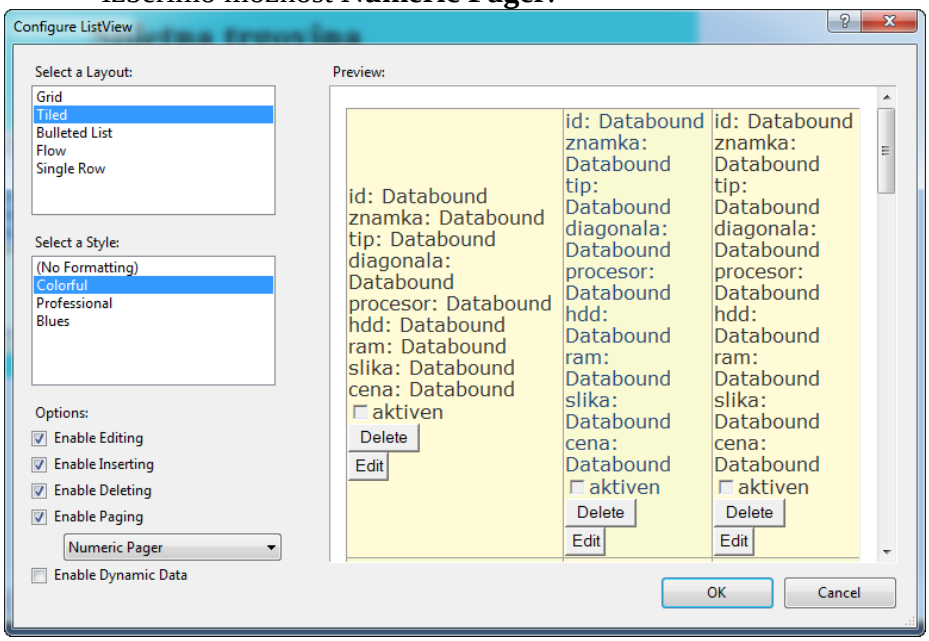
Zapis je še vedno v bazi, le prikazan ne bo več.

11 Administracija z izdelki

Ena od pomembnih stvari je administracija z izdelki. Nekdo od upravljalcev spletne trgovine mora vzdrževati izdelke v bazi, spreminjati cene, imena ter dodajati nove preko obrazca itd. Ta problem bomo rešili z gradnikoma ListView in SqlDataSource.

Iz Solution Explorerja iz **mape admin** odprimo datoteko **Admin.aspx**. Odpre se nam prazen obrazec, v katerega bomo z miško iz Database Explorerja **povlekli tabelo tabIzdelkov**.
To smo storili zato, da se nam ne bo potrebno ukvarjati s SqlDataSource-om, ampak imamo v njem že definirane vse izraze: SELECT, INSERT, UPDATE in DELETE, ki so prilagojeni na tabelo tabIzdelkov. Privzeto se nanj postavi tudi gradnik **GridView**, ki ga takoj **izbrišemo**, ter iz orodjarne Toolbox izberemo gradnik **ListView in ga povlečemo na obrazec**. Kliknimo na zavihek **ListView Task** in mu določimo **DataSource = SqlDataSource1**, nato pa izberimo **Configure ListView**.
Spremenimo nekaj nastavitev:

- Select a Layout: **Tiled.**
- Select a Style: **Colorful.**
- Označimo opcije: **Enable Editing, Inserting, Deleting in Paging.**
- Izberimo možnost **Numeric Pager**.

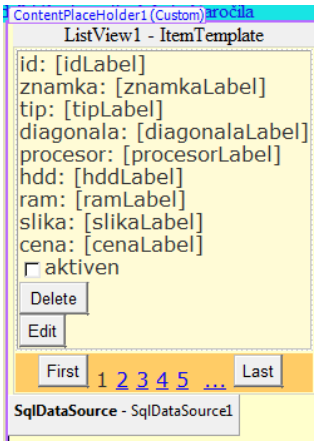


Slika 103: Nastavitev oblike gradnika ListView

Kliknimo **OK**.
Sedaj pa odprimo HTML stran oz. preklopimo na **Source** ter iz HTML-ja **izbrišimo** nepotrebne dele gradnika ListView. Izbrišimo značke:

- `<AlternatingItemTemplate>…</AlternatingItemTemplate>`
- `<EmptyDataTemplate>…</EmptyDataTemplate>`
- `<SelectedItemTemplate>…</SelectedItemTemplate>`

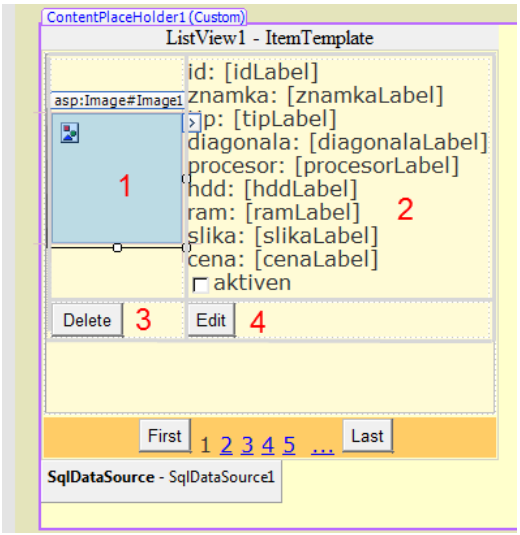
V pogledu Design kliknimo na zavihek **ListView Task** in izberimo pogled **Current View = ItemTemplate**. To je pogled posamezne celice, v kateri je izdelek. Tu lahko podatke razvrstimo in oblikujemo, kakor želimo. Osnovni pogled je tak, kot ga vidimo na spodnji sliki.



Slika 104: ListView in ItemTemplate

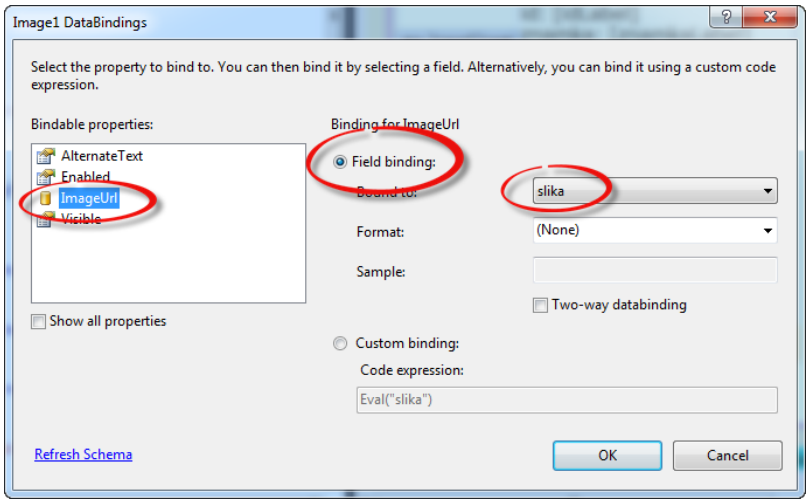
Ker pa bi radi malo spremenili pozicije podatkov gumbov in slike, bomo vnesli tabelo velikosti 2x2, s katero bomo pozicionirali gradnike.
Kliknimo pred besedilo **id:** in v menijski vrstici kliknemo **Table/Insert Table**. Odpre se nam okno, v katerem določimo lastnosti tabele. Važno je, da imamo **2 stolpca in 2 vrstici**. Kliknimo **OK**.

Sedaj iz orodjarne ToolBox v **prvo** celico dodajmo gradnik **Image**, v **drugo** premaknimo vse **podatke**, v **tretjo** gumb **Delete** in v **četrto** gumb **Edit**, tako kot vidimo na spodnji sliki.



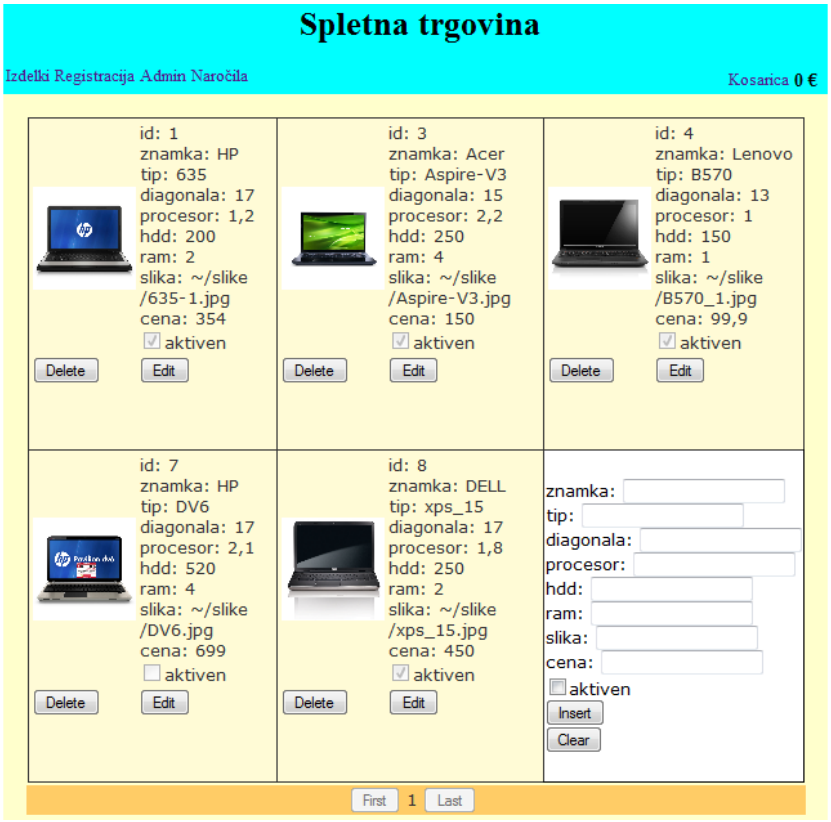
Slika 105: Oblikovanje ItemTemplate

Sedaj označimo gradnik **Image** in kliknimo na zavihek **Image Task,** nato pa **Edit Data Bindings**. Tu izberemo, iz kje se bo naložila slika. Izberimo **ImageUrl**, možnost **Field binding** ter iz spustnega seznama izberimo vir: **slika,** tako kot je prikazano na spodnji sliki. Kliknimo **OK**.



Slika 106: ListView in določanje vira sliki

Zaženimo projekt (F5) in kliknimo na povezavo **Admin**. Podatke, slike in gumbe smo dobili razvrščene ravno tako, kot smo hoteli.



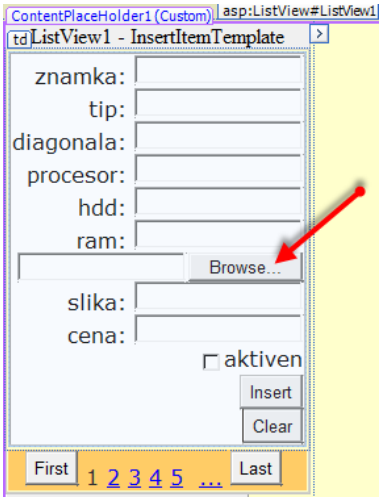
Slika 107: Prikaz urejenih izdelkov z gradnikom ListView v brskalniku

Kot opazimo, zadnja celica vedno omogoča nov vnos. Ta obrazec nam omogoča vnos vseh podatkov, le slike ne moremo prenesti na strežnik in moramo ročno vpisati pot do slike na strežniku. Bomo vedno v okno slika pisali pot do slike? Kaj pa če bi se slike začnejo podvajati ali prepisovati? Katero sliko bo prikazal kje? Pa rešimo to težavo na elegantnejši način. V Okno **Insert Template** bomo dodali nov gradnik za nalaganje datotek **FileUpload**. Ko bo uporabnik, ki

administrira z izdelki izbral sliko, bo program v ozadju preveril, ali slika ni prevelika, jo shranil pod svojim imenom (datum in čas), nato pa jo shranil v projekt na strežniku.

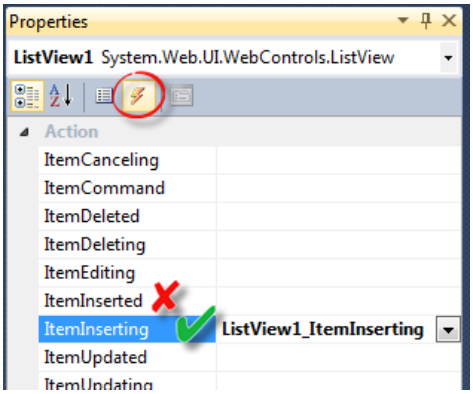
Gradniku ListView kliknimo na **ListView Task** in izberimo pogled **InsertItemTemplate**. Iz Orodjarne izberimo gradnik **FileUpload** ter ga vrinimo **med** polje **ram in slika**. Naredimo še en lepotni popravek in v HTML kodi pod značko **InserItemTemplate** v značko **td** dodajmo lastnost **align="right"**. To pomeni, da bo vse poravnano na levo.

```
<InsertItemTemplate>
    <td runat="server" style="" align="right">
```



Slika 108: InsertItemTemplate in FileUpload

Naredimo sedaj ta obrazec tako, da bo uporabnik na administratorski strani samo izbral sliko, ob kliku na gumb pa se bodo vsi podatki shranili v bazo, slika pa prenesla v mapo Slike na strežniku. Kliknimo na gradnik **ListView** tako, da v oknu **Properties** izberemo gumb, ki prikaže **dogodke**.



Slika 109: Dogodek ItemInserting in metoda ListView1_ItemInserting

Dvakrat kliknimo na dogodek **ItemInserting** in **ne ItemInserted**!!! V datoteki **Admin.aspx.cs** se generira metoda **ListView1_ItemInserting**, ki jo sproži dogodek **ItemInserting**.
Preden začnemo delati in upravljati z datotekami, moramo **vključiti knjižnico:System.IO**;

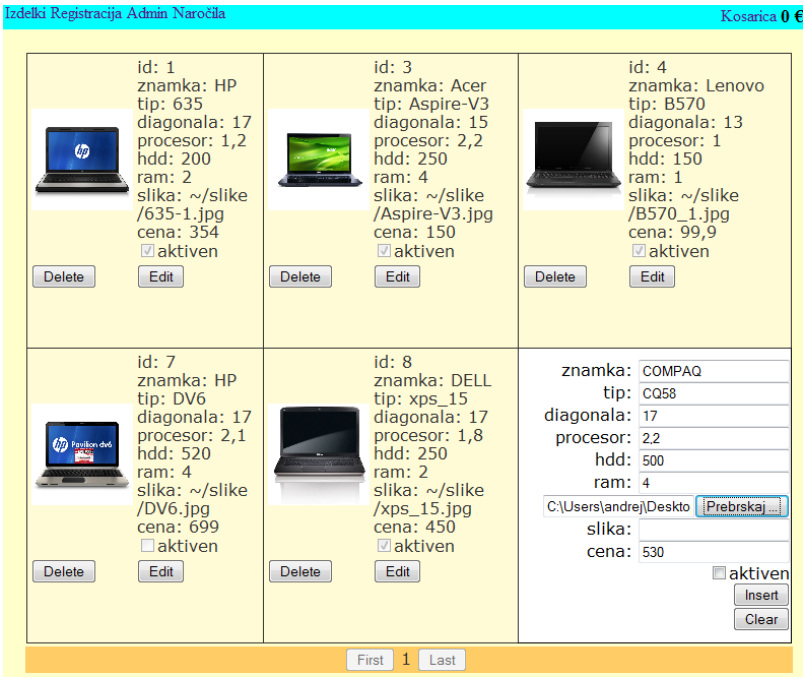
```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;
```

Še koda metode ListView1_ItemInserting:

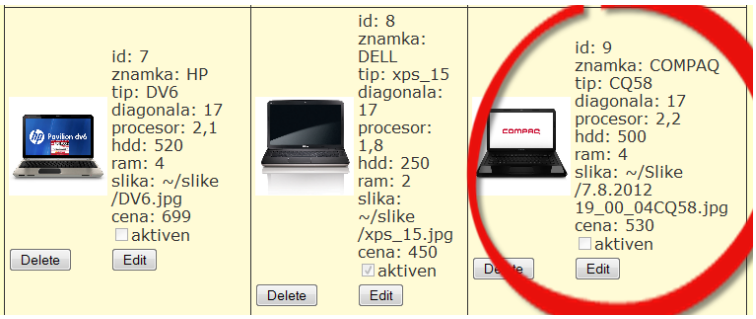
```
protected void ListView1_ItemInserting(object sender, ListViewInsertEventArgs e)
{
//V objekt fu1 prekopirano celoten gradnik FileUpload, ki ima ID="FileUpload1".
//To pomeni, da je sedaj fu1 kopija gradnika FileUpload1, zato ima enake vrednosti
in lastnosti. Tako bomo lahko iz njega pridobili vse potrebne podatke in vsebino
za shranjevanje datoteke oz. slike.
    var fu1 = ((FileUpload)e.Item.FindControl("FileUpload1"));
    //zaščitimo se s tem, da preverimo, če je uporabnik res izbral datoteko.
    if (fu1.HasFile)
    {
        //Omejimo velikost datoteke na 1MB. Če je večja, je ne sprejme.
        if (fu1.FileBytes.Length < 1000000)
        {
            //Varovalni blok, v primeru da pride do napake.
            try
            {
                //Iz naložene datoteke pridobimo njeno ime.
                string imeDatoteke = Path.GetFileName(fu1.FileName);
       //Korenska mapa in podmapa Slike, v katero bo aplikacija shranjevala slike.
                string mapa = Server.MapPath(@"~/Slike/");
                //Če ta mapa še ne obstaja, jo ustvari.
                if (!Directory.Exists(mapa))
                {
                    Directory.CreateDirectory(mapa);
                }
    //Dodaten del imena pri sliki za primer, če bosta sliki z istim imenom, zraven
                //dodamo še datum in čas, pa se ime ne more podvojiti.
                string s = DateTime.Now.ToString().Replace(":", "_");
                //Sliko shranimo v podmapo Slike na strežniku.
                fu1.SaveAs(mapa + s + imeDatoteke);
            //Preko Bind("slika") dodelimo vrednost, in sicer pot ter ime slike,
ki smo jo shranili.
                e.Values["slika"] = @"~/Slike/" + s + imeDatoteke;
            }
            catch(Exception ex)
            {
                //če je prišlo do napake, lahko izpišemo, za katero napako gre
npr:
                //Label2.Text = "Dodajanje ni uspelo<br />" + ex.Message;
                 Response.Write(ex.Message);
            }
        }
    }
}
```

Ko se izvede metoda **ListView1_ItemInserting**, se takoj za tem podatki iz TextBoxov zapišejo v bazo, nato pa se lahko sproži dogodek ItemInserted.
Preizkusimo vstavljanje novega izdelka. Zaženimo projekt (F5), kliknimo na **povezavo Admin** ter poizkusimo vstaviti nov izdelek.



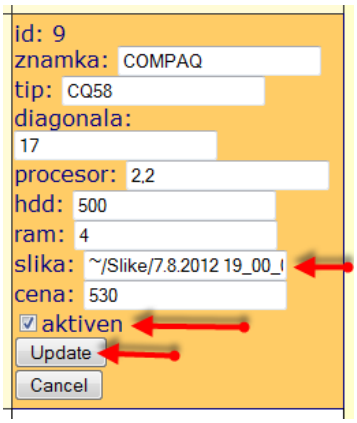
Slika 110: Vstavljanje novega izdelka v brskalniku

Ko klinemo gumb Insert, se izdelek doda v bazo. Kot rezultat pa je prikazan nov izdelek.



Slika 111: Prikaz novega izdelka v brskalniku

Ker smo opazili, da mu nismo označili aktivnosti, lahko kliknemo gumb Edit. Na spodnji sliki sedaj lahko popravimo karkoli vrednost in ob kliku na gumb Update se bodo podatki v bazi posodobili. Zopet imamo tu problem s sliko, zato bomo na podoben način kot pri vstavljanju tudi tukaj na **ItemUpdating** s pomočjo gradnika **FileUpoad** spremenili sliko.
Označimo polje aktiven, kliknimo Update in polje aktiven je dobilo kljukico. Preverimo še, če se je izdelek prikazal na strani Izdelki.aspx.

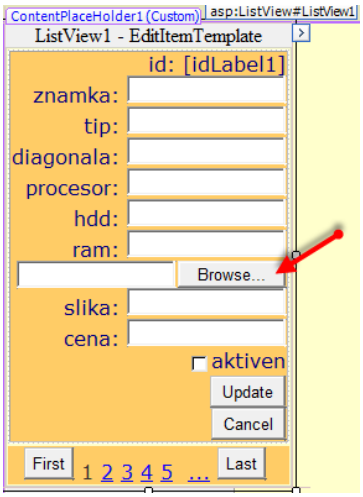


Slika 112: EditItemTemplate

Podobno kot pri pogledu InsertItem bomo tudi tu naredili desno poravnavo. V HTML kodi znački **EditItemTemplate** znački **td** dodajmo lastnost **align="right"**.

```
        <EditItemTemplate>
            <td runat="server" align="right" style="background-color:
#FFCC66;color: #000080;">
                id:
                <asp:Label...
```

Zopet **vrinimo gradnik FileUpload med ram in sliko**.



Slika 113: EditItemTemplate in FileUpload

Kliknimo na gradnik **ListView/Properties** in gumb za prikaz dogodkov. Dvokliknimo na dogodek **ItemUpdating** in **ne ItemUpdated**.
Generira se nam metoda **ListView1_ItemUpdating**, v katero zapišemo kodo, ki shrani in spremeni sliko v bazi na podoben način kot pri vstavljanju.

```
protected void ListView1_ItemUpdating(object sender, ListViewUpdateEventArgs e)
{
//Preberemo index celice, ki jo urejamo.
int ind = e.ItemIndex;
var fu2 = ((FileUpload)ListView1.Items[ind].FindControl("FileUpload2"));
    if (fu2.HasFile)
    {
        if (fu2.FileBytes.Length < 1000000)
```

```
        {
            try //Varovalni blog (v primeru, da pride do napake).
            {
                //Pridobimo ime datoteke, ki smo jo naložili.
                string imeDatoteke = Path.GetFileName(fu2.FileName);
                //Pridobimo celotno pot do mape, v kateri bomo hranili sliko.
                //Pot do slike pa je zapisana v bazi.
                string mapa = Server.MapPath(@"~/Slike/");
                if (!Directory.Exists(mapa))
                {   //Če ta mapa še ne obstaja, jo ustvarimo.
                    Directory.CreateDirectory(mapa);
                }
//Pot serverja in ime slike v bazi, vendar je ena mapa preveč, zato iz baze
vzamemo
//Samo ime slike brez mape.
                string staraSlika =e.OldValues["slika"].ToString();
                //Mapo in sliko razdelimo tam, kjer je znak '/'.
                string[] ss = staraSlika.Split('/');
                //Zadnji del je ime slike.
                staraSlika = ss[ss.Length - 1];
                //Staro sliko izbrišemo iz strežnika.
                File.Delete(mapa + staraSlika);
                //Vsaki sliki bomo pred ime dodali trenutni datum in čas, zato
                //da ne bomo kakšne slike z istim imenom prepisali (povozili)!!!
                //Nato vsa dvopičja nadomestimo s podčrtajem, ker ime slike ne sme
vsebovati znaka ':'.
                string s = DateTime.Now.ToString().Replace(":", "_");
                //Sliko shranimo v mapo ~/Slike/
                fu2.SaveAs(mapa + s + imeDatoteke);
                // Preko Bind("slika") dodelimo vrednost, in sicer pot ter ime
slike, ki smo jo shranili v mapo Slike.
                e.NewValues["slika"] = @"~/Slike/" + s + imeDatoteke;
            }
            catch (Exception ex)
            { //če je prišlo do napake, izpišemo, za katero napako gre
                //Label2.Text = "Popravljanje ni uspelo<br />" + ex.Message;
                //ali na vrhu okna: Response.Write(ex.Message);
            }
        }
    }
}
```

Zaženimo projekt in poizkusimo zamenjati zadnje podatke in sliko z novimi vrednostmi in kliknimo Update. Sedaj se mora zamenjati tudi slika, stara slika pa se izbriše iz mape Slike oz. aplikacije.

Ostane nam samo še gumb Delete, ki bo ravno tako kot Update izbrisal staro sliko iz strežnika iz mape Slike.
Zadeva je zelo podobna kot prej. Označimo ListView, prikažemo vse dogodke in dvokliknemo na dogodek **ItemDeleting**. Koda, ki jo moramo zapisati v **ListView1_ItemDeleting**:

```
protected void ListView1_ItemDeleting(object sender, ListViewDeleteEventArgs e)
{
    string mapa = Server.MapPath(@"~/Slike/");
//Ker tukaj ne moremo preko argumenta e pridobiti vrednosti, kjer je zapisana
slika, jo lahko dobimo na malo drugačen način.
//Iz izbrisanega indexa celice poiščemo kontrolo z ID="slikaLabel" jo castamo kot
Label in lastnost Text preberemo iz nje.
```

```
    string slika =
    ((Label)ListView1.Items[e.ItemIndex].FindControl("slikaLabel")).Text;
//Mapo in sliko razdelimo tam, kjer je znak '/'
    string[] ss = slika.Split('/');
    //Zadnji del je ime slike.
    slika = ss[ss.Length - 1];
    //Sliko izbrišemo iz podmape Slike iz strežnika.
    File.Delete(mapa + slika);
}
```

Zaženimo projekt (F5) ter preizkusimo vse Admin možnosti. Sedaj deluje Insert, Edit (tako da izbriše staro sliko) in Delete (ravno tako izbriše fizično sliko iz mape Slike).

HTML koda gradnikov ListView in SqlDataSource:

```
    <asp:ListView ID="ListView1" runat="server" DataSourceID="SqlDataSource1"
        DataKeyNames="id" GroupItemCount="3" InsertItemPosition="LastItem"
        oniteminserting="ListView1_ItemInserting"
        onitemupdating="ListView1_ItemUpdating"
        onitemdeleting="ListView1_ItemDeleting">
        <EditItemTemplate>
            <td runat="server" align="right" style="background-color:
#FFCC66;color: #000080;">
                id:
                <asp:Label ID="idLabel1" runat="server" Text='<%# Eval("id") %>'
/>
                <br />znamka:
                <asp:TextBox ID="znamkaTextBox" runat="server" Text='<%#
Bind("znamka") %>' />
                <br />tip:
                <asp:TextBox ID="tipTextBox" runat="server" Text='<%# Bind("tip")
%>' />
                <br />diagonala:
                <asp:TextBox ID="diagonalaTextBox" runat="server"
                    Text='<%# Bind("diagonala") %>' />
                <br />procesor:
                <asp:TextBox ID="procesorTextBox" runat="server"
                    Text='<%# Bind("procesor") %>' />
                <br />hdd:
                <asp:TextBox ID="hddTextBox" runat="server" Text='<%# Bind("hdd")
%>' />
                <br />ram:
                <asp:TextBox ID="ramTextBox" runat="server" Text='<%# Bind("ram")
%>' />
                <br />
                <asp:FileUpload ID="FileUpload2" runat="server" />
                <br />
                slika:
                <asp:TextBox ID="slikaTextBox" runat="server" Text='<%#
Bind("slika") %>' />
                <br />cena:
                <asp:TextBox ID="cenaTextBox" runat="server" Text='<%#
Bind("cena") %>' />
                <br />
                <asp:CheckBox ID="aktivenCheckBox" runat="server"
                    Checked='<%# Bind("aktiven") %>' Text="aktiven" />
                <br />
                <asp:Button ID="UpdateButton" runat="server" CommandName="Update"
                    Text="Update" />
```

```
                <br />
                <asp:Button ID="CancelButton" runat="server" CommandName="Cancel"
                    Text="Cancel" />
                <br />
            </td>
        </EditItemTemplate>

        <EmptyItemTemplate>
<td runat="server" />
        </EmptyItemTemplate>
        <GroupTemplate>
            <tr ID="itemPlaceholderContainer" runat="server">
                <td ID="itemPlaceholder" runat="server">
                </td>
            </tr>
        </GroupTemplate>
        <InsertItemTemplate>
            <td runat="server" style="" align="right">
                znamka:
                <asp:TextBox ID="znamkaTextBox" runat="server" Text='<%#
Bind("znamka") %>' />
                <br />tip:
                <asp:TextBox ID="tipTextBox" runat="server" Text='<%# Bind("tip")
%>' />
                <br />diagonala:
                <asp:TextBox ID="diagonalaTextBox" runat="server"
                    Text='<%# Bind("diagonala") %>' />
                <br />procesor:
                <asp:TextBox ID="procesorTextBox" runat="server"
                    Text='<%# Bind("procesor") %>' />
                <br />hdd:
                <asp:TextBox ID="hddTextBox" runat="server" Text='<%# Bind("hdd")
%>' />
                <br />ram:
                <asp:TextBox ID="ramTextBox" runat="server" Text='<%# Bind("ram")
%>' />
                <br />
                <asp:FileUpload ID="FileUpload1" runat="server" />
                <br />slika:
                <asp:TextBox ID="slikaTextBox" runat="server" Text='<%#
Bind("slika") %>' />
                <br />cena:
                <asp:TextBox ID="cenaTextBox" runat="server" Text='<%#
Bind("cena") %>' />
                <br />
                <asp:CheckBox ID="aktivenCheckBox" runat="server"
                    Checked='<%# Bind("aktiven") %>' Text="aktiven" />
                <br />
                <asp:Button ID="InsertButton" runat="server" CommandName="Insert"
                    Text="Insert" />
                <br />
                <asp:Button ID="CancelButton" runat="server" CommandName="Cancel"
                    Text="Clear" />
                <br />
            </td>
        </InsertItemTemplate>
        <ItemTemplate>
            <td runat="server" style="background-color: #FFFBD6;color: #333333;">
                <table class="style1">
                    <tr>
                        <td>
```

```
                            <asp:Image ID="Image1" runat="server" Height="100px"
Width="100px"
                                ImageUrl='<%# Eval("slika") %>' />
                        </td>
                        <td>
                            id:
                            <asp:Label ID="idLabel" runat="server" Text='<%#
Eval("id") %>'></asp:Label>
                            <br />
                            znamka:
                            <asp:Label ID="znamkaLabel" runat="server" Text='<%#
Eval("znamka") %>'></asp:Label>
                            <br />
                            tip:
                            <asp:Label ID="tipLabel" runat="server" Text='<%#
Eval("tip") %>'></asp:Label>
                            <br />
                            diagonala:
                            <asp:Label ID="diagonalaLabel" runat="server"
Text='<%# Eval("diagonala") %>'></asp:Label>
                            <br />
                            procesor:
                            <asp:Label ID="procesorLabel" runat="server" Text='<%#
Eval("procesor") %>'></asp:Label>
                            <br />
                            hdd:
                            <asp:Label ID="hddLabel" runat="server" Text='<%#
Eval("hdd") %>'></asp:Label>
                            <br />
                            ram:
                            <asp:Label ID="ramLabel" runat="server" Text='<%#
Eval("ram") %>'></asp:Label>
                            <br />
                            slika:
                            <asp:Label ID="slikaLabel" runat="server" Text='<%#
Eval("slika") %>'></asp:Label>
                            <br />
                            cena:
                            <asp:Label ID="cenaLabel" runat="server" Text='<%#
Eval("cena") %>'></asp:Label>
                            <br />
                            <asp:CheckBox ID="aktivenCheckBox" runat="server"
                                Checked='<%# Eval("aktiven") %>' Enabled="False"
Text="aktiven" />
                        </td>
                    </tr>
                    <tr>
                        <td>
                            <asp:Button ID="DeleteButton" runat="server"
CommandName="Delete"
                                Text="Delete" />
                        </td>
                        <td>
                            <asp:Button ID="EditButton" runat="server"
CommandName="Edit" Text="Edit" />
                        </td>
                    </tr>
                </table>

            </td>
        </ItemTemplate>
```

```
        <LayoutTemplate>
            <table runat="server">
                <tr runat="server">
                    <td runat="server">
                        <table ID="groupPlaceholderContainer" runat="server"
border="1"
                            style="background-color: #FFFFFF;border-collapse:
collapse;border-color: #999999;border-style:none;border-width:1px;font-family:
Verdana, Arial, Helvetica, sans-serif;">
                            <tr ID="groupPlaceholder" runat="server">
                            </tr>
                        </table>
                    </td>
                </tr>
                <tr runat="server">
                    <td runat="server"
                        style="text-align: center;background-color: #FFCC66;font-
family: Verdana, Arial, Helvetica, sans-serif;color: #333333;">
                        <asp:DataPager ID="DataPager1" runat="server"
PageSize="12">
                            <Fields>
                                <asp:NextPreviousPagerField ButtonType="Button"
ShowFirstPageButton="True"
                                    ShowNextPageButton="False"
ShowPreviousPageButton="False" />
                                <asp:NumericPagerField />
                                <asp:NextPreviousPagerField ButtonType="Button"
ShowLastPageButton="True"
                                    ShowNextPageButton="False"
ShowPreviousPageButton="False" />
                            </Fields>
                        </asp:DataPager>
                    </td>
                </tr>
            </table>
        </LayoutTemplate>
    </asp:ListView>
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
        ConnectionString="<%$ ConnectionStrings:Database1ConnectionString1 %>"
        DeleteCommand="DELETE FROM [tabIzdelkov] WHERE [id] = @id"
        InsertCommand="INSERT INTO [tabIzdelkov] ([znamka], [tip], [diagonala],
[procesor], [hdd], [ram], [slika], [cena], [aktiven]) VALUES (@znamka, @tip,
@diagonala, @procesor, @hdd, @ram, @slika, @cena, @aktiven)"
        ProviderName="<%$
ConnectionStrings:Database1ConnectionString1.ProviderName %>"
        SelectCommand="SELECT [id], [znamka], [tip], [diagonala], [procesor],
[hdd], [ram], [slika], [cena], [aktiven] FROM [tabIzdelkov]"
        UpdateCommand="UPDATE [tabIzdelkov] SET [znamka] = @znamka, [tip] = @tip,
[diagonala] = @diagonala, [procesor] = @procesor, [hdd] = @hdd, [ram] = @ram,
[slika] = @slika, [cena] = @cena, [aktiven] = @aktiven WHERE [id] = @id">
        <UpdateParameters>
            <asp:Parameter Name="znamka" Type="String" />
            <asp:Parameter Name="tip" Type="String" />
            <asp:Parameter Name="diagonala" Type="Double" />
            <asp:Parameter Name="procesor" Type="Double" />
            <asp:Parameter Name="hdd" Type="Double" />
            <asp:Parameter Name="ram" Type="Double" />
            <asp:Parameter Name="slika" Type="String" />
            <asp:Parameter Name="cena" Type="Double" />
            <asp:Parameter Name="aktiven" Type="Boolean" />
            <asp:Parameter Name="id" Type="Int32" />
```

```
        </UpdateParameters>
        <DeleteParameters>
            <asp:Parameter Name="id" Type="Int32" />
        </DeleteParameters>
        <InsertParameters>
            <asp:Parameter Name="znamka" Type="String" />
            <asp:Parameter Name="tip" Type="String" />
            <asp:Parameter Name="diagonala" Type="Double" />
            <asp:Parameter Name="procesor" Type="Double" />
            <asp:Parameter Name="hdd" Type="Double" />
            <asp:Parameter Name="ram" Type="Double" />
            <asp:Parameter Name="slika" Type="String" />
            <asp:Parameter Name="cena" Type="Double" />
            <asp:Parameter Name="aktiven" Type="Boolean" />
        </InsertParameters>
    </asp:SqlDataSource>
```

12 Varnost

Po urejeni administraciji z izdelki se pojavi vprašanje, ali kupec lahko pride na stran Admin.aspx in začne spreminjati ceno, slike itd. Ravno zato moramo poskrbeti, da do občutljivih strani lahko dostopajo samo izbrani uporabniki ali skupine. Na začetku smo ustvarili mapo admin, vanjo pa premaknili datoteki Admin.aspx in Narocila.aspx. ASP.NET ima namreč tak princip zaščite strani, da dovoljuje ali blokira vsebino v mapah. Za vsako mapo posebej določimo, ali ima uporabnik ali skupina uporabnikov do njene vsebine dostop ali ne. Privzeto imajo dostop do datotek vsi uporabniki (Anonymus users).

12.1 Pravice dostopa uporabnikov do vsebine v mapah

Preden začnemo z nastavitvami varnosti, odprimo datoteko **Site1.Master** in vanjo zraven košarice iz orodjarne iz zavihka Login dodajmo gradnika **LoginName** in **LoginStatus**, tako kot je na spodnji sliki. Od prej imamo narejeno že tabelo z eno vrstico in štirimi celicami, zato bomo uporabili še ostali dve celici.

Slika 114: razporeditev gradnikov UserName in Login

Vsebina značke **div.login** v Site1.Master:

```
            <div class="login">
            <table>
            <tr>
                <td>
                <asp:Menu ID="Menu2" runat="server">
                    <Items>
                        <asp:MenuItem NavigateUrl="~/Kosarica.aspx"
Text="Kosarica" Value="Kosarica"></asp:MenuItem>
                    </Items>
                </asp:Menu>
                </td>
                <td>
                <asp:Label ID="Label1" runat="server" Text="0 €" Font-Bold="True"
Font-Size="Large"></asp:Label>
```

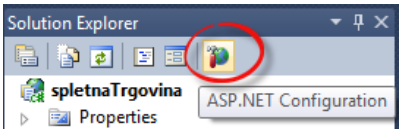
```
            </td>
            <td>   
                <asp:LoginName ID="LoginName1" runat="server" /> 
             </td>
            <td>
                <asp:LoginStatus ID="LoginStatus1" runat="server" />
            </td>
        </tr>
        </table>
        </div>
```

Gradnik **LoginName** je namenjen prikazovanju uporabniškega imena, ki je trenutno prijavljen. Če zaženemo projekt, opazimo, da je prijavljen uporabnik **MYPC\andrej**, kar pomeni, da je MYPC ime strežniškega računalnika, andrej pa je uporabniški račun v sistemu Windows (seveda trenutno prijavljen).

Slika 115: Prikaz prijavljenega uporabnka

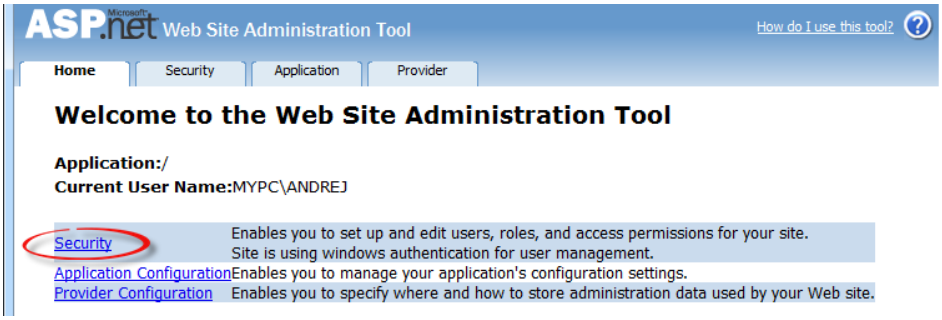
Tudi če sedaj kliknemo na gumb Logout, se ne bo zgodilo nič, ker aplikacija ne more kar odjaviti prijave v sistemu Windows. Zato bomo poskrbeli, da se bo lahko vsak uporabnik registriral oz. naredil svoj uporabniški račun, ter mu dovolili, katere strani lahko odpira in katerih ne. Namesto MYPC\andrej bo izpisalo uporabniško ime, s katerim smo prijavljeni v aplikacijo in ne v Windows.

V oknu Solution Explorer kliknimo na gumb **ASP.NET Configuration**.



Slika 116: Povezava do orodja za spletno administracijo

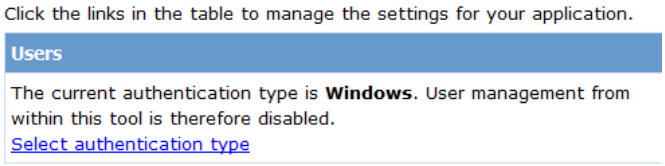
V spletnem brskalniku se odpre administracijsko orodje za konfiguriranje naše spletne aplikacije. Izberimo **Security**.



Slika 117: Začetna stran orodja za administracijo aplikacije

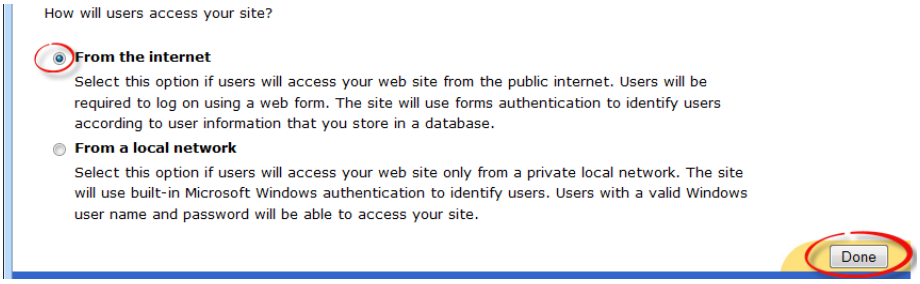
Odpre se stran Security, v kateri bomo najprej nastavili način prijave.

Pod skupino **Users** kliknimo **Select authentication type**.



Slika 118: Izbira avtentikacije 1

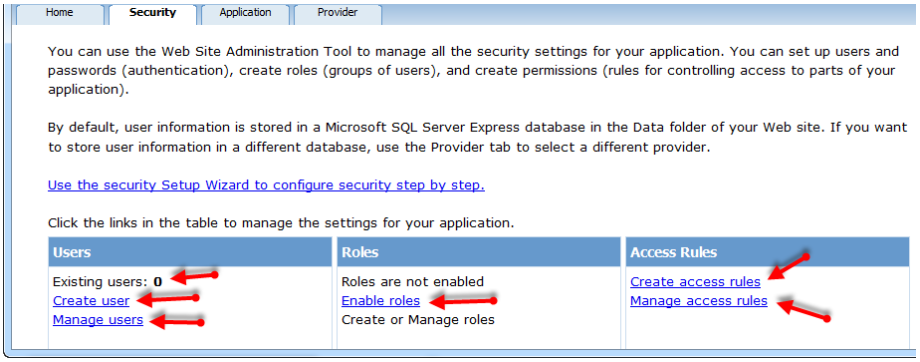
V tem koraku imamo dve možnosti. **From a loacal network** pomeni, da bo v aplikacijo prijavljen preko prijave Windows.
Izberimo možnost **From the internet**, ta način pa bo uporabniško ime tisto, ki je registrirano v bazi. Ko klinemo **Done**, se v mapi **App_Data** ustvari nova baza z imenom **ASPNETDB.MDF**, v kateri se shranijo vsi podatki o uporabnikih, skupinah in pravicah dostopa do posameznih map itd.



Slika 119: Izbira avtentikacije 2

Kliknimo gumb **Done**.

- Pod skupino Users se prikaže število registriranih uporabnikov v naši aplikaciji.
- Pod možnostjo Create user lahko ročno dodamo novega uporabnka ter določimo, v katero skupino pripada.
- Manage users – urejamo uporabnika ter mu spreminjamo skupine. Enable roles – omogočimo skupine v naši aplikaciji.
- Možnost Enable role omogoči uporabniške skupine.
- Create access rules – ustvarimo novo pravilo, kdo lahko in kdo ne sme dostopati do datotek v posamezni mapi.
- Manage access rules – urejamo ta pravila.



Slika 120: Orodje za urejanje uporabnikov, skupin in pravil

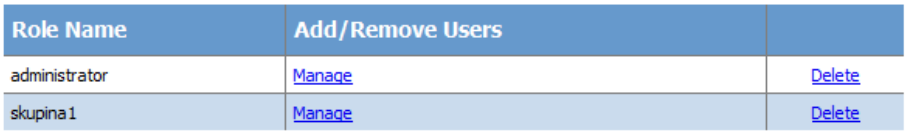
Kliknimo možnost **Enable roles**. S tem bomo omogočili skupine. Prikaže se nam, koliko skupin obstaja v naši aplikaciji, možnost, da izključimo skupine, ter možnost, da ustvarimo ali urejamo novo skupino.

Slika 121: Uporabniške skupini (Roles)

Kliknimo **Create or Manage roles**. Odpre se okno, v katerem ustvarjamo nove skupine. Vpišimo **administrator** ter kliknimo **Add Role**, nato pa dodamo še **skupina1** ter kliknimo **Add Role**.

Slika 122: Dodajanje nove skupine

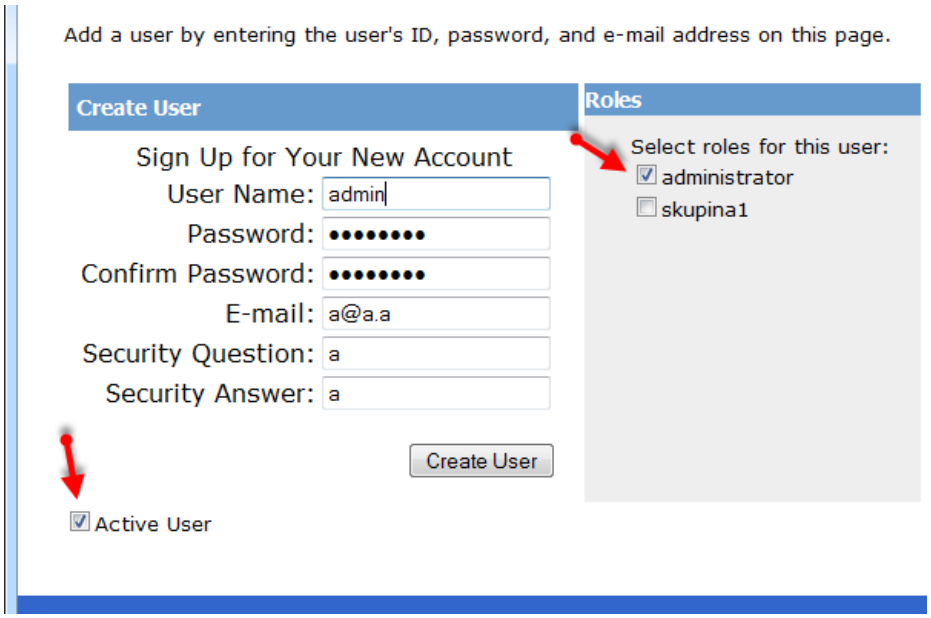
Tako smo ustvarili 2 skupini. Administrator in skupina1. Vse skupine se prikažejo v spodnji tabeli in jih lahko kadarkoli izbrišemo.



Slika 123: Seznam vseh skupin

Kliknimo gumb **Back**. Opazimo, da imamo sedaj pod izpisom Existing roles 2 skupini.
Ustvarimo še nekaj uporabnikov.
Kliknimo na možnost **Create user** ter napišimo sledeče podatke:

1. Username = **admin**
2. Password = **…….** (7 pik, ker mora biti geslo dolžine najmanj 7 znakov, vsebovati mora pa tudi posebne znake. V nadaljnje bomo za vse uporabnike ustvarili isto geslo, da jih kasneje ne bomo pozabili. Lahko dodelite tudi drugo geslo).
3. E-mail: **a@a.a** (na ta naslov se bo poslalo pozabljeno ime in geslo).
4. Security Question: **a**
5. Securiti Answer: **a**
6. Na koncu pa mu dodelimo tudi skupino **administrator**, ter kliknemo gumb **Create user**.

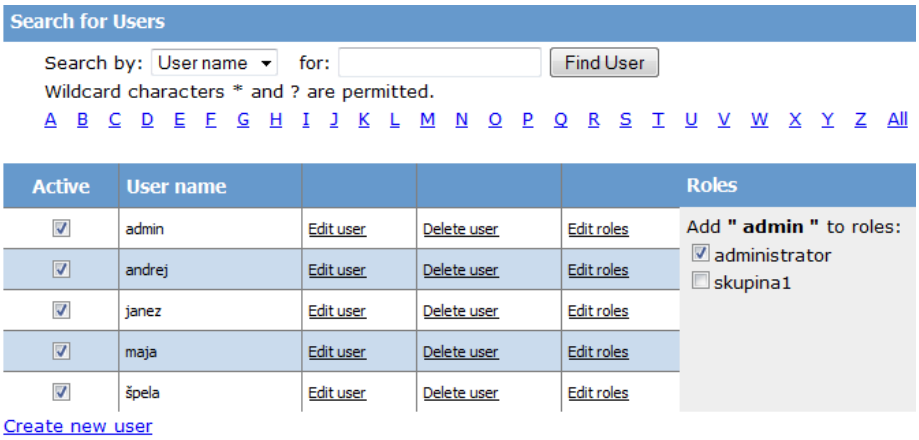


Slika 124: Dodajanje novega uporabnika

Dodajmo še nekaj uporabniških imen in jim določimo naslednje skupine:

- andrej (administrator, skupina1),
- janez (ni v nobeni skupini),
- špela (skupina1),
- maja (ni v nobeni skupini).

Kliknimo gumb **Back**. V aplikaciji imamo sedaj 5 uporabnikov. Kliknimo na gumb **Manage users**. Na tem mestu lahko pregledujemo vse uporabnike ter jim urejamo podatke in jim dodeljujemo skupine.

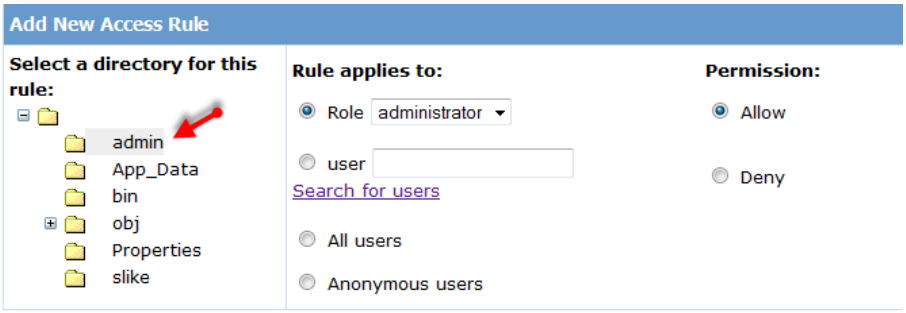


Slika 125: Seznam vseh uporabnikov

Kliknimo gumb **Back**, potem pa kliknimo **Create access rules.**
Odpre se nam okno, v katerem bomo nastavili pravila za posamezno mapo, kdo lahko dostopa do vsebine v tej mapi in kdo ne.
V oknu **Select a directory for this rule:** izberemo, v kateri mapi bo veljalo to pravilo. Če izberemo korensko mapo (tista, ki nima napisa), bo to veljalo za vse datoteke v tej korenski mapi.

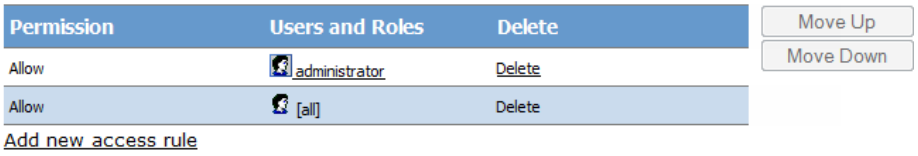
Če izberemo mapo **admin**, tako kot je označeno na spodnji sliki, bodo ta pravila veljala samo za dostop do strani **Admin.aspx in Naročila.aspx**, ker sta v mapi admin.

Naslednji del je **Rule applies to:** Tu pa izberemo, za koga bodo veljala ta pravila. Če imamo izbrano opcijo **Role**, lahko izberemo posamezno skupino, vendar samo eno na enkrat. To pravilo potem velja samo za izbrano skupino. Če bi želeli dostop za še kakšno skupino, bi dodali novo pravilo. Če izberemo opcijo **user**, lahko s **Search for users** obkljukamo želene uporabnike, da nam jih ni potrbno vtipkati v vnosno polje. Če izberemo opcijo **All users**, to pravilo velja za čisto vse uporabnike. Prijavljene in neprijavljene.
In še nazadnje **Anonymus users:** to so **neprijavljeni uporabniki**, kar pride prav zlasti, ko dovolimo dostop samo prijavljenim uporabnikom (recimo, da odda naročilo v košarici).
Z opcijo **Premissions:** določimo ali omogočimo ali onemogočimo dostop do vsebine v mapi.



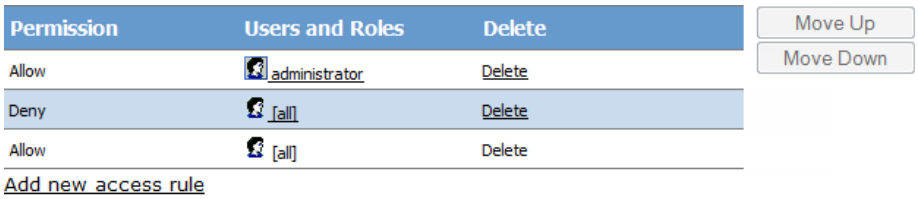
Slika 126: Določanje pravil dostopa

Označimo mapo admin, izberimo opcijo **Role** ter izberimo skupino **administrator** ter dovoljenje **Allow**, nato pa kliknimo **OK**. Vrne nas na stran Security, nato pa kliknimo **Manage access rules**, kjer bomo dodali nova pravila za to mapo.
Tu bomo uredili seznam pravil za **mapo admin**. Kot vidimo, imamo sedaj dve pravili. Spodnje je privzeto in vsem dovoljuje dostop do vsebine mape admin. Nato smo dodali še pravilo, ki skupini administratorjev dovoljuje isto pravico. Torej spet lahko vsi dostopajo. Dodajmo še pravilo, ki vsem onemogoča dostop do strani, zato kliknimo **Add new access rule**.



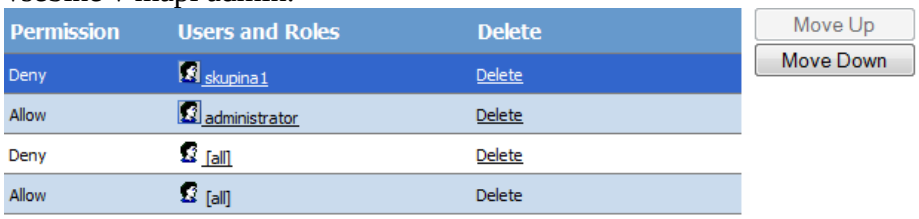
Slika 127: Določanje pravil dostopa 2

Dobili smo novo pravilo, ki vsem uporabnikom onemogoča dostop do vsebine mape admin. Rekli bi, da se sedaj zgornji pravili izključujeta med seboj, vendar ne. **Tisto pravilo, ki je višje na seznamu, ima večjo veljavo in izniči tisti izključujoči del spodnjega pravila**.



Slika 128: Določanje pravil dostopa 3

Dodajmo še eno pravilo, in sicer: **skupina1 Deny**. Za to mapo imamo še eno pravilo, označimo ga in z gumbom **Move Up** premaknemo čisto **na vrh seznama**. S tem smo vsem, ki pripadajo **skupina1, onemogočili dostop do vsebine v mapi admin**. Oseba, ki je v skupini skupina1 in administrator, ima sedaj onemogočen dostop, ker je prepoved skupine1 višje na seznamu kot pa odobritev administratorjem. Če pa je oseba samo v skupini administrator, lahko dostopa do vsebine v mapi admin.



Slika 129: Določanje pravil dostopa 4

Zraven bi lahko na enak način dodali še pravila za posamezne uporabnike, vendar to lahko naredite sami. Če bi imeli večjo aplikacijo, bi verjetno imeli tudi več map, katerim lahko nastavljamo različne pravice dostopa. Sicer nesmiselno lahko pa celo tako, da imajo dostop do določenih vsebin samo neprijavljeni uporabniki (Anonymus users), administrator pa ima zavrnjeno. Skratka, s temi pravili lahko določimo prav vse.
Kliknimo Done ter zaprimo brskalnik.

12.2 Logiranje

Odprimo še datoteko **Registracija.aspx** in vanjo iz orodjarne iz zavihka Login dodajmo gradnik **CreateUserWizzard**. To je gradnik, ki sam poskrbi za registracijo novega uporabnika, če vanj vnesemo vse podatke. Gradniku lahko spremenimo besedilo in obliko.
HTML:

```
<asp:CreateUserWizard ID="CreateUserWizard1" runat="server">
    <WizardSteps>
        <asp:CreateUserWizardStep runat="server" />
        <asp:CompleteWizardStep runat="server" />
    </WizardSteps>
</asp:CreateUserWizard>
```

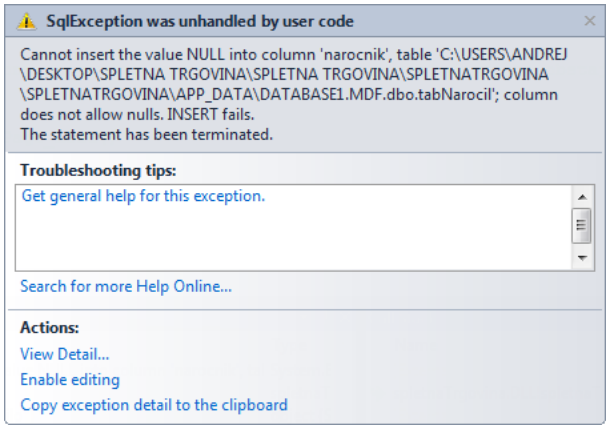
Odprimo datoteko **LogIn.aspx** ter iz orodjarne iz zavihka Login dodajmo gradnik **Login**. Preko tega obrazca se bomo lahko prijavili v našo aplikacijo.
HTML:

```
<asp:Login ID="Login1" runat="server">
</asp:Login>
```

Vse skupaj shranimo in zaženimo projekt (F5). Če sedaj v brskalniku kliknemo na povezavo Admin, nas le-ta ne spusti do urejanja izdelkov, ampak se moramo najprej prijaviti z uporabniškim imenom, ki je v skupini administrator in ni v skupini skupina1. V prijavni obrazec vpišimo uporabniško ime: **admin**, geslo: **…….**, oz. kar smo določili sami. Aplikacija nas po

uspešni prijavi avtomatično preusmeri na **Admin.aspx**. Poizkusimo še uporabniško ime **andrej**, ki pa je v skupini administrator in skupina1. Preizkusimo še ostala uporabniška imena.

Odjavimo se ter poizkusimo izbrati en izdelek, ga dodajmo v košarico in oddajmo naročilo. Zgodila se je izjema. Dobili smo obvestilo, da ne more vnesti NULL vrednosti v polje naročnik.



Slika 130: Obvestilo o izjemi SqlExeption

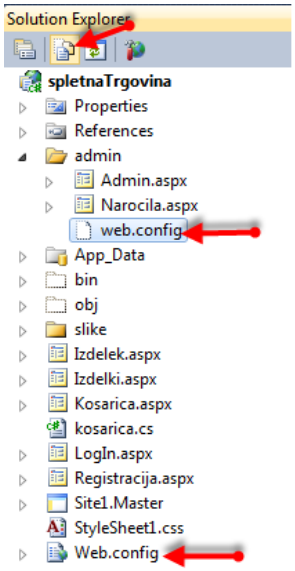
Če nihče ni prijavljen v aplikacijo, tudi ni uporabniškega imena. Torej moramo poskrbeti za zaščito tako, da bomo oddali naročilo samo, če bo uporabnik prijavljen v aplikacijo.
Odprimo datoteko Kosarica.aspx.cs, kjer bomo dodali if stavek, ki bo preveril, če je obiskovalec prijavljen v aplikacijo.
V metodi Button1_Click dodajmo vso kodo v if blok, tako kot vidimo na spodnji kodi

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (User.Identity.IsAuthenticated)
     {
        //ustvarimo novo zbirko tab, ki je tipa List<kosarica>
        List<kosarica> tab = new List<kosarica>();
        //Preverimo, če je ni prazna
        if (Session["kos"] != null)
        {
             //Potem prepišemo košarico iz Sessiona v zbirko tab.
             tab = (List<kosarica>)Session["kos"];
             //Shranimo trenutni datum in čas. Možno je, da moramo datum in čas
zapisati v točno določeni obliki
             string dat = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");
             //Zanka, ki se izvede tolikokrat, kolikor je naročil v zbirki tab.
             for (int i = 0; i < tab.Count; i++)
             {
                //Vnesemo Insert parametre v SqlDataSource.
                //V naročnika shranimo uporabniško ime.
                SqlDataSource1.InsertParameters["narocnik"].DefaultValue =
User.Identity.Name;
                SqlDataSource1.InsertParameters["idIzdelek"].DefaultValue =
tab[i].id.ToString();
                //Kadar bomo podajali parametre, je decimalno ločilo znak pika. V
C# pa pri pretvarjanju v string dobimo
                //decimalno ločilo vejico, zato moramo v številu vejico zamenjati
s piko.
```

```
                SqlDataSource1.InsertParameters["kolicina"].DefaultValue =
tab[i].kolicina.ToString().Replace(',', '.');
                SqlDataSource1.InsertParameters["skupnaCena"].DefaultValue =
tab[i].cenaSkupaj.ToString().Replace(',', '.');
                SqlDataSource1.InsertParameters["datum"].DefaultValue = dat;
                SqlDataSource1.InsertParameters["poslano"].DefaultValue = "FALSE";
                //Zapis se doda v bazo.
                SqlDataSource1.Insert();
            }
            //Izpraznimo zbirko tab.
            tab = new List<kosarica>();
            //Spremenljivko "kos" v Sessionu postavimo na vrednost null. To
pomeni, da ta ne obstaja več
            Session["kos"] = null;
        }
        //Osvežimo stran s klicem metode Page_Load()
        Page_Load(sender, e);
    }
    else { Response.Write("Prosim, če se prijavite!"); }
}
```

Če bomo hoteli neprijavljeni naročiti izdelek, se nam bo na vrhu strani izpisalo besedilo: **Prosim, če se prijavite**!

Oglejmo si še, kaj se zgodi, če kateri od map dodelimo pravila dostopa. V Solution Explorerju kliknimo na gumb **Show All Files**. Prikažejo se nam vse mape in datoteke, tudi tiste, ki so skrite. Ena od teh je tudi **web.config** v mapi admin.



Slika 131: Prikaz datoteke web.config v mapi admin

Če si ogledamo vsebino te .xml datoteke, vidimo, da so se vanjo zapisala pravila, kdo ima odobreno in kdo zavrnjeno pravico do uporabe vsebine v tej mapi.
web.config:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
```

```
    <system.web>
        <authorization>
            <deny roles="skupina1" />
            <allow roles="administrator" />
            <deny users="*" />
        </authorization>
    </system.web>
</configuration>
```

13 Oblika

Oblika in pogled trenutno nista videti ravno najbolj profesionalno. Lahko bi začeli pisati CSS. Lažja pot pa je, da vse skupaj vgradimo v neko že vnaprej pripravljeno predlogo (Template). Najbolje bo, da pobrskamo po spletu ter si ogledamo, kakšno brezplačno predlogo (Free CSS Template), nato pa jo uporabimo kar v našem projektu. Seveda ga bomo malo prilagodili našim potrebam in željam.

13.1 Vstavljanje v predlogo

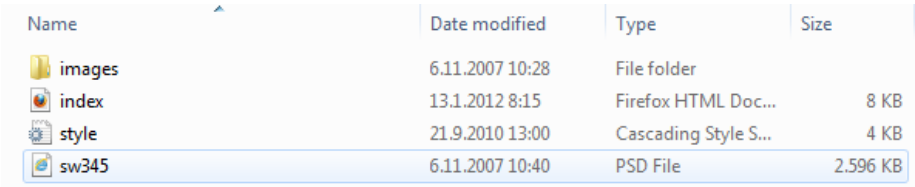
Na tej povezavi si oglejmo nekaj brezplačnih predlog.
http://www.websitetemplates.org/free-templates/

Recimo, da smo se odločili za to na spodnji povezavi, vendar se morate registrirati:
http://www.websitetemplates.org/free-templates/preview/Computer-online-shop.html
Lahko pa dobite direktno kopijo na spodnji povezavi:
http://arh.tsckr.si/SiPS/NSA/spletna trgovina/ComputerStoreTemplate.zip
Izgled predloge je sledeč:

Slika 132: Brezplačna predloga iz spleta

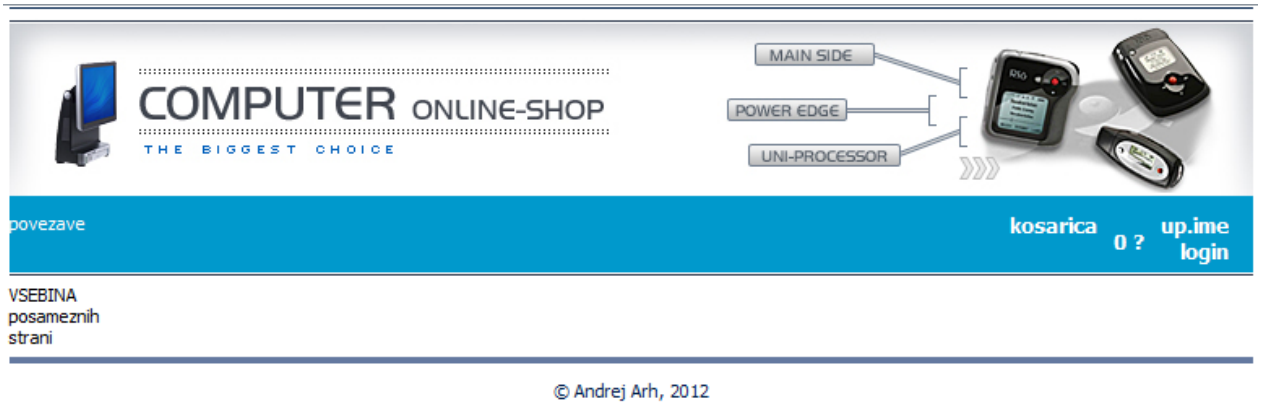
Povlecimo predlogo s spleta. Dobili smo stisnjeno datoteko **ComputerStoreTemplate.zip**. To datoteko **razpakirajmo**. V njej so naslednje datoteke, ki jih potrebujemo:

- **Index.html** (v tej je zapisana html koda, ki jo bomo potem vgradili v MasterPage).
- **Style.css** (v njej je zapisana oblika).
- Mapa **images**, v kateri so vse potrebne slike, ki jih potrebuje predloga.



Slika 133: Vsebina datoteke ComputerStoreTamplate.zip

Ker nekaterih delov predloge ne bomo potrebovali, jih iz datoteke index.html enostavno izbrišemo. Cilj je, da index.html pripravimo do te mere, da ga bomo lahko vgradili v MasterPage. Ko smo malo predelali **index.html in style.css**, smo dobili izgled, kakršen je na spodnji sliki:



Slika 134: Predelana HTML in CSS vsebina iz predloge

Nova vsebina datoteke index.html:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Home</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-5" />
<link rel="stylesheet" type="text/css" href="style.css" />
</head>

<body>
    <div id="header">
        <img src="images/name.jpg" /><img src="images/pic1.jpg" />
        <div class="menu1">
        <div class="menu">
                povezave
            </div>
        <ul class="user">
            <li>
```

```
                kosarica
            </li>
            <li class="cena">
                0 €
            </li>
            <li>
                up.ime
                <br />
                login
            </li>
        </ul>

        </div>
    </div>
    <div id="content">
          <div >
        VSEBINA <br />posameznih <br /> strani
          </div>
    </div>
    <div id="footer">
          &copy Andrej Arh, 2012
    </div>
</body>
</html>
```

Nova vsebina datoteke style.css:

```
*{
margin:0px;
padding:0px
}
img{border:0px}
html{
width:100%;
height:100%;
}
body{
width:766px;
margin:auto;
font-family:Tahoma;
font-size:11px;
}

.select{
width:108px;
font-size:10px;
height:15px;
vertical-align:middle;
margin:0 0 0 5px;
display:inline;
font-family:Tahoma;
}
.squ{
padding:1px 8px 3px 0
}
#header{
float:left
}
.menu1
{
    float: left;
```

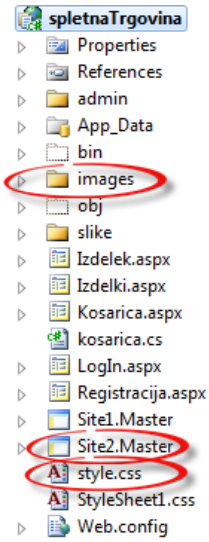
```
    width: 766px;
    background-color: #0099CC;
    color: White;
}
.menu1 a:hover
{
    color:#FFFF99;
    font-weight:bolder;
    text-decoration:none;
}
.menu li{
    float: left;
    width: 110px;
    height: 29px;
    font-size: large;
    list-style-type: none;
    text-align:center;
    vertical-align:middle;
}
.menu a
{
    color:White;
    text-decoration:underline;
}
.menu{
    float:left;
    padding-top:10px;
}
.user
{
    float:right;
    list-style-type:none;
    color:White;
    padding-right:10px;
    padding-top:5px;
}
.user li
{
    padding:5px;
    float:left;
    font-size:small;
    font-weight:bold;
    text-align: right;
}
.cena
{
    margin-top:10px;
}
#content{
float:left;
margin:1px 0 0 0;
border-top:1px solid #303B4F;
width:100%;
padding:5px 0 6px 0;
}

.dots{
background-image:url(images/dots.gif);
background-position:bottom left;
background-repeat:repeat-x
}
```

```
.style1{margin:0 0 14px 0}
.right{
float:left;
padding:0 0 0 5px
}
.item{
background-image:url(images/bordbg.gif);
background-position:top left;
background-repeat:repeat-y;
width:274px;
margin:0 3px 3px 0;
float:left
}
*html .item{
margin:0 0 3px 0
}
.item img{
float:left
}
.item div{
margin:6px 7px 5px 7px;
float:left
}
.items{
padding:0 30px 5px 0;
}
.item h1{
font-family:Tahoma;
font-size:11px;
font-weight:bold;
color:#00214B;
text-decoration:underline;
padding:7px 0 7px 0
}
.item span{
display:block;
padding:0 5px 5px 0;
width:115px;
font-size:10px;
font-family:Tahoma;
float:left;
color:#333A43
}
.item span strong{
display:block;
clear:both
}
.item p{
color:#666D75;
font-size:11px;
}
.item p strong{
font-family:Tahoma;
font-size:18px;
color:#FF3000;
margin:0 10px 0 0
}
#footer{
width:100%;
float:left;
border-top:4px solid #6479A0;
```

```
padding:10px 0 15px 0;
text-align:center;
color:#00214B
}
.powered{
color:#00214B
}
.terms{
color:#0051B6
}
table,td
{
    padding: 5px;
    text-align: right;
}

#content .inner_copy {border:0;color:#f00;float:left;width:50%!important;margin:-
202px 0 0 0;overflow:hidden;line-height:0;padding:0;font-size:12px}
```

Sedaj moramo samo paziti, da vse postavimo na svoje mesto. Še ena zelo pomembna stvar. Kadar katerakoli HTML značka kaže na relativno povezavo v projektu, npr. povezava na sliko ali kakšno drugo datoteko, nastane problem. Ko se nahajamo v kakšni podmapi, kot v našem primeru Admin.aspx ali Naročila.aspx, bi moral biti pred imenom dodan ukaz **../**, kar pomeni v mapi en nivo nazaj proti korenski mapi. Zato bo naslovna slika prikazana na vseh straneh, razen Admin.aspx in Naročila.aspx. Iz tega razloga **vedno uporabljajmo ASP.NET gradnike, kot so Image, ImageButton, Menu, Hyperlink …, za katere ASP.NET sam poskrbi, da se poti vedno pravilno naslovijo, ne glede na lokacijo v mapi ali podmapi**.

V projekt **prekopirajmo (Ctrl+c, Ctrl+v) datoteko style.css** ter ji dodelimo tako kodo, kot je prikazano zgoraj.
Ravno tako v projekt **skopirajmo** celotno mapo **image**, ki smo jo dobili v brezplačni predlogi.
Ustvarimo še novo **MasterPage stran**, ki naj bo poimenovana privzeto, **Site2.Master**.



Slika 135: Dodane datoteke za novo predlogo

Odprimo datoteko **Site2.Master** ter vanjo prekopirajmo **body iz datoteke index.html**. Obenem pazimo, da v Site2.Master ne izbrišemo gradnika **ContentPlaceHolder,** ampak z njim **nadomestimo** napis **Vsebina posamezne strani**.
HTML koda datoteke Site2.Master na bo sledeča:

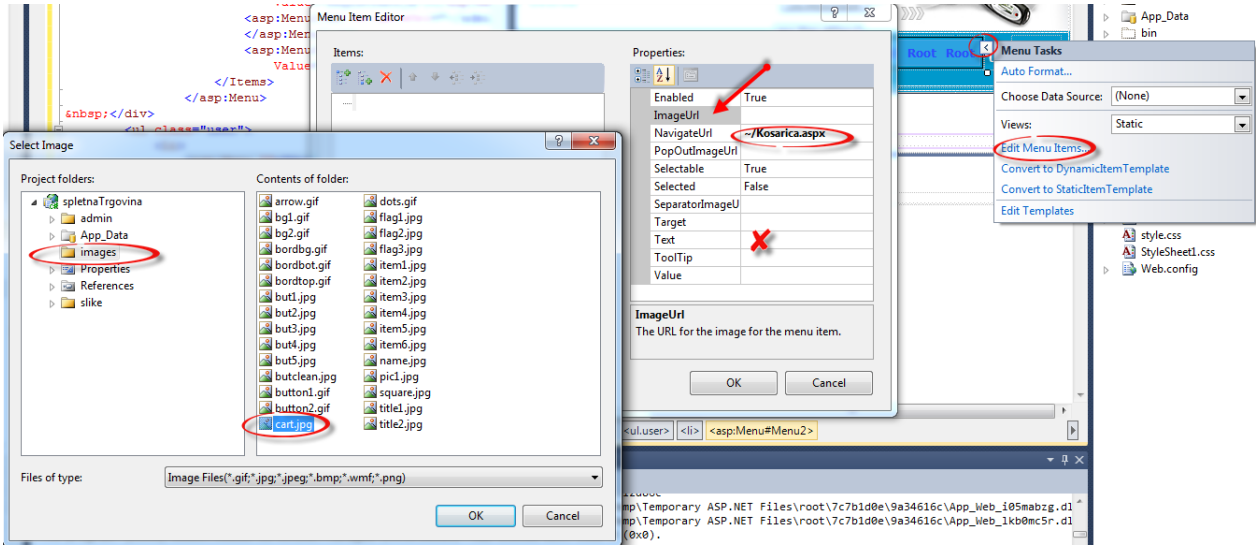
```
<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="Site2.master.cs"
Inherits="spletnaTrgovina.Site2" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <asp:ContentPlaceHolder ID="head" runat="server">
    </asp:ContentPlaceHolder>
    <link href="style.css" rel="stylesheet" type="text/css" />
</head>

<body>
    <form id="form1" runat="server">
    <div>
    <div id="header">
        <img src="images/name.jpg" /><img src="images/pic1.jpg" />
            <div class="menu1">
        <div class="menu">
                povezave
            </div>
        <ul class="user">
            <li>
                kosarica
            </li>
            <li class="cena">
                0 €
            </li>
            <li>
                up.ime
                <br />
                login
            </li>
        </ul>

        </div>
      </div>
      <div id="content">
            <div >
 <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
 </asp:ContentPlaceHolder>
            </div>
      </div>
      <div id="footer">
            &copy Andrej Arh, 2012
      </div>
    </div>
    </form>
</body>
</html>
```

Sedaj pa nadomestimo besedilo v zgornji kodi z naslenjimi gradniki:

- Povezave: Menu
- Kosarica: Menu
- 0 €: Label
- Up.ime: LoginName
- Login: LoginStatus (označimo ga ter v oknu properties/Font določimo belo barvo)

Določimo **navigacijski meni** oz. povezave, tako kot smo to naredili za začetku te literature v datoteki Site1.Master. Določimo **košarico** ter ji **namesto besedila prikažimo sliko**. Kliknimo na zavihek **Menu Task**, izberimo **Edit Menu Items**. Ko dodamo novo povezavo, **izbrišimo Text**, določimo povezavo **NavigateUrl**, kliknimo na **ImageUrl** na gumb 

. Odpre se okno, v katerem izberemo, katera slika se bo prikazala namesto besedila. Izberimo **mapo images** ter sliko **cart.jpg**, OK in OK.



Slika 136: Določanje slike namesto napisa košarica

Odprimo še datoteko Site2.Master.cs ter določimo izračun skupne cene v košarici. Kodo lahko kar skopiramo iz datoteke Site1.Master.cs.
Koda:

```
protected void Page_Load(object sender, EventArgs e)
{

}

protected void Page_PreRender(object sender, EventArgs e)
{
    Label1.Text = "0 €";
    List<kosarica> tab = new List<kosarica>();
    if (Session["kos"] != null)
    {
        tab = (List<kosarica>)Session["kos"];
        double skupaj = tab.Sum(y => y.cenaSkupaj);
        Label1.Text = skupaj + " €";
    }
}
```

Če sedaj želimo, da se na posameznih straneh vključi nova predloga Site2.Master, moramo to spremeniti na naših obrazcih. Odprimo datoteko **Izdelki.aspx** ter v HTML kodi popravimo lastnost, namesto `MasterPageFile="~/Site1.Master"` določimo **`MasterPageFile="~/Site2.Master"`**. S tem smo dosegli, da bo vključena nova predloga.
HTML glabe obrazca Izdelki.aspx:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site2.Master"
AutoEventWireup="true" CodeBehind="Izdelki.aspx.cs"
Inherits="spletnaTrgovina.WebForm1" %>
```

Enako storimo za vse ostale obrazce, ki smo jih ustvarili. Zaženimo sedaj projekt, se prijavimo kot admin ter pojdimo na stran Admin.aspx ali Naročila.aspx. **Opazimo, da se naslovna slika (Header) ne prikaže**. To pa zato, kot smo že prej omenili, **išče sliko v mapi images/slika.jpg, ker pa se nahajamo v podmapi, bi morali sliko iskati en nivo nazaj ../images/slika.jpg. Zatorej raje namesto značk <img> uporabimo ASP.NET ekvivalentni gradnik Image, da ne bomo imeli teh težav.**
Končna HTML koda datoteke Site2.Master:

```
<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="Site2.master.cs"
Inherits="spletnaTrgovina.Site2" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <asp:ContentPlaceHolder ID="head" runat="server">
    </asp:ContentPlaceHolder>
    <link href="style.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
    <div>
    <div id="header">
        <asp:Image ID="Image1" runat="server"
ImageUrl="~/images/name.jpg" /><asp:Image ID="Image2" runat="server"
ImageUrl="~/images/pic1.jpg" />
        <div class="menu1">
        <div class="menu">
                <asp:Menu ID="Menu1" runat="server">
                    <Items>
                        <asp:MenuItem NavigateUrl="~/Izdelki.aspx" Text="Izdelki"
Value="Izdelki">
                        </asp:MenuItem>
                        <asp:MenuItem NavigateUrl="~/Registracija.aspx"
Text="Registracija"
                            Value="Registracija"></asp:MenuItem>
                        <asp:MenuItem NavigateUrl="~/admin/Admin.aspx"
Text="Admin" Value="Admin">
                        </asp:MenuItem>
                        <asp:MenuItem NavigateUrl="~/admin/Narocila.aspx"
Text="Naročila"
                            Value="Naročila"></asp:MenuItem>
                    </Items>
                </asp:Menu>
 </div>
```

```
        <ul class="user">
            <li>
                <asp:Menu ID="Menu2" runat="server">
                    <Items>
                        <asp:MenuItem ImageUrl="~/images/cart.jpg"
NavigateUrl="~/Kosarica.aspx">
                        </asp:MenuItem>
                    </Items>
            </asp:Menu>
            </li>
            <li class="cena">
                <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
            </li>
            <li>
                <asp:LoginName ID="LoginName1" runat="server" />
                <br />
                <asp:LoginStatus ID="LoginStatus1" runat="server"
ForeColor="White" />
            </li>
        </ul>

        </div>
      </div>
      <div id="content">
            <div >

        <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
        </asp:ContentPlaceHolder>
            </div>
      </div>
      <div id="footer">
            &copy Andrej Arh, 2012
      </div>
    </div>
    </form>
</body>
</html>
```

Pri tem bi samo poudarili, da **morata biti gradnika Image v naslovnici tesno skupaj**, drugače bo skupna slika preširoka na en znak, zato ker je presledek, več presledkov ali nova vrstica v HTML kodi en presledek. Torej gradnika Image morata biti tesno skupaj, tako kot vidimo v zgornji kodi.

13.2 Končni izgled spletne trgovine

Na spodnji povezavi lahko dobite cel projekt spletne trgovine.
http://arh.tsckr.si/SiPS/NSA/spletna trgovina/spletnaTrgovina.zip

Če primerno oblikujemo še vse ostale gradnike ListView, GridView, Label, Button … v posameznih obrazcih, lahko dobimo izgled:

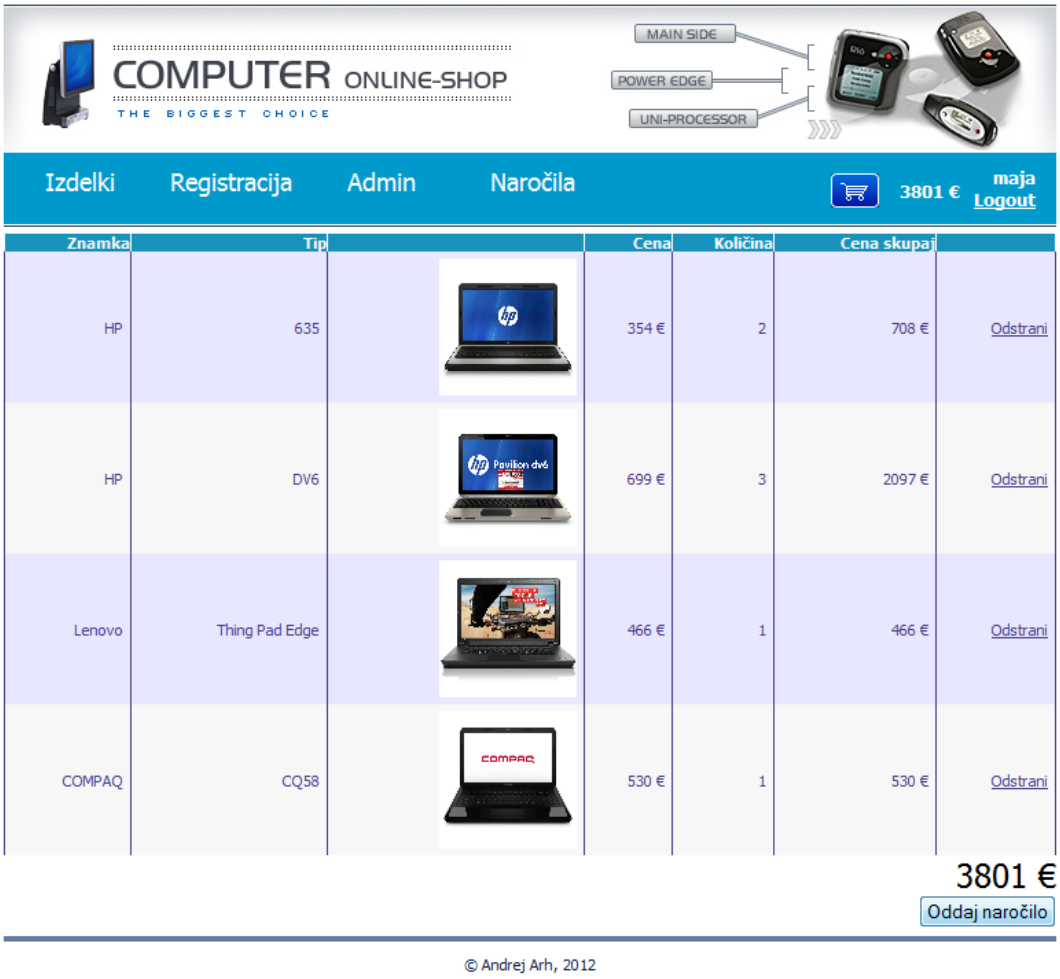
Izdelki.aspx

Slika 137: Izdelki.aspx

Izdelek.aspx

Slika 138: Izdelek.aspx

Kosarica.aspx



Slika 139: Kosarica.aspx

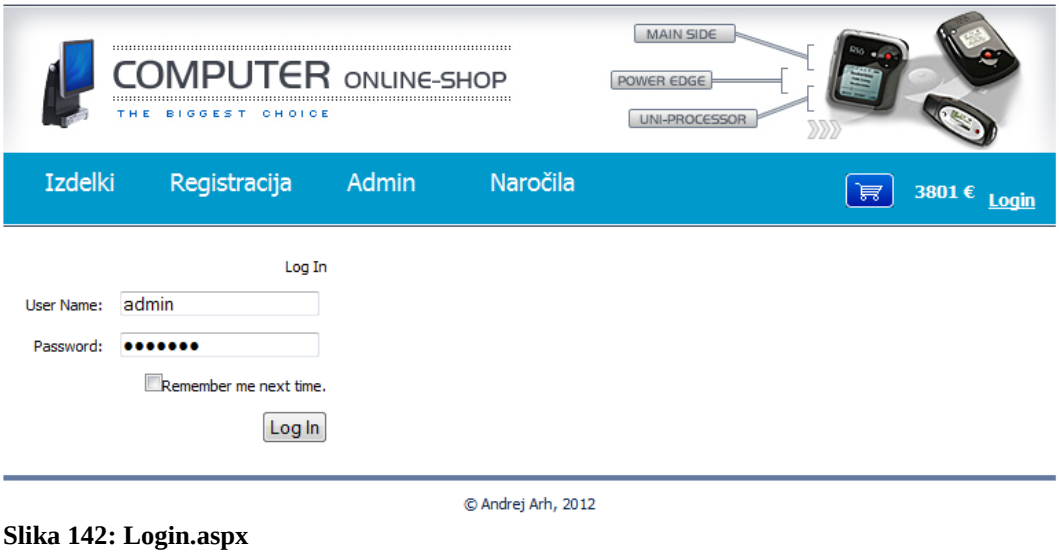
Admin.aspx

Slika 140: Admin.aspx

Narocila.aspx

Slika 141: Narocila.aspx

Login.aspx



Slika 142: Login.aspx

Registracija.aspx

Slika 143: Registracija.aspx

14 Literatura

Beginning ASP.NET 3.5 in C# 2008: From Novice to Professional, Second Edition
Copyright © 2007 by Matthew MacDonald

Pro ASP.NET 3.5 in C# 2008, Second Edition
Copyright © 2007 by Matthew MacDonald and Mario Szpuszta