Upute za rješavanje ASP.NET web forms dijela projektnog zadatka

1. Setup

1.1. Kreirati ASP.NET Framework solution s podrškom za MVC

- Bitno: ne stavljati solution i projekt u isti folder!
- ASP.Net Web Application
- project name: Javno
- solution name: RWA
- .NET framework min. 4.7.2
- odabrati MVC
- autentifikacija i HTTPS neće biti potrebni u ovom trenutku

1.2. Dodati u solution ASP.NET Framework projekt s podrškom za web forme

- ASP.Net Web Application
- project name: Admin
- odabrati Web Forms
- autentifikacija i HTTPS neće biti potrebni

1.3. Savjet: pokretanje projekta

Moguće je odjednom pokretati oba projekta, samo je u Solution Exploreru potrebno uključiti "Multiple startup projects" (properties od solutiona). Ako je lakše u određenom trenutku pokretati pojedini web, onda desni gumb na projekt i odabir "Set as Startup Project".

Napomena: za početak dok radimo samo na Admin dijelu, lakše je odabrati samo njega kao startup projekt.

2. Login/logout i zaštita od neovlaštenog pristupa

Osnova za ovdje opisan proces nalazi se na linku https://docs.microsoft.com/en-us/troubleshoot/developer/webapps/aspnet/development/forms-based-authentication (https://docs.microsoft.com/en-us/troubleshoot/developer/webapps/aspnet/development/forms-based-authentication).

> *NAPOMENA: ako si želite pojednostaviti život, deinstalirajte* `Microsoft.AspNet.FriendlyUrls` *Nuget paket iz Admin projekta. On omogućava ljepše URL-ove (npr. Default umjesto Default.aspx), ali stvara dodatni posao s login sustavom. U slučaju deinstalacije trebat će također obrisati i* `RouteConfig.RegisterRoutes()` *poziv iz* `Globas.asax`*. Pokušajte nakon deinstalacije buildati projekt pa ćete vidjeti zašto*

2.1. Konfiguracija autentifikacije

- promjena se treba učiniti u web.config datoteci
- iz gore navedenog linka treba napraviti konfiguraciju iz sekcije "Configure security settings in the Web.config File"

```
<system.web>
    ...
    <authentication mode="Forms">
        <forms name=".ASPXFORMSDEMO" loginUrl="Logon.aspx" protection="All" path="/" timeout="30" />
    </authentication>
    <authorization>
        <deny users ="?" />
        <allow users = "*" />
    </authorization>
</system.web>
```

2.2. Login forma - osnovno

- u Admin projekt dodati novu stranicu Logon.aspx

- napomena: za login forme/stranice se ne koristi Master Page
- kopirati iz gore navedenog linka HTML kod iz "Create a Logon.aspx page" unutar forme (unutar `<form>` taga)

```
<h3>
  <font face="Verdana">Logon Page</font>
</h3>
<table>
    <tr>
        <td>Email:</td>
        <td><input id="txtUserName" type="text" runat="server"></td>
        <td><ASP:RequiredFieldValidator ControlToValidate="txtUserName"
            Display="Static" ErrorMessage="*" runat="server"
            ID="vUserName" /></td>
    </tr>
    <tr>
        <td>Password:</td>
        <td><input id="txtUserPass" type="password" runat="server"></td>
        <td><ASP:RequiredFieldValidator ControlToValidate="txtUserPass"
        Display="Static" ErrorMessage="*" runat="server"
        ID="vUserPass" />
        </td>
    </tr>
    <tr>
        <td>Persistent Cookie:</td>
        <td><ASP:CheckBox id="chkPersistCookie" runat="server" autopostback="false" /></td>
        <td></td>
    </tr>
</table>
<input type="submit" Value="Logon" runat="server" ID="cmdLogin"><p></p>
<asp:Label id="lblMsg" ForeColor="red" Font-Name="Verdana" Font-Size="10" runat="server" />
```

- u formu dodati server-side gumb za spremanje (izbaciti onaj koji je bio već unutra)
  - savjet: najbolje bi bilo dodati LinkButton, jer se najlakše stilizira
  - napomena: da biste stilizirali kontrole ove stranice, na nju treba "ručno" dodati Bootstrap CSS

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

- provjerite izgleda li forma kako ste očekivali

2.3. Login forma - login pomoću cookie-a

- login i kontrola pristupa rade se pomoću cookie-a
- algoritam:
  - na klik gumba provjeriti je li korisnik unio ispravno korisničko ime i lozinku
  - ako jest, koristimo gotovu funkcionalnost za postavljanje cookie-a i redirect na default stranicu
  - ako nije, vraćamo ga na login stranicu

```
protected void lbLogin_Click(object sender, EventArgs e)
{
    // FormsAuthentication.RedirectFromLoginPage() automatically generates
    // the forms authentication cookie!
    if (ValidateUser(txtUserName.Value, txtUserPass.Value))
        FormsAuthentication.RedirectFromLoginPage(txtUserName.Value, chkPersistCookie.Checked);
    else
        Response.Redirect("Logon.aspx", true);
}
```

- za provjeru korisničkih podataka koristimo pojednostavljeni proces bez baze podataka
  - napomena: ovo nemojte ovako raditi u produkciji!

```
private bool ValidateUser(string userName, string passWord)
{
    if (userName == "123" && passWord == "123")
        return true;

    return false;
}
```

- provjerite radi li login s podacima 123/123
    - napomena: About i Contact neće raditi jer sadrže neispravan link, slobodno to popravite

2.4. Logout kontrola

- dodati logout gumb u Site.Master, spojiti ga na server-side handler

```
<ul class="nav navbar-nav">
    <li><a runat="server" href="~/">Home</a></li>
    <li><a runat="server" href="~/About.aspx">About</a></li>
    <li><a runat="server" href="~/Contact.aspx">Contact</a></li>
    <li><asp:LinkButton ID="lbLogout" runat="server" Text="Logout" /></li>
</ul>
```

- dodajte click-handler; kod za logout je jednostavan:

```
FormsAuthentication.SignOut();
Response.Redirect("Logon.aspx", true);
```

- isprobajte radi li logout

3. Navigacija

Osnova za ovdje opisan proces nalazi se na linku https://docs.microsoft.com/en-us/aspnet/web-forms/overview/data-access/introduction/master-pages-and-site-navigation-cs (https://docs.microsoft.com/en-us/aspnet/web-forms/overview/data-access/introduction/master-pages-and-site-navigation-cs).

3.1. Navigacija - sitemap

- dodati sitemap u projekt (Add > New Item... / Web / General / Site Map)
- sadržaj Web.sitemap datoteke neka za početak reflektira trenutni sadržaj našeg site-a

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
    <siteMapNode url="" title="" description="">
        <siteMapNode url="Default.aspx" title="Default" description="" />
        <siteMapNode url="About.aspx" title="About" description="" />
        <siteMapNode url="Contact.aspx" title="Contact" description="" />
    </siteMapNode>
</siteMap>
```

- u Site.Master datoteci umjesto fiksnih `<li>` tagova unutar `<ul class="nav navbar-nav">` sada ćemo napraviti iscrtavanje sitemape iz Web.sitemap datoteke; za to je potrebno dodati `SiteMapDataSource` kontrolu na koju se veže `Repeater` kontrola

> *Mogli smo iskoristiti `Menu` kontrolu, ali ona nije dovoljno fleksibilna i imali bismo problema sa stiliziranjem*

```
<div class="navbar-collapse collapse">
    <ul class="nav navbar-nav">
        <asp:Repeater runat="server" ID="menu" DataSourceID="SiteMapDataSource1">
            <ItemTemplate>
                <li>
                    <asp:HyperLink runat="server" NavigateUrl='<%# Eval("Url") %>'><%# Eval("Title") %></asp:HyperLink>
                </li>
            </ItemTemplate>
        </asp:Repeater>
        <asp:SiteMapDataSource ID="SiteMapDataSource1" runat="server" ShowStartingNode="false" />
        <li>
            <asp:LinkButton ID="lbLogout" runat="server" Text="Logout" OnClick="lbLogout_Click" /></li>
    </ul>
</div>
```

- provjerimo radi li navigacija ispravno

> *Ubuduće kako mijenjamo strukturu našeg aplikacije, tako ćemo i ažurirati Web.sitemap*

4. Dodavanje stranica / struktura site-a

4.1. "Landing page" / default stranica

Postoje dva pristupa za ovo:

- Pristup 1: obrisati sve iz Default.aspx i koristiti tu stranicu kao listu apartmana
- Pristup 2: za listu apartmana dodati ApartmentList.aspx (s Master Page-om), te u web.config promijeniti default stranicu na serveru

```
<configuration>
    ...
    <system.webServer>
        <defaultDocument>
            <files>
                <clear />
                <add value="ApartmentList.aspx" />
            </files>
        </defaultDocument>
    </system.webServer>
    ...
</configuration>
```

Oba pristupa su sasvim u redu. Nakon što ste odabrali pristup, implementirajte ga i ažurirajte `Web.sitemap` ako je potrebno. Dalje će u rješenju biti odabran pristup (2).

4.2. Dodavanje drugih stranica

> *Sve stranice dodajemo kao stranice koje imaju Master Page.*

- dodati novu praznu stranicu za listu tagova (TagList.aspx)
- dodati novu praznu stranicu za listu korisnika (UserList.aspx)
- ažurirati Web.sitemap tako da pokazuje linkove na `TagList.aspx` i `UserList.aspx`, a ne `About` i `Contact`

> *Sada možemo obrisati višak stranica (About, Contact, te Default ako ga ne koristimo)*

- isprobajmo radi li navigacija s novim stranicama

5. Lista apartmana - komunikacija s bazom podataka

Za početak ćemo "nabrzaka" isprobati povlačenje podataka u GridView. Za to ćemo koristiti `SqlHelper`. U drugom ćemo koraku napraviti pravi sloj prema bazi podataka.

> *Da bismo koristili SqlHelper, moramo instalirati Microsoft.ApplicationBlocks.Data u naš Admin projekt (vidi: https://www.nuget.org/packages/Microsoft.ApplicationBlocks.Data/ (https://www.nuget.org/packages/Microsoft.ApplicationBlocks.Data/)).*

> *Naredba za instalaciju: `Install-Package Microsoft.ApplicationBlocks.Data -Version 2.0.0` Pazite u koji projekt instalirate paket (Admin).*

5.1. Lista apartmana - testiranje spajanja na bazu podataka

Testirat ćemo spajanje na stranici `ApartmentList.aspx`.

- da bismo se mogli spojiti na bazu, moramo dodati connection string u web.config

```
<connectionStrings>
    <add name="rwadb" connectionString="Server=.;Database=RwaApartmani;Trusted_connection=True" />
</connectionStrings>
```

> *Napomena: možda Vaš connection string neće izgledati jednako - ovisi kako koristite bazu podataka*

- u `Page_Load()` kodirajmo povlačenje podataka

```
var connStr = ConfigurationManager.ConnectionStrings["rwadb"].ConnectionString;

string query =
    "SELECT Id, Guid, CreatedAt, DeletedAt, OwnerId, TypeId, " +
    "StatusId, CityId, Address, Name, NameEng, Price, MaxAdults, " +
    "MaxChildren, TotalRooms, BeachDistance " +
    "FROM dbo.Apartment";

var ds = SqlHelper.ExecuteDataset(connStr, CommandType.Text, query);
var res = ds.Tables[0];
```

- provjerimo debuggerom jesmo li dobili podatke

5.2. Lista apartmana - prikaz podataka

- dodajmo GridView za listu apartmana (ID = "gvListaApartmana")

```
<asp:GridView ID="gvListaApartmana" runat="server" CssClass="table table-striped table-condensed table-hover">
</asp:GridView>
```

- proslijedimo u taj gridview dobivenu tablicu sa servera

```
gvListaApartmana.DataSource = ds.Tables[0];
gvListaApartmana.DataBind();
```

- provjerimo prikazuju li nam se podaci

> *Napomena: ovi podaci nam baš i ne odgovaraju tome što želimo dobiti; to ćemo kasnije promijeniti*

5.3. Lista apartmana - kreiranje sloja prema bazi podataka

- iskoristiti ćemo proceduru za povlačenje podataka iz tablice (vidi `dbo.GetApartments`)
- dodajmo Models/Apartment.cs i implementirajmo u njemu sve potrebne karakteristike apartmana s odgovarajućim tipovima

```
public class Apartment
{
    public int Id { get; set; }
    public Guid Guid { get; set; }
    public DateTime CreatedAt { get; set; }
    public DateTime? DeletedAt { get; set; }
    public int OwnerId { get; set; }
    public string OwnerName { get; set; }
    public int TypeId { get; set; }
    public int StatusId { get; set; }
    public string StatusName { get; set; }
    public int? CityId { get; set; }
    public string CityName { get; set; }
    public string Address { get; set; }
    public string Name { get; set; }
    public decimal Price { get; set; }
    public int? MaxAdults { get; set; }
    public int? MaxChildren { get; set; }
    public int? TotalRooms { get; set; }
    public int? BeachDistance { get; set; }
}
```

> *Moramo tu paziti na nullability (npr. ako je u bazi tip `int NOT NULL`, u modelu treba biti `int?`). Osim toga, primijetimo da su u model dodane i string-reprezentacije vezanih tablica `ApartmentOwner`, `ApartmentStatus` i `City`, a ne samo njihovi ID-evi.*

- da bismo dobro enkapsulirali sloj prema bazi podataka, trebamo funkcionalnost repozitorija
- dodajmo klasu Repositories/ApartmentRepository.cs; neka se connection string dohvaća u konstruktoru

```
private readonly string _connectionString;

public ApartmentRepository()
{
    _connectionString = ConfigurationManager.ConnectionStrings["rwadb"].ConnectionString;
}
```

- implementirajmo sada metodu za dohvat apartmana preko stored procedure; metoda mora vraćati listu modela `Models.Apartment`.

```
public List<Models.Apartment> GetApartments()
{
    var ds = SqlHelper.ExecuteDataset(
        _connectionString,
        CommandType.StoredProcedure,
        "dbo.GetApartments");

    var apList = new List<Models.Apartment>();
    foreach (DataRow row in ds.Tables[0].Rows)
    {
        var ap = new Models.Apartment();
        ap.Id = Convert.ToInt32(row["ID"]);
        ap.Guid = Guid.Parse(row["Guid"].ToString());
        ap.CreatedAt = Convert.ToDateTime(row["CreatedAt"]);
        ap.DeletedAt =
            row["DeletedAt"] != DBNull.Value ?
                (DateTime?)Convert.ToDateTime(row["DeletedAt"]) :
                null;
        ap.OwnerId = Convert.ToInt32(row["OwnerId"]);
        ap.OwnerName = row["OwnerName"].ToString();
        ap.TypeId = Convert.ToInt32(row["TypeId"]);
        ap.StatusId = Convert.ToInt32(row["StatusId"]);
        ap.StatusName = row["StatusName"].ToString();
        ap.CityId =
            row["CityId"] != DBNull.Value ?
                (int?)Convert.ToInt32(row["CityId"]) :
                null;
        ap.CityName = row["CityName"]?.ToString();
        ap.Address = row["Address"].ToString();
        ap.Name = row["Name"].ToString();
        ap.Price = Convert.ToDecimal(row["Price"]);
        ap.MaxAdults =
            row["MaxAdults"] != DBNull.Value ?
                (int?)Convert.ToInt32(row["MaxAdults"]) :
                null;
        ap.MaxChildren =
            row["MaxChildren"] != DBNull.Value ?
                (int?)Convert.ToInt32(row["MaxChildren"]) :
                null;
        ap.TotalRooms =
            row["TotalRooms"] != DBNull.Value ?
                (int?)Convert.ToInt32(row["TotalRooms"]) :
                null;
        ap.BeachDistance =
            row["BeachDistance"] != DBNull.Value ?
                (int?)Convert.ToInt32(row["BeachDistance"]) :
                null;
        apList.Add(ap);
    }

    return apList;
}
```

> *Obratite pažnju da za svaki redak u bazi stvaramo novi model i punimo varijablu tog novog modela sa stupcem (poljem) iz tog retka.*

- sada možemo u konstruktoru stranice `ApartmentList.aspx` instancirati `ApartmentRepository` tako da ga (kada je potrebno) koristimo za dohvat podataka iz baze
- možemo kompletnu SQL naredbu zamijeniti s pozivom iz repozitorija

```
private readonly ApartmentRepository _apartmentRepository;

public ApartmentList()
{
    _apartmentRepository = new ApartmentRepository();
}

protected void Page_Load(object sender, EventArgs e)
{
    gvListaApartmana.DataSource = _apartmentRepository.GetApartments();
    gvListaApartmana.DataBind();
}
```

- na kraju ostavimo u gridview-u samo stupce koje trebamo prikazati

```
<asp:GridView ID="gvListaApartmana" runat="server" AutoGenerateColumns="false" CssClass="table table-striped table-condensed ta
    <Columns>
        <asp:BoundField DataField="OwnerName" HeaderText="Vlasnik" />
        <asp:BoundField DataField="StatusName" HeaderText="Status" />
        <asp:BoundField DataField="CityName" HeaderText="Grad" />
        <asp:BoundField DataField="Address" HeaderText="Adresa" />
        <asp:BoundField DataField="Name" HeaderText="Naziv" />
        <asp:BoundField DataField="MaxAdults" HeaderText="Broj odraslih" ItemStyle-HorizontalAlign="Right" />
        <asp:BoundField DataField="MaxChildren" HeaderText="Broj djece" ItemStyle-HorizontalAlign="Right" />
        <asp:BoundField DataField="TotalRooms" HeaderText="Broj soba" ItemStyle-HorizontalAlign="Right" />
        <asp:BoundField DataField="BeachDistance" HeaderText="Udaljenost od plaže" ItemStyle-HorizontalAlign="Right" />
        <asp:BoundField DataField="Price" HeaderText="Cijena" DataFormatString="{0:C}" ItemStyle-HorizontalAlign="Right" />
    </Columns>
</asp:GridView>
```

- provjerimo prikazuju li podaci tako kako smo definirali u gridviewu

5.4. Lista apartmana - filtriranje i sortiranje

U ApartmentList.aspx ćemo dodati kontrole za filtriranje i sortiranje. Trebaju nam dropdown kontrole za filtriranje po statusu i gradu, te za poredak u tablici. Za to ćemo iskoristiti `DropDownList` kontrole.

- kreirajmo navedene dropdowne iznad gridview-a

```
<div>
    <asp:Label ID="lblStatus" runat="server" Text="Status:"></asp:Label>
    <asp:DropDownList ID="ddlStatus" runat="server" CssClass="form-control"></asp:DropDownList>
</div>
<div>
    <asp:Label ID="lblCity" runat="server" Text="Grad:"></asp:Label>
    <asp:DropDownList ID="ddlCity" runat="server" CssClass="form-control"></asp:DropDownList>
</div>
<div>
    <asp:Label ID="lblOrder" runat="server" Text="Sortiranje:"></asp:Label>
    <asp:DropDownList ID="ddlOrder" runat="server" CssClass="form-control"></asp:DropDownList>
</div>
<hr />
<div><asp:GridView ID="gvListaApartmana" runat="server" CssClass="table table-striped table-condensed table-hover"></asp:GridVi
```

- da bismo mogli koristiti podatke o statusu, gradu i redoslijedu sortiranja, trebamo dodati modele i repozitorije za njih

```
// Models/Status.cs
public class Status
{
    public int Id { get; set; }
    public string Name { get; set; }
}

// Models/City.cs
public class City
{
    public int Id { get; set; }
    public Guid Guid { get; set; }
    public string Name { get; set; }
}

// Models/Order.cs
public class Order
{
    public int Id { get; set; }
    public string Name { get; set; }
}
```

```
// Repositories/StatusRepository.cs
public class StatusRepository
{
    public List<Models.Status> GetStatuses()
    {
        return new List<Models.Status>
        {
            new Models.Status { Id = 0, Name = "(odabir statusa)" },
            new Models.Status { Id = 1, Name = "Zauzeto" },
            new Models.Status { Id = 2, Name = "Rezervirano" },
            new Models.Status { Id = 3, Name = "Slobodno" },
        };
    }
}

// Repositories/CityRepository.cs
public class CityRepository
{
    private readonly string _connectionString;

    public CityRepository()
    {
        _connectionString = ConfigurationManager.ConnectionStrings["rwadb"].ConnectionString;
    }

    public List<Admin.Models.City> GetCities()
    {
        var ds = SqlHelper.ExecuteDataset(
            _connectionString,
            CommandType.StoredProcedure,
            "dbo.GetCities");

        var cityList = new List<Admin.Models.City>();
        cityList.Add(new Admin.Models.City { Id = 0, Name = "(odabir grada)" });
        foreach (DataRow row in ds.Tables[0].Rows)
        {
            var city = new Admin.Models.City();
            city.Id = Convert.ToInt32(row["ID"]);
            city.Guid = Guid.Parse(row["Guid"].ToString());
            city.Name = row["Name"].ToString();
            cityList.Add(city);
        }

        return cityList;
    }
}

// Repositories/OrderRepository.cs
public class OrderRepository
{
    public List<Models.Order> GetOrders()
    {
        return new List<Models.Order>
        {
            new Models.Order { Id = 0, Name = "(kriterij sortiranja)" },
            new Models.Order { Id = 1, Name = "BrojSoba" },
            new Models.Order { Id = 2, Name = "BrojOdraslih" },
            new Models.Order { Id = 3, Name = "BrojDjece" },
            new Models.Order { Id = 4, Name = "Cijena" },
        };
    }
}
```

- sada možemo koristiti te repozitorije u listi apartmana, pa ih inicijalizirajmo u konstruktoru

```
public partial class ApartmentList : System.Web.UI.Page
{
    private readonly ApartmentRepository _apartmentRepository;
    private readonly StatusRepository _statusRepository;
    private readonly CityRepository _cityRepository;
    private readonly OrderRepository _orderRepository;

    public ApartmentList()
    {
        _apartmentRepository = new ApartmentRepository();
        _statusRepository = new StatusRepository();
        _cityRepository = new CityRepository();
        _orderRepository = new OrderRepository();
    }
    ...
}
```

- probajmo bindati kolekcije dobivene iz repozitorija na dropdowne

```
protected void Page_Load(object sender, EventArgs e)
{
    ddlStatus.DataSource = _statusRepository.GetStatuses();
    ddlStatus.DataBind();

    ddlCity.DataSource = _cityRepository.GetCities();
    ddlCity.DataBind();

    ddlOrder.DataSource = _orderRepository.GetOrders();
    ddlOrder.DataBind();

    gvListaApartmana.DataSource =
        _apartmentRepository.GetApartments(null, null, null);
    gvListaApartmana.DataBind();
}
```

> *Isprobajmo. To ne radi dobro jer nismo naveli DataValueField i DataTextField za binding, zato se u samim kontrolama prikazuje samo naziv modela tipa* `Admin.Model.Status`

- stavimo na svaki `asp:DropDownList` atribute koji označavaju što se koristi kao vrijednosta a što kao prikaz podatka
  - vrijednost se postavlja atributom `DataValueField="Id"`
  - prikaz se postavlja atributom `DataTextField="Name"`

> *Isprobajmo. Sada radi, s jednom iznimkom. Kada se promijeni odabir u dropdownu, ništa se ne događa.*

- dodajmo `OnSelectedIndexChanged` event na svaki `asp:DropDownList`
- još je bitno dodati `AutoPostBack="True"`, inače neće biti postbacka na promjenu odabira

> *Znači kod svake promjene odabira moramo napraviti ponovni data binding ili "rebind". Na taj način ćemo primijeniti promjenu odabira.*

- dodajmo metodu `RebindApartments()` koja će biti pozvana kada treba napraviti ponovni binding radi promjene odabira; dodatno, moramo i proslijediti te odabire repozitoriju, što znači da moramo promijeniti metodu repozitorija `GetApartments()`

```
private void RebindApartments()
{
    var status = int.Parse(ddlStatus.SelectedValue);
    var cityId = int.Parse(ddlCity.SelectedValue);
    var order = int.Parse(ddlOrder.SelectedValue);

    gvListaApartmana.DataSource =
        _apartmentRepository.GetApartments(status, cityId, order);
    gvListaApartmana.DataBind();
}
```

```
public List<Admin.Models.Apartment> GetApartments(
    int? statusId,
    int? cityId,
    int? order)
```

- ... i te iste parametre dalje proslijediti u proceduru; parametri se prosljeđuju u proceduru samo ako su postavljeni:

```
var commandParameters = new List<SqlParameter>();

if (statusId.HasValue && statusId.Value != 0)
    commandParameters.Add(new SqlParameter("@statusId", statusId));

if (cityId.HasValue && cityId.Value != 0)
    commandParameters.Add(new SqlParameter("@cityId", cityId));

if (order.HasValue && order.Value != 0)
    commandParameters.Add(new SqlParameter("@order", order));

var ds = SqlHelper.ExecuteDataset(
    _connectionString,
    CommandType.StoredProcedure,
    "dbo.GetApartments",
    commandParameters.ToArray());
```

- funkciju `RebindApartments()` pozovimo sada na:
    - svaku promjenu odabira, npr.

```
protected void ddlStatus_SelectedIndexChanged(object sender, EventArgs e)
{
    RebindApartments();
}
```

    - `Page_Load()`, umjesto "ručnog" bindinga, s time da rebind treba raditi samo na postback inače će kod svakog događaja ponovno biti napunjen isti set podataka iz baze (reset state), što će raditi probleme s funkcionalnosti dropdowna

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        ddlStatus.DataSource = _statusRepository.GetStatuses();
        ddlStatus.DataBind();

        ddlCity.DataSource = _cityRepository.GetCities();
        ddlCity.DataBind();

        ddlOrder.DataSource = _orderRepository.GetOrders();
        ddlOrder.DataBind();

        RebindApartments();
    }
}
```

- isprobajmo filtriranje i sortiranje

5.5. Lista apartmana - cookies za filtriranje i sortiranje

Da bismo spremili podatke u cookie, koristit ćemo klasu `HttpCookie`. Pojednostavniti ćemo si život tako da implementiramo spremanje sve tri `DropDownList` kontrole u jednu metodu `FormToCookie()`. S druge strane, kada nam zatreba čitanje implementirat ćemo `CookieToForm()`.

- želimo kod odabira/promjene bilo kojeg dropdowna spremiti vrijednost tog odabira u cookie

```
private void FormToCookie()
{
    HttpCookie cookie = Request.Cookies["ApartmentList"];
    if (cookie == null)
        cookie = new HttpCookie("ApartmentList");

    cookie["Status"] = ddlStatus.SelectedValue;
    cookie["City"] = ddlCity.SelectedValue;
    cookie["Order"] = ddlOrder.SelectedValue;
    cookie.Expires = DateTime.Now.AddMinutes(30);
    Response.SetCookie(cookie);
}

protected void ddlStatus_SelectedIndexChanged(object sender, EventArgs e)
{
    RebindApartments();
    FormToCookie();
}

protected void ddlCity_SelectedIndexChanged(object sender, EventArgs e)
{
    RebindApartments();
    FormToCookie();
}

protected void ddlOrder_SelectedIndexChanged(object sender, EventArgs e)
{
    RebindApartments();
    FormToCookie();
}
```

- provjerimo u debuggeru postavlja li se cookie (promijeniti par puta odabir)
- na koncu, želimo kod inicijalnog loada stranice (ne-postback) učitati tu vrijednost i postaviti je

```
private void CookieToForm()
{
    HttpCookie cookie = Request.Cookies["ApartmentList"];
    if (cookie != null)
    {
        if (cookie["Status"] != null)
        {
            ddlStatus.SelectedValue = cookie["Status"];
        }
        if (cookie["City"] != null)
        {
            ddlCity.SelectedValue = cookie["City"];
        }
        if (cookie["Order"] != null)
        {
            ddlOrder.SelectedValue = cookie["Order"];
        }
    }
}

protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        ...

        CookieToForm();

        RebindApartments();
    }
}
```

- provjerimo u debuggeru čita li se cookie i postavljaju li se vrijednosti dropdowna kada odemo i vratimo se na stranicu
- provjerimo radi li to čak i kada se odlogiramo i logiramo ponovno

6. Dodavanje novog apartmana

U svrhu dodavanja apartmana dodat ćemo novu formu. Nju ćemo kasnije koriastiti također i za promjenu apartmana, na jednostavan način:

- ako u formi postoji id za apartman, radi se o promjeni
- inače se radi o dodavanju

6.1. Dodavanje novog apartmana - priprema web forme

- dodati novu formu `ApartmentEditor.aspx` koja ima Master Page
- dodati ispod gridview-a u listi apaprtmana gumb koji vodi na tu formu, npr.

```
<div><asp:LinkButton ID="lbApartmentEditor" runat="server" Text="Add apartment" CssClass="btn btn-primary" OnClick="lbApartmentEdito
```

```
protected void lbApartmentEditor_Click(object sender, EventArgs e)
{
    Response.Redirect("ApartmentEditor.aspx");
}
```

- neka forma `ApartmentEditor.aspx` ima u sebi HTML kod za karakteristike apartmana

```
<div class="row">
    <div class="col-md-6">
        <div class="form-group">
            <label>Vlasnik</label>
            <asp:DropDownList ID="ddlApartmentOwner" runat="server" CssClass="form-control"></asp:DropDownList>
        </div>
        <div class="form-group">
            <label>Status</label>
            <asp:DropDownList ID="ddlStatus" runat="server" CssClass="form-control"></asp:DropDownList>
        </div>
        <div class="form-group">
            <label>Naziv</label>
            <asp:TextBox ID="tbName" runat="server" CssClass="form-control"></asp:TextBox>
        </div>
        <div class="form-group">
            <label>Adresa</label>
            <asp:TextBox ID="tbAddress" runat="server" CssClass="form-control"></asp:TextBox>
        </div>
        <div class="form-group">
            <label>Grad</label>
            <asp:DropDownList ID="ddlCity" runat="server" CssClass="form-control"></asp:DropDownList>
        </div>
        <div class="form-group">
            <label>Cijena</label>
            <div class="input-group">
                <span class="input-group-addon" id="sizing-addon1">€</span>
                <asp:TextBox ID="tbPrice" runat="server" CssClass="form-control"></asp:TextBox>
            </div>
        </div>
        <div class="form-group">
            <label>Broj odraslih</label>
            <asp:TextBox ID="tbMaxAdults" runat="server" TextMode="Number" CssClass="form-control"></asp:TextBox>
        </div>
        <div class="form-group">
            <label>Broj djece</label>
            <asp:TextBox ID="tbMaxChildren" runat="server" TextMode="Number" CssClass="form-control"></asp:TextBox>
        </div>
        <div class="form-group">
            <label>Broj soba</label>
            <asp:TextBox ID="tbTotalRooms" runat="server" TextMode="Number" CssClass="form-control"></asp:TextBox>
        </div>
        <div class="form-group">
            <label>Udaljenost od plaže</label>
            <asp:TextBox ID="tbBeachDistance" runat="server" TextMode="Number" CssClass="form-control"></asp:TextBox>
        </div>
    </div>
    <div class="col-md-6">
        <!-- TU IDE ODABIR TAGOVA -->
    </div>
</div>
```

- isprobati radi li redirect na novu formu i prikazuje li se ona ispravno
- dodajmo sada binding za vlasnika, status i grad
    - kreirajmo model klasu ApartmentOwner.cs (Id, Name)
    - Status model već imamo, pa ne trebamo novu klasu
    - City model već imamo, pa ne trebamo novu klasu
    - kreirajmo ApartmentOwnerRepository i dodajmo ga u konstruktoru od `ApartmentEditor.aspx`

```
public class ApartmentOwnerRepository
{
    private readonly string _connectionString;

    public ApartmentOwnerRepository()
    {
        _connectionString = ConfigurationManager.ConnectionStrings["rwadb"].ConnectionString;
    }

    public List<Admin.Models.ApartmentOwner> GetApartmentOwners()
    {
        var ds = SqlHelper.ExecuteDataset(
            _connectionString,
            CommandType.StoredProcedure,
            "dbo.GetApartmentOwners");

        var ownerList = new List<Admin.Models.ApartmentOwner>();
        ownerList.Add(new Admin.Models.ApartmentOwner { Id = 0, Name = "(odabir vlasnika)" });
        foreach (DataRow row in ds.Tables[0].Rows)
        {
            var owner = new Admin.Models.ApartmentOwner();
            owner.Id = Convert.ToInt32(row["ID"]);
            owner.Name = row["Name"].ToString();
            ownerList.Add(owner);
        }

        return ownerList;
    }
}
```

- dodajmo StatusRepository u `ApartmentEditor.aspx`
- dodajmo CityRepository u `ApartmentEditor.aspx`
- dodajmo i implementirajmo RebindCities(), RebindStatuses() i RebindApartmentOwners()

```
private void RebindApartmentOwners()
{
    ddlApartmentOwner.DataSource = _apartmentOwnerRepository.GetApartmentOwners();
    ddlApartmentOwner.DataBind();
}

private void RebindCities()
{
    ddlCity.DataSource = _cityRepository.GetCities();
    ddlCity.DataBind();
}

private void RebindStatuses()
{
    ddlStatus.DataSource = _statusRepository.GetStatuses();
    ddlStatus.DataBind();
}
```

- pozivamo sve Rebind* metode u Page_Load(), ako nije Postback

> *Napomena: binding će raditi ispravno samo ako imamo `DataValueField` i `DataTextField` pa će ih trebati dodati*

- provjerimo povlače li se `Vlasnik`, `Status` i `Grad` podaci iz baze u dropdown-e

6.2. Dodavanje novog apartmana - dodavanje tagova

Ovaj dio je kompliciran jer se tagovi dodaju u viewstate sve dok se ne spremi novi apartman. Treba nam prvo **podatkovni model** za tagove, te repozitorij. Dodavanje ćemo vršiti iz dropdown liste, a koja podatke treba dobivati (bindati) iz tags repozitorija.

```
// Models/Tag.cs
public class Tag
{
    public int Id { get; set; }
    public string Name { get; set; }
}
```

```
public class TagRepository
{
    private readonly string _connectionString;

    public TagRepository()
    {
        _connectionString = ConfigurationManager.ConnectionStrings["rwadb"].ConnectionString;
    }

    public List<Admin.Models.Tag> GetTags()
    {
        var ds = SqlHelper.ExecuteDataset(
            _connectionString,
            CommandType.StoredProcedure,
            "dbo.GetTags");

        var tagList = new List<Admin.Models.Tag>();
        tagList.Add(new Admin.Models.Tag { Id = 0, Name = "(odabir taga)" });
        foreach (DataRow row in ds.Tables[0].Rows)
        {
            var tag = new Admin.Models.Tag();
            tag.Id = Convert.ToInt32(row["ID"]);
            tag.Name = row["Name"].ToString();
            tagList.Add(tag);
        }

        return tagList;
    }
}
```

- dodajmo **dropdown** listu tagova u `ApartmentEditor.aspx` i bindajmo je na podatke iz repozitorija (primijetite DataValueField i DataTextField)

```
<div class="form-group">
    <label>Odabir tagova</label>
    <asp:DropDownList
        ID="ddlTags"
        runat="server"
        CssClass="form-control"
        DataValueField="Id"
        DataTextField="Name">
    </asp:DropDownList>
</div>
```

```
private readonly TagRepository _tagRepository;

public ApartmentEditor()
{
    ...
    _tagRepository = new TagRepository();
}

private void RebindTags()
{
    ddlTags.DataSource = _tagRepository.GetTags();
    ddlTags.DataBind();
}

protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        ...
        RebindTags();
    }
}
```

- provjerimo povlače li se tagovi u `Odabir tagova` dropdown
- listu odabranih tagova ćemo prikazati koristeći **repeater**; dodajmo ga odmah ispod dropdowna za odabir tagova

```
<div class="form-group">
    <label>Odabir tagova</label>
    <asp:DropDownList
        ID="ddlTags"
        runat="server"
        CssClass="form-control"
        DataValueField="Id"
        DataTextField="Name">
    </asp:DropDownList>
</div>
<asp:Repeater ID="repTags" runat="server">
    <HeaderTemplate>
        <ul class="list-group">
    </HeaderTemplate>
    <ItemTemplate>
        <li class="list-group-item">
            <asp:HiddenField runat="server" ID="hidTagId" Value='<%# Eval("ID")   %>' />
            <asp:Label runat="server" ID="txtTagName" Text='<%# Eval("Name") %>' />
        </li>
    </ItemTemplate>
    <FooterTemplate>
        </ul>
    </FooterTemplate>
</asp:Repeater>
```

- u dropdown tagova dodajmo `SelectedIndexChanged` event handler te `AutoPostBack="true"` tako da kontrola reagira na promjenu dropdowna

> *primijetite da je dodan binding na ID i Name svojstva*

- sada ćemo na `SelectedIndexChanged` dodati odabrani tag iz dropdowna u repeater

```
protected void ddlTags_SelectedIndexChanged(object sender, EventArgs e)
{
    var tags = GetRepeaterTags(); // TODO
    var newTag = GetSelectedTag(); // TODO
    if (tags.Any(x => x.Id == newTag.Id))
        return;

    tags.Add(newTag);

    repTags.DataSource = tags;
    repTags.DataBind();
}
```

- **problem s repeaterom**: Podaci koji se trebaju dodati i prikazati u repeateru nisu već u bazi, nego se održavaju u viewstateu. Repeater za to nema najbolju podršku - on podatke čuva unutar kontrola koje su u `<ItemTemplate>` sekciji (HiddenField, Label) a ne u originalnom obliku (model Tag). Da bismo izvukli model podataka, te kontrole onda treba na adekvatan način pronaći i pročitati podatke iz njih, te rekreirati model.

```
private List<Models.Tag> GetRepeaterTags()
{
    var repTagsItems = repTags.Items;
    var tags = new List<Models.Tag>();
    foreach (RepeaterItem item in repTagsItems)
    {
        var tag = new Models.Tag();
        tag.Id = int.Parse((item.FindControl("hidTagId") as HiddenField).  Value);
      tag.Name = (item.FindControl("txtTagName") as Label).Text;
        tags.Add(tag);
    }
    return tags;
}
```

- i još nam preostaje kreiranje novog taga iz onoga što je korisnik izabrao iz dropdowna

```
private Models.Tag GetSelectedTag()
{
    var selectedValue = ddlTags.SelectedItem.Value;
    var newTag = new Models.Tag
    {
        Id = int.Parse(ddlTags.SelectedItem.Value),
        Name = ddlTags.SelectedItem.Text
    };
    return newTag;
}
```

> *Sada će `SelectedIndexChanged` handler dohvatiti odabrani tag u dropdownu i listu tagova koji su već u repeateru. Provjeriti će je li odabrani tag već u repeateru i ako nije dodati će ga. Na koncu je potreban rebind repeatera da bi se prikazala promijenjena lista tagova u njemu.*

  - provjerimo dodaju li se tagovi kod odabira iz dropdowna

6.3. Dodavanje novog apartmana - brisanje tagova

- implementirati ćemo i brisanje tagova - za početak dodajmo `LinkButton` stiliziran Bootstrap kanticom za smeće, odmah ispod labele taga

```
<li class="list-group-item">
    <asp:HiddenField runat="server" ID="hidTagId" Value='<%# Eval("ID")   %>' />
    <asp:Label runat="server" ID="txtTagName" Text='<%# Eval("Name") %>' />
    <asp:LinkButton runat="server" ID="btnDeleteTag" CssClass="btn">
        <span class="glyphicon glyphicon-trash"></span>
    </asp:LinkButton>
</li>
```

- provjerimo pokazuje li se gumb na svakom tagu koji se doda

- sada spojimo gumb s event handlerom

```
OnClick="btnDeleteTag_Click"
```

```
protected void btnDeleteTag_Click(object sender, EventArgs e)
{
    var tags = GetRepeaterTags();

    // Nije lako naći pravu kontrolu
    var lbSender = (LinkButton)sender;
    var parentItem = (RepeaterItem)lbSender.Parent;
    var hidTagId = (HiddenField)parentItem.FindControl("hidTagId");
    var tagId = int.Parse(hidTagId.Value);

    var existingTag = tags.FirstOrDefault(x => x.Id == tagId);
    tags.Remove(existingTag);

    repTags.DataSource = tags;
    repTags.DataBind();
}
```

- provjerimo radi li brisanje na ispravan način

6.4. Dodavanje novog apartmana - spremanje podataka

- dodajmo pred kraj sadržaja gumb za spremanje koji poziva prazan event handler

```
<div><asp:LinkButton ID="lblSave" runat="server" Text="Spremi" CssClass="btn btn-primary" OnClick="lblSave_Click" /></div>
```

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

}
```

- sada dodajmo u `Apartment` model listu tagova tako da bismo i njih mogli spremiti

```
public List<Tag> Tags { get; set; }
```

> *Da bismo spremili apartman s listom tagova potrebno je kreirati proceduru u bazi koja će spremiti jedan apartman i više tagova za taj apartman. Procedura u bazi podataka je već implementirana. Koristimo se user-defined table type `dbo.KeyList` da bi se to omogućilo. Pozivanje procedure s takvim tipom iz ADO.NET-a izgleda specifično (kod ide u `ApartmentRepository`). Proučite kod u `CreateApartment` metodi.*

```
// Repositories/ApartmentRepository.cs
public void CreateApartment(Admin.Models.Apartment apartment)
{
    var commandParameters = new List<SqlParameter>();
    commandParameters.Add(new SqlParameter("@guid", apartment.Guid));
    commandParameters.Add(new SqlParameter("@ownerId", apartment.OwnerId));
    commandParameters.Add(new SqlParameter("@typeId", apartment.TypeId));
    commandParameters.Add(new SqlParameter("@statusId", apartment.StatusId));
    commandParameters.Add(new SqlParameter("@cityId", apartment.CityId));
    commandParameters.Add(new SqlParameter("@address", apartment.Address));
    commandParameters.Add(new SqlParameter("@name", apartment.Name));
    commandParameters.Add(new SqlParameter("@price", apartment.Price));
    commandParameters.Add(new SqlParameter("@maxAdults", apartment.MaxAdults));
    commandParameters.Add(new SqlParameter("@maxChildren", apartment.MaxChildren));
    commandParameters.Add(new SqlParameter("@totalRooms", apartment.TotalRooms));
    commandParameters.Add(new SqlParameter("@beachDistance", apartment.BeachDistance));

    DataTable dtTags = new DataTable();
    dtTags.Columns.AddRange(
        new DataColumn[1] {
            new DataColumn("Key", typeof(int)) });

    foreach (var tag in apartment.Tags)
        dtTags.Rows.Add(tag.Id);

    commandParameters.Add(new SqlParameter("@tags", dtTags));

    SqlHelper.ExecuteNonQuery(
        _connectionString,
        CommandType.StoredProcedure,
        "dbo.CreateApartment",
        commandParameters.ToArray());
}
```

Da bismo spremili novi `Apartment` objekt u bazu kao novi redak, potrebno je prvo sakupiti sve podatke i onda pozvati petodu repozitorija za spremanje.

```
private Models.Apartment GetFormApartment()
{
    int statusId = int.Parse(ddlStatus.SelectedValue);

    int? cityId = int.Parse(ddlCity.SelectedValue);
    if (cityId == 0)
        cityId = null;

    int ownerId = int.Parse(ddlApartmentOwner.SelectedValue);

    decimal price = decimal.Parse(tbPrice.Text);

    int? maxAdults = null;
    if (!string.IsNullOrEmpty(tbMaxAdults.Text))
        maxAdults = int.Parse(tbMaxAdults.Text);

    int? maxChildren = null;
    if (!string.IsNullOrEmpty(tbMaxChildren.Text))
        maxChildren = int.Parse(tbMaxChildren.Text);

    int? totalRooms = null;
    if (!string.IsNullOrEmpty(tbTotalRooms.Text))
        totalRooms = int.Parse(tbTotalRooms.Text);

    int? beachDistance = null;
    if (!string.IsNullOrEmpty(tbBeachDistance.Text))
        beachDistance = int.Parse(tbBeachDistance.Text);

    return
        new Models.Apartment
        {
            // Id kreira baza
            Guid = Guid.NewGuid(),
            // CreatedAt kreira baza
            // DeletedAt mora biti prazan
            OwnerId = ownerId,
            TypeId = 999, // hardkodirano
            StatusId = statusId,
            CityId = cityId,
            Address = tbAddress.Text,
            Name = tbName.Text,
            Price = price,
            MaxAdults = maxAdults,
            MaxChildren = maxChildren,
            TotalRooms = totalRooms,
            BeachDistance = beachDistance,
            Tags = GetRepeaterTags()
        };
}
```

- sada možemo dodati kod u `lblSave_Click` koji kreira `Apartment` objekt i sprema ga (hint: trebat će nam i `_apartmentRepository`)

```
protected void lblSave_Click(object sender, EventArgs e)
{
    var apartment = GetFormApartment();
    _apartmentRepository.CreateApartment(apartment);

    Response.Redirect("ApartmentList.aspx");
}
```

  - provjerimo sprema li se apartman u bazu, zajedno s tagovima

7. Ažuriranje apartmana

Iskoristimo upravo napravljenu formu za dodavanje apartmana da bismo implementirali ažuriranje. Kao što smo već napomenuli, prvo ćemo provjeriti je li poslan query string u stranicu i ako jest dohvatit ćemo apartman iz baze.

7.1. Ažuriranje apartmana - dohvat podataka

Za to nam treba implementirana metoda u repozitoriju. Metoda je jako slična metodi za dohvat liste apartmana.

```
public Admin.Models.Apartment GetApartment(int id)
{
    var commandParameters = new List<SqlParameter>();
    commandParameters.Add(new SqlParameter("@id", id));

    var ds = SqlHelper.ExecuteDataset(
        _connectionString,
        CommandType.StoredProcedure,
        "dbo.GetApartment",
        commandParameters.ToArray());

    var row = ds.Tables[0].Rows[0];
    var ap = new Admin.Models.Apartment();
    ap.Id = Convert.ToInt32(row["ID"]);
    ap.Guid = Guid.Parse(row["Guid"].ToString());
    ap.CreatedAt = Convert.ToDateTime(row["CreatedAt"]);
    ap.DeletedAt =
        row["DeletedAt"] != DBNull.Value ?
            (DateTime?)Convert.ToDateTime(row["DeletedAt"]) :
            null;
    ap.OwnerId = Convert.ToInt32(row["OwnerId"]);
    ap.OwnerName = row["OwnerName"].ToString();
    ap.TypeId = Convert.ToInt32(row["TypeId"]);
    ap.StatusId = Convert.ToInt32(row["StatusId"]);
    ap.StatusName = row["StatusName"].ToString();
    ap.CityId =
        row["CityId"] != DBNull.Value ?
            (int?)Convert.ToInt32(row["CityId"]) :
            null;
    ap.CityName = row["CityName"]?.ToString();
    ap.Address = row["Address"].ToString();
    ap.Name = row["Name"].ToString();
    ap.Price = Convert.ToDecimal(row["Price"]);
    ap.MaxAdults =
        row["MaxAdults"] != DBNull.Value ?
            (int?)Convert.ToInt32(row["MaxAdults"]) :
            null;
    ap.MaxChildren =
        row["MaxChildren"] != DBNull.Value ?
            (int?)Convert.ToInt32(row["MaxChildren"]) :
            null;
    ap.TotalRooms =
        row["TotalRooms"] != DBNull.Value ?
            (int?)Convert.ToInt32(row["TotalRooms"]) :
            null;
    ap.BeachDistance =
        row["BeachDistance"] != DBNull.Value ?
            (int?)Convert.ToInt32(row["BeachDistance"]) :
            null;

    return ap;
}
```

- sada u `Page_Load()` stranice `ApartmentEditor.aspx` provjerimo je li poslan query string i ako jest onda dohvatimo podatke (model) iz baze.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        string qryStrId = Request.QueryString["id"];

        int? id = null;
        if(!string.IsNullOrEmpty(qryStrId))
            id = int.Parse(qryStrId);

        if (id.HasValue)
        {
            var dbApartment = _apartmentRepository.GetApartment(id.Value);
            SetExistingApartment(dbApartment);
        }
        RebindApartmentOwners();
        RebindCities();
        RebindStatuses();
        RebindTags();
    }
}
```

- metoda `SetExistingApartment()` je tu da bi nam postavila podatke iz modela u kontrole stranice

```
private void SetExistingApartment(Models.Apartment apartment)
{
    ddlStatus.SelectedValue = apartment.StatusId.ToString();
    ddlApartmentOwner.SelectedValue = apartment.OwnerId.ToString();
    tbName.Text = apartment.Name;
    tbAddress.Text = apartment.Address;
    ddlCity.SelectedValue = apartment.CityId.ToString();
    tbPrice.Text = apartment.Price.ToString();
    tbMaxAdults.Text = apartment.MaxAdults?.ToString();
    tbMaxChildren.Text = apartment.MaxChildren?.ToString();
    tbTotalRooms.Text = apartment.TotalRooms?.ToString();
    tbBeachDistance.Text = apartment.BeachDistance?.ToString();

    repTags.DataSource = apartment.Tags;
    repTags.DataBind();
}
```

- provjeriti prikazuju li se podaci, npr. pomoću URL-a `http://localhost:{port}/ApartmentEditor.aspx?Id=1`
- tagovi se ne pokazuju jer nova metoda `GetApartment()` repozitorija ne dohvaća te podatke

7.2. Ažuriranje apartmana - dohvat tagova

- napravimo u `ApartmentRepository` repozitoriju novu metodu koja dohvaća tagove

```
public List<Admin.Models.Tag> GetApartmentTags(int apartmentId)
{
    var commandParameters = new List<SqlParameter>();
    commandParameters.Add(new SqlParameter("@apartmentId", apartmentId));
    var ds = SqlHelper.ExecuteDataset(
        _connectionString,
        CommandType.StoredProcedure,
        "dbo.GetApartmentTags",
        commandParameters.ToArray());
    var tags = new List<Admin.Models.Tag>();
    foreach (DataRow row in ds.Tables[0].Rows)
    {
        tags.Add(new Models.Tag
        {
            Id = Convert.ToInt32(row["Id"]),
            Name = row["Name"].ToString(),
        });
    }
    return tags;
}
```

- sada je možemo upotrijebiti prije `SetExistingApartment()`

```
var dbApartment = _apartmentRepository.GetApartment(id.Value);
dbApartment.Tags = _apartmentRepository.GetApartmentTags(id.Value);
SetExistingApartment(dbApartment);
```

- provjerimo radi li povlačenje podataka o tagovima ispravno

7.3. Ažuriranje apartmana - spremanje podataka

- sada možemo spremiti promijenjeni apartman umjesto da kreiramo novi
- za spremanje trebamo novu metodu u repozitoriju

```
public void UpdateApartment(Admin.Models.Apartment apartment)
{
    var commandParameters = new List<SqlParameter>();
    commandParameters.Add(new SqlParameter("@id", apartment.Id));
    commandParameters.Add(new SqlParameter("@guid", apartment.Guid));
    commandParameters.Add(new SqlParameter("@ownerId", apartment.OwnerId));
    commandParameters.Add(new SqlParameter("@typeId", apartment.TypeId));
    commandParameters.Add(new SqlParameter("@statusId", apartment.StatusId));
    commandParameters.Add(new SqlParameter("@cityId", apartment.CityId));
    commandParameters.Add(new SqlParameter("@address", apartment.Address));
    commandParameters.Add(new SqlParameter("@name", apartment.Name));
    commandParameters.Add(new SqlParameter("@price", apartment.Price));
    commandParameters.Add(new SqlParameter("@maxAdults", apartment.MaxAdults));
    commandParameters.Add(new SqlParameter("@maxChildren", apartment.MaxChildren));
    commandParameters.Add(new SqlParameter("@totalRooms", apartment.TotalRooms));
    commandParameters.Add(new SqlParameter("@beachDistance", apartment.BeachDistance));

    DataTable dtTags = new DataTable();
    dtTags.Columns.AddRange(
        new DataColumn[1] {
            new DataColumn("Key", typeof(int)) });

    foreach (var tag in apartment.Tags)
        dtTags.Rows.Add(tag.Id);

    commandParameters.Add(new SqlParameter("@tags", dtTags));

    SqlHelper.ExecuteNonQuery(
        _connectionString,
        CommandType.StoredProcedure,
        "dbo.UpdateApartment",
        commandParameters.ToArray());

}
```

> *Kod spremanja trebamo provjeriti radimo li trenutno kreiranje novog apartmana ili ažuriranje postojećeg. Najjednostavnije je provjeriti postoji li varijabla `id` u query stringu.*

```
protected void lblSave_Click(object sender, EventArgs e)
{
    var isNewApartment = (Request.QueryString["id"] == null);

    if (isNewApartment)
    {
        var apartment = GetFormApartment();
        _apartmentRepository.CreateApartment(apartment);
    }
    else
    {
        var apartment = GetFormApartment();
        apartment.Id = int.Parse(Request.QueryString["id"]);
        _apartmentRepository.UpdateApartment(apartment);
    }

    Response.Redirect("ApartmentList.aspx");
}
```

- isprobajmo funkcionalnost ažuriranja
- provjerimo radi li i dalje funkcionalnost dodavanja

7.4. Ažuriranje apartmana - "Uredi" gumb u GridView kontroli

- dodajmo u gridview u ApartmentsList.aspx novi stupac koji nas vodi na uređivanje apartmana

```
<asp:GridView ID="gvListaApartmana" ...>
    <Columns>
        ...
        <asp:TemplateField HeaderText="">
            <ItemTemplate>
                <asp:HyperLink ID="hlEditor" runat="server" CssClass="btn btn-primary" Text="Uredi" NavigateUrl='<%# Eval("Id",
            </ItemTemplate>
        </asp:TemplateField>
    </Columns>
</asp:GridView>
```

- isprobajmo novi link na uređivanje apartmana
- dodajmo u ApartmentEditor.aspx gumb za navigaciju natrag na listu

8. Ažuriranje apartmana - upload slika

Koncept "upload slika" ne odnosi se samo na upload slika na server, već i na uređivanje podataka o tim slikama, kao što su naziv slike i odabir reprezentativne slike. Osim dodavanja slike trebamo i mogućnost ažuriranja i brisanja slike. Opisati ćemo proces gdje su slike spremljene na disk a njihovi metapodaci (relativna putanja do slike, naziv, reprezentativnost) u bazi podataka.

8.1. Upload slika - model i repozitorij

- kreirajmo model za sliku apartmana
  - napomena: `Id` je nulabilan jer kod kreiranja slike nije poznat, obzirom da ga dodjeljuje baza podataka
  - napomena: `DoDelete` je zastavica koja označava da ćemo obrisati sliku; zastavica kao takva neće biti spremljena u bazu s obzirom da će slika biti obrisana, ona će samo u procesu služiti kao naznaka brisanja

```
public class ApartmentPicture
{
    public int? Id { get; set; }
    public string Path { get; set; }
    public string Name { get; set; }
    public bool IsRepresentative { get; set; }
    public bool DoDelete { get; set; }
}
```

- slike pripadaju apartmanu, pa ako želimo spremiti slike s apartmanom moramo promijeniti i model apartmana tako da mu dodamo kolekciju slika

```
public class Apartment
{
    ...
    public List<ApartmentPicture> ApartmentPictures { get; set; }
}
```

- sada u repozitoriju moramo omogućiti dohvat i spremanje slika; kreirajmo metodu repozitorija za dohvat

```
public List<Admin.Models.ApartmentPicture> GetApartmentPictures(int apartmentId)
{
    var commandParameters = new List<SqlParameter>();
    commandParameters.Add(new SqlParameter("@apartmentId", apartmentId));

    var ds = SqlHelper.ExecuteDataset(
        _connectionString,
        CommandType.StoredProcedure,
        "dbo.GetApartmentPictures",
        commandParameters.ToArray());

    var pics = new List<Admin.Models.ApartmentPicture>();
    foreach (DataRow row in ds.Tables[0].Rows)
    {
        pics.Add(new Models.ApartmentPicture
        {
            Id = Convert.ToInt32(row["Id"]),
            Path = row["Path"].ToString(),
            Name = row["Name"].ToString(),
            IsRepresentative = bool.Parse(row["IsRepresentative"].ToString())
        });
    }

    return pics;
}
```

- zatim, kod spremanja apartmana (repozitorij, `CreateApartment()`, prije `SqlHelper.ExecuteNonQuery`) dodajmo i dio koji sprema slike iz modela

> *U slučaju da je korisnik odabrao da će obrisati sliku (zastavica `DoDelete`), jednostavno nećemo proslijediti tu sliku u proceduru te će se procedura pobrinuti za njeno brisanje iz baze.*

```
DataTable dtPics = new DataTable();
dtPics.Columns.AddRange(
    new DataColumn[] {
        new DataColumn("Id", typeof(int)),
        new DataColumn("Path", typeof(string)),
        new DataColumn("Name", typeof(string)),
        new DataColumn("IsRepresentative", typeof(bool)),
        new DataColumn("DoDelete", typeof(bool)),
    });

foreach (var apartmentPicture in apartment.ApartmentPictures)
{
    if (!apartmentPicture.DoDelete)
    {
        dtPics.Rows.Add(
            apartmentPicture.Id,
            apartmentPicture.Path,
            apartmentPicture.Name,
            apartmentPicture.IsRepresentative,
            apartmentPicture.DoDelete);
    }
}

commandParameters.Add(new SqlParameter("@pictures", dtPics));
```

8.2. Upload slika - korisničko sučelje

Najveći dio posla je u korisničkom sučelju `ApartmentEditor.aspx`. Treba nam mogućnost uploada slika, te repeater kontrola za listu uploadanih slika i njihovih podataka.

- dodajmo neposredno prije gumba za spremanje mogućnost uploada, za što ćemo koristiti FileUpload kontrolu (npr. prije gumba za spremanje)

```
<asp:FileUpload ID="uplImages" runat="server" CssClass="hidden" AllowMultiple="true" />
<hr />
<div><asp:LinkButton ID="lblSave" runat="server" Text="Spremi" CssClass="btn btn-primary" OnClick="lblSave_Click" /></div>
```

- kontrola izgleda "sirovo" - stilizirajmo je pomoću Bootstrapa

```
<div class="container">
  <div class="form-group">
      <label class="btn btn-default">
          Upload slika
              <asp:FileUpload ID="uplImages" runat="server" CssClass="hidden" AllowMultiple="true" />
      </label>
  </div>
</div>
```

> *Kada se kontrola tako stilizira, imamo problem sa indikatorom koji treba pokazati što je točno odabrano za upload. Razlog tome je to što je upload kontrola skrivena zbog Bootstrap stila. Popravimo to malim Javascript programom koji pokazuje te podatke unutar div-a koristeći Bootstrap labele. Pokušajte odabrati par datoteka za upload - stranica vam neće dati informaciju što je odabrano.*

```
<div class="container">
  <div class="form-group">
      <label class="btn btn-default">
          Upload slika
              <asp:FileUpload ID="uplImages" runat="server" CssClass="hidden" AllowMultiple="true" OnChange="handleFileSelect(t
      </label>
      <div id="uplImageInfo"></div>
      <script>
          function handleFileSelect(files) {
              for (var i = 0; i < files.length; i++) {
                  $span = $("<span class='label label-info'></span>").text(files[i].name);
                  $('#uplImageInfo').append($span);
                  $('#uplImageInfo').append("<br />");
              }
          }
      </script>
  </div>
</div>
```

- pokušajte sada odabrati par datoteka za upload - labele se prikazuju odmah ispod gumbića

8.3. Upload slika - spremanje datoteka

Idemo sada "uhvatiti" uploadane slike sa strane servera i spremiti ih u bazu i na disk. Prvo ćemo definirati mjesto gdje se spremaju slike i staviti tu informaciju u varijablu. Zatim ćemo napraviti metodu za spremanje slika i pozvati je kod spremanja apartmana.

```
public partial class ApartmentEditor : System.Web.UI.Page
{
  ...
  private readonly string _picPath = "/Content/Pictures/";
  ...
}
```

```
private List<string> SaveUploadedImagesToDisk()
{
    var files = new List<string>();
    if (uplImages.HasFiles)
    {
        var uplImagesRoot = Server.MapPath(_picPath);
        if (!Directory.Exists(uplImagesRoot))
            Directory.CreateDirectory(uplImagesRoot);
        foreach (HttpPostedFile uploadedFile in uplImages.PostedFiles)
        {
            var uplImagePath = Path.Combine(uplImagesRoot, uploadedFile.FileName);
            uploadedFile.SaveAs(uplImagePath);
            files.Add(uploadedFile.FileName);
        }
    }
    return files;
}
```

```
protected void lblSave_Click(object sender, EventArgs e)
{
  var files = SaveUploadedImagesToDisk();
  ...
}
```

- provjerimo uploadaju li se slike u mapu `/Content/Pictures/`

8.4. Upload slika - spremanje podataka o novim slikama

Nakon što smo spremili slike na disk, trebamo ažurirati model da bi metoda koja ga sprema obavila svoj posao ispravno, to jest da bi spremila podatke o slici.

Ako pogledamo metodu `lblSave_Click()`, ona poziva:

- `GetFormApartment()` koji dohvaća podatke iz forme
- `CreateApartment()` ili `UpdateApartment()`, ovisno o tome što se radi

Plan je ovakav:

- da bismo spremili podatke o slikama prvo trebamo u `GetFormApartment()` sakupiti podatke o slikama koje već postoje (iz repeatera). Zatim trebamo dodati nove (tako smo napravili i funkcionalnost tagova).

```
private List<Models.ApartmentPicture> GetRepeaterPictures()
{
    var repApartmentPicturesItems = repApartmentPictures.Items;
    var pictures = new List<Models.ApartmentPicture>();
    foreach (RepeaterItem item in repApartmentPicturesItems)
    {
        var pic = new Models.ApartmentPicture();
        pic.Id = int.Parse((item.FindControl("hidApartmentPictureId") as HiddenField).Value);
        pic.Name = (item.FindControl("txtApartmentPicture") as TextBox).Text;
        pic.Path = (item.FindControl("imgApartmentPicture") as Image).ImageUrl;
        pic.IsRepresentative = (item.FindControl("cbIsRepresentative") as CheckBox).Checked;
        pic.DoDelete = (item.FindControl("cbDelete") as CheckBox).Checked;
        pictures.Add(pic);
    }
    return pictures;
}
```

```
private Models.Apartment GetFormApartment()
{
    ...
    return
        new Models.Apartment
        {
            ...
            Tags = GetRepeaterTags(),
            ApartmentPictures = GetRepeaterPictures()
        };
}
```

- s obzirom da smo u `CreateApartment()` već implementirali kreiranje slika, sada ćemo to napraviti i u `UpdateApartment()`
- dodajmo kod odmah prije poziva `SqlHelper.ExecuteNonQuery()`

```
DataTable dtPics = new DataTable();
dtPics.Columns.AddRange(
    new DataColumn[] {
        new DataColumn("Id", typeof(int)),
        new DataColumn("Path", typeof(string)),
        new DataColumn("Name", typeof(string)),
        new DataColumn("IsRepresentative", typeof(bool)),
        new DataColumn("DoDelete", typeof(bool)),
    });

foreach (var apartmentPicture in apartment.ApartmentPictures)
{
    if (!apartmentPicture.DoDelete)
    {
        dtPics.Rows.Add(
            apartmentPicture.Id,
            apartmentPicture.Path,
            apartmentPicture.Name,
            apartmentPicture.IsRepresentative,
            apartmentPicture.DoDelete);
    }
}

commandParameters.Add(new SqlParameter("@pictures", dtPics));
```

- izmijenimo sada kod za spremanje na sljedeći način

```
protected void lblSave_Click(object sender, EventArgs e)
{
    var files = SaveUploadedImagesToDisk();
    var apartmentPictures =
        files.Select(x => new ApartmentPicture
        {
            Path = x,
            Name = Path.GetFileNameWithoutExtension(x),
            IsRepresentative = false
        }).ToList();
    var isNewApartment = (Request.QueryString["id"] == null);
    if (isNewApartment)
    {
        var apartment = GetFormApartment();
        apartment.ApartmentPictures = apartmentPictures;
        _apartmentRepository.CreateApartment(apartment);
        Response.Redirect($"ApartmentList.aspx");
    }
    else
    {
        var apartment = GetFormApartment();
        apartment.Id = int.Parse(Request.QueryString["id"]);
        apartment.ApartmentPictures.AddRange(apartmentPictures);
        _apartmentRepository.UpdateApartment(apartment);
        Response.Redirect($"ApartmentEditor.aspx?{Request.QueryString}");
    }
}
```

Kod ažuriranja slike dodajemo koristeći `.AddRange()`, a kod kreiranja jednostavno koristimo cijelu kolekciju slika.

Kod ažuriranja ćemo napraviti redirect na tu istu stranicu da bismo ubuduće vidjeli što se dogodilo sa uploadom slika. Kod kreiranja nove slike želimo vidjeti svih slika da bismo znali da je nova slika dodana.

- provjerimo jesu li podaci o uploadanim slikama u bazi

8.5. Upload slika - prikaz

Slike se sada nalaze na disku, te imamo njihove podatke u bazi. Vrijeme je da ih prikažemo koristeći repeater.

- 'Id' slike će nam trebati kao `HiddenField`
- sliku ćemo prikazati kao `Image` kontrolu, kojoj je `ImageUrl` postavljen na path iz modela
- naziv slike (`Name`) ćemo prikazati kao `TextBox` koji je Bootstrap-stiliziran
- informaciju o tome koja je slika reprezentativna/glavna ćemo prikazati kao Bootstrap-stiliziran `CheckBox`
- informaciju o tome da neku sliku treba brisati kod spremanja ćemo također prikazati kao Bootstrap-stiliziran `CheckBox`
- dodajmo repeater odmah iznad `lblSave LinkButton` kontrole

```
<asp:Repeater ID="repApartmentPictures" runat="server">
    <ItemTemplate>
        <div class="form-group">
            <div class="row">
                <asp:HiddenField runat="server" ID="hidApartmentPictureId" Value='<%# Eval("ID") %>' />
                <div class="col-md-3">
                    <a href="<%# Eval("Path") %>">
                        <asp:Image ID="imgApartmentPicture" runat="server" CssClass="img-thumbnail" ImageUrl='<%# Eval("Path")
                    </a>
                </div>
                <div class="col-md-3">
                    <div class="form-group">
                        <asp:TextBox ID="txtApartmentPicture" runat="server" CssClass="form-control" Text='<%# Eval("Name") %>'
                    </div>
                    <div class="form-group">
                        <label class="btn btn-success">Glavna slika
                            <asp:CheckBox ID="cbIsRepresentative" runat="server" CssClass="is-representative-picture" Checked='
                        </label>
                    </div>
                    <div class="form-group">
                        <label class="btn btn-danger">
                            <span class="glyphicon glyphicon-trash"></span>
                            <asp:CheckBox ID="cbDelete" runat="server" Checked="false" />
                        </label>
                    </div>
                </div>
            </div>
        </div>
    </ItemTemplate>
</asp:Repeater>
```

- sada trebamo napraviti binding na tu kontrolu; napravit ćemo to na isti način kao kod tagova
- prvo ćemo kod dohvata podataka ažurirati model sa slikama iz baze

```
protected void Page_Load(object sender, EventArgs e)
{
    ...
    dbApartment.Tags = _apartmentRepository.GetApartmentTags(id.Value);
    dbApartment.ApartmentPictures = _apartmentRepository.GetApartmentPictures(id.Value);
    ...
}
```

- dodatno, tu trebamo obratiti pažnju na to da u bazi nije potpuna putanja do slike pa ćemo je "popraviti"

```
protected void Page_Load(object sender, EventArgs e)
{
    ...
    dbApartment.Tags = _apartmentRepository.GetApartmentTags(id.Value);
    dbApartment.ApartmentPictures = _apartmentRepository.GetApartmentPictures(id.Value);
    // Fix picture path, database does not contain full path!
    for (int i = 0; i < dbApartment.ApartmentPictures.Count; i++)
    {
        dbApartment.ApartmentPictures[i].Path =
            Path.Combine(_picPath, dbApartment.ApartmentPictures[i].Path);
    }
    ...
}
```

- postavljanje vrijednosti kontrola iz podataka modela se vrši u metodi `SetExistingApartment()`, gdje ćemo pri kraju onda i napraviti databind na repeater za slike (`repApartmentPictures`)

```
private void SetExistingApartment(Models.Apartment apartment)
{
    ...
    repTags.DataSource = apartment.Tags;
    repTags.DataBind();

    repApartmentPictures.DataSource = apartment.ApartmentPictures;
    repApartmentPictures.DataBind();
}
```

- prikaz slika bi sada trebao funkcionirati

8.6. Upload slika - dodatni Javascript

Imamo jedan nezgodan problem - kada se slika odabere kao reprezentativna, slika koja je prije bila odabrana se ne isključi. To jest, više slika je moguće odabrati kao reprezentativne, što nije bila ideja.

- to možemo popraviti dodavanjem jednog malog Javascript programa (odmah iza repeatera za slike) koji će reagirati na odabir `Checkbox` kontrole i maknuti oznake s drugih `Checkbox` kontrola

```
<script>
    $(function () {
        var repPicCheckboxes = $(".is-representative-picture > input[type=checkbox]");

        repPicCheckboxes.change(function () {
            currentCheckbox = this;
            if (currentCheckbox.checked) {
                repPicCheckboxes.each(function () {
                    otherCheckbox = this
                    if (currentCheckbox != otherCheckbox && otherCheckbox.checked) {
                        otherCheckbox.checked = false;
                    }
                })
            }
        });
    })
</script>
```

9. Brisanje apartmana

9.1. Brisanje apartmana - korisničko sučelje

- dodajmo novu formu ApartmentDelete.aspx koja ima Master Page
- u `ApartmentsList.aspx` dodajmo u gridview novi stupac koji nas vodi na tui novu formu za brisanje apartmana

```
<asp:GridView ID="gvListaApartmana" ...>
    <Columns>
        ...
        <asp:TemplateField HeaderText="">
            <ItemTemplate>
                <asp:HyperLink ID="hlDelete" runat="server" CssClass="btn btn-danger" Text="Briši" NavigateUrl='<%# Eval("Id",
            </ItemTemplate>
        </asp:TemplateField>
    </Columns>
</asp:GridView>
```

9.2. Brisanje apartmana - forma za brisanje

Kod brisanja želimo prikazati osnovnu info o apartmanu i postaviti pitanje tipa "želite li obrisati... ?"

- dodajmo instancu repozitorija `ApartmentRepository` u ApartmentDelete.aspx.cs (konstruktor)
- povucimo podatke za koje imamo `id` u query stringu; dodajmo i SetFormApartment() metodu koja će u formu upisati te osnovne podatke

```
public partial class ApartmentDelete : System.Web.UI.Page
{
    private readonly ApartmentRepository _apartmentRepository;

    public ApartmentDelete()
    {
        _apartmentRepository = new ApartmentRepository();
    }

    protected void Page_Load(object sender, EventArgs e)
    {
          string qryStrId = Request.QueryString["id"];
          int? id = null;
          if (!string.IsNullOrEmpty(qryStrId))
          {
              id = int.Parse(qryStrId);
              var dbApartment = _apartmentRepository.GetApartment(id.Value);
              SetFormApartment(dbApartment);
          }
    }

    private void SetFormApartment(Models.Apartment apartment)
    {
    }
}
```

- dodajmo kontrole za osnovne podatke na stranicu ApartmentDelete.aspx, zajeno sa 2 gumba za potvrdu brisanja i odustajanje

```
<div class="form-group">
    <label>Vlasnik</label>
    <asp:Label ID="lblApartmentOwner" runat="server" CssClass="form-control"></asp:Label>
</div>
<div class="form-group">
    <label>Naziv</label>
    <asp:Label ID="lblName" runat="server" CssClass="form-control"></asp:Label>
</div>
<div class="form-group">
    <label>Adresa</label>
    <asp:Label ID="lblAddress" runat="server" CssClass="form-control"></asp:Label>
</div>
<div class="form-group">
    <label>Grad</label>
    <asp:Label ID="lblCity" runat="server" CssClass="form-control"></asp:Label>
</div>
<div>
    <asp:LinkButton ID="lbConfirmDelete" runat="server" Text="Potvrdi brisanje" CssClass="btn btn-danger" />
    <asp:LinkButton ID="lbBack" runat="server" Text="Odustani" CssClass="btn btn-primary" />
</div>
```

- napunimo labele podacima

```
private void SetFormApartment(Models.Apartment apartment)
{
    lblApartmentOwner.Text = apartment.OwnerName;
    lblName.Text = apartment.Name;
    lblAddress.Text = apartment.Address;
    lblCity.Text = apartment.CityName;
}
```

- provjerimo prikaz podataka u formi za brisanje tako da kliknemo "Briši" na stranici s listom apartmana

9.3. Brisanje apartmana - brisanje u bazi podataka

- dodajmo click handlere na potvrdu brisanja i odustani
- za potvrdu brisanja trebamo novu metodu u repozitoriju koju ćemo pozvati, te na kraju redirect

- u repozitoriju...

```
public void DeleteApartment(int id)
{
    var commandParameters = new List<SqlParameter>();
    commandParameters.Add(new SqlParameter("@id", id));

    SqlHelper.ExecuteNonQuery(
        _connectionString,
        CommandType.StoredProcedure,
        "dbo.DeleteApartment",
        commandParameters.ToArray());
}
```

- u ApartmentDelete.aspx...

```
protected void lbConfirmDelete_Click(object sender, EventArgs e)
{
    string qryStrId = Request.QueryString["id"];
    var id = int.Parse(qryStrId);
    _apartmentRepository.DeleteApartment(id);

    Response.Redirect("ApartmentList.aspx");
}
```

```
protected void lbBack_Click(object sender, EventArgs e)
{
    Response.Redirect("ApartmentList.aspx");
}
```

- provjerimo rade li brisanje (tako da pregledamo ima li redak u bazi popunjen stupac `DeletedAt`) i odustajanje

Ostalo

Kako pozvati web-metodu na serveru?

- recimo da je naša stranica koju editiramo ApartmentList.aspx
- dodati `<asp:ScriptManager>` ako ga nema na stranici ApartmentList.aspx (ili u Site.masteru)
- u `<asp:ScriptManager>` dodati atribute `EnablePageMethods="true" EnablePartialRendering="true"`
- napisati u codebehind web-metodu, npr.

```
[WebMethod]
[ScriptMethod]
public static string Test()
{
    return "Test123";
}
```

- napisati Javascript handler koji može pozvati tu web metodu

```
<script>
    function DoTest() {
        //console.log("Debug DoTest()");
        $.ajax({
            type: "POST",
            url: "ApartmentList.aspx/Test",
            contentType: "application/json; charset=utf-8",
            dataType: "json",
            success: function (response) {
                console.log(response.d);
            },
            failure: function (response) {
                console.log(response);
            }
        });
    }
</script>
```

- ako sada želimo pozvati web-metodu npr. na klik nekog linka:

```
<asp:HyperLink runat="server" Text="Click" CssClass="btn btn-primary" NavigateUrl="javascript:void(0);" OnClick="DoTest()" />
```

- ako želimo da web-metoda vraća JSON podatke, dodamo atribut s JSON formatom na web metodu i tada možemo pristupiti vraćenim podacima

```
[WebMethod]
[ScriptMethod(ResponseFormat = ResponseFormat.Json)]
public static List<Models.City> Test()
{
    var repo = new CityRepository();
    return repo.GetCities();
}
```

```
<script>
    function DoTest() {
        $.ajax({
            type: "POST",
            url: "ApartmentList.aspx/Test",
            contentType: "application/json; charset=utf-8",
            dataType: "json",
            success: function (response) {
                var data = response.d;
                for (item in data) {
                    console.log(data[item].Name);
                }
            },
            failure: function (response) {
                console.log(response);
            }
        });
    }
</script>
```