

Višeslojna aplikacija - složeniji primjeri

Refleksija i korisnički atributi

Izvješća

2014/15.09

Poslovni objekt sa stavkama

❑ **Primjer:**  **FirmaWin \ Firma.BLL \ BusinessEntities \ Dokument.cs**

```
public partial class Dokument : BusinessBase {
    public Dokument() {
        // Inicijalizacija praznih stavaka
        stavke = new BusinessBaseList<Stavka>();
        stavke.ListChanged +=
            new ListChangedEventHandler(Stavke_ListChanged);
        datDokumenta = DateTime.Now;
    }
    private BusinessBaseList<Stavka> stavke;
    public BusinessBaseList<Stavka> Stavke
    {
        get { return stavke; }
    }
    private void Stavke_ListChanged(...) {
        // Kad se promijeni stavka osvježiti iznos dokumenta...
        OnPropertyChanged("IznosDokumenta");
        if (e.ListChangedType == ListChangedType.ItemAdded)
            stavke[e.NewIndex].BllProvider = this.BllProvider;
    }
}
```

Promjena stavke

- ❑ **Primjer:**  **FirmaWin \ Firma.BLL\ BusinessEntities \ Stavka.cs**
- ❑ **Promjena artikla u stavci treba ažurirati zavisne vrijednosti (npr. jed. mjere)**
 - Obavljeno u poslovnom objektu, a ne u prezentacijskom sloju

```
public int? SifArtikla{
    get { return sifArtikla; }
    set{
        ...
        sifArtikla = value;
        PropertyChanged("SifArtikla");
        if (value.HasValue){
            var bllProvider = new ArtiklBllProvider();
            Artikl a = bllProvider.Fetch(sifArtikla.Value);
            JedMjereArtikla = a.JedMjere;
            JedCijArtikla = a.CijArtikla;
            ...
        }
    }
}
```

- Promjene ostalih podataka preko svojstava (podižu događaj *PropertyChanged*)
- Promjena stavke uzrokuje događaj *ListChanged* (obrada ažurira iznos dokumenta)

Validacija složenog objekta

- ❑ **Primjer:**  **FirmaWin \ Firma.BLL \ BusinessEntities \ Dokument.cs**
 - Validacija svojstava dokumenta + validacija svake stavke

```
public override void Validate() {  
    DoValidation("VrDokumenta");  
    DoValidation("BrDokumenta");  
    DoValidation("DatDokumenta");  
    DoValidation("IdPartnera");  
    DoValidation("IdPrethDokumenta");  
    DoValidation("PostoPorez");  
    foreach (var stavka in Stavke)  
    {  
        if (Stavke[i].BllProvider == null) {  
            Stavke[i].BllProvider = this.BllProvider;  
        }  
        stavka.Validate();  
    }  
    ...  
}
```

Konstruktor forme dokumenta

❏ Primjer: FirmaWin \ Firma \ Forms \ DokumentForm


- inicijalizacija poslovnog sloja za referencirane podatke


```
public DokumentForm()  
{  
    InitializeComponent();  
  
    // Dohvat svih partnera, artikala i dokumenata za lookup  
    // Dohvat svih dokumenata  
    partnerInfoBindingSource.DataSource =  
        partnerBll.FetchLookup();  
    artiklBindingSource.DataSource = artiklBll.FetchLookup();  
    prethDokumentBindingSource.DataSource =  
        dokumentBll.FetchLookup();  
  
    dokumentBindingSource.DataSource = dokumentBll.FetchLazy();  
}
```

Povezivanje stavaka

❑ Primjer: FirmaWin \ Firma \ Forme \ DokumentForm

■ Stavke povezane u dizajnu

Properties	
dokumentBindingSource System.Windows.Forms.BindingSource	
	
+	(ApplicationSettings)
(Name)	dokumentBindingSource
AllowNew	True
DataMember	
DataSource	Firma.BusinessEntities.Dokument
Filter	
GenerateMember	True
Modifiers	Private
Sort	

Properties	
stavkaBindingSource System.Windows.Forms.Binding	
	
▶	(ApplicationSettings)
(Name)	stavkaBindingSource
AllowNew	True
DataMember	Stavke
DataSource	dokumentBindingSource
Filter	
GenerateMember	True
Modifiers	Private
Sort	

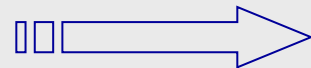
■ Pri promjeni označenog dokumenta, pridružuje mu se konkretni bll objekt

```
private void dokumentBindingSource_CurrentChanged(  
    object sender, EventArgs e)  
{  
    Dokument d = dokumentBindingSource.Current as Dokument;  
    if (d != null && d.BllObject == null)  
        d.BllObject = dokumentBll;  
}
```

Padajuće liste referentnih podataka

❑ **Primjer:** **FirmaWin \ Firma \ Forme \ DokumentForm**

- Tip izvora Firma.BusinessEntities.LookupData
 - Svojstva Key i Text
- Partner
 - idPartneraComboBox
 - DataSource = partnerInfoBindingSource
 - DisplayMember = Text
 - ValueMember = Key
 - SelectedValue = dokumentBindingSource - IdPartnera
- Prethodni dokument
 - idPrethDokumentaComboBox
 - DataSource = prethDokumentBindingSource
 - DisplayMember = Text
 - ValueMember = Key
 - SelectedValue = dokumentBindingSource - IdPrethDokumenta



Dohvat vrijednost za padajuću listu

❑ Primjer: FirmaWin \ Firma.BLL \ PartnerBllProvider.cs

- Postupak FetchLookup iz DAL sloja vraća Dictionary<int, string>
 - Parovi (vrijednost primarnog ključa, tekst koji treba prikazati)
 - Pretvara se u listu elemenata tipa LookupData

```
public List<LookupData> FetchLookup() {  
    Dictionary<int, string> lookupData = dal.FetchLookup();  
    var list = lookupData.Select(d => new LookupData {  
        Key = d.Key, Text = d.Value }).ToList();  
    return list;  
}
```

❑ Primjer: FirmaWin \ Firma.EF \ PartnerDalProvider.cs

- Poziva se pohranjena procedura (naziv partnera složenijeg oblika)
- Rezultat se pretvara u Dictionary<int, string>

```
public Dictionary<int, string> FetchLookup() {  
    using (FirmaEntities ctx = new FirmaEntities()) {  
        return ctx.ap_PartnerLookup().  
            ToDictionary(p => p.IdPartnera, p => p.Naziv);  
    }  
}
```


Primjer pohranjene procedure za padajuću listu

```
CREATE PROCEDURE [dbo].[ap_PartnerLookup] AS
BEGIN
    SELECT * FROM fn_PartnerLookup() ORDER BY Naziv, IdPartnera
END
```

```
CREATE FUNCTION [dbo].[fn_PartnerLookup]() RETURNS TABLE AS
RETURN (
    SELECT IdTvrtke AS IdPartnera, NazivTvrtke
           + ' (' + OIB + '/' + MatBrTvrtke + ')' AS Naziv
    FROM Tvrtka
    INNER JOIN Partner ON Partner.IdPartnera = Tvrtka.IdTvrtke
    UNION
    SELECT IdOsobe AS IdPartnera, PrezimeOsobe + ', '
           + ImeOsobe + ' (' + OIB + ')' AS Naziv
    FROM Osoba
    INNER JOIN Partner ON Partner.IdPartnera = Osoba.IdOsobe
)
```

Izračunata vrijednost iznosa dokumenta

❑ **Primjer:**  **FirmaWin \ Firma \ Forme \ DokumentForm**

```
iznosDokumentaTextBox.DataBindings.Add(new Binding("Text",  
    dokumentBindingSource, "IznosDokumenta", true,  
    DataSourceUpdateMode.OnPropertyChanged, string.Empty, "C2"));
```

❑ **Primjer:**  **FirmaWin \ Firma.BLL \ BusinessEntities \ Dokument.cs**

```
public decimal? IznosDokumenta {  
    get {  
        CheckLazyLoad();  
        decimal rez = 0;  
        foreach (Stavka s in this.stavke) {  
            if (s.Iznos.HasValue)  
                rez += s.Iznos.Value;  
        }  
  
        if (postoPorez.HasValue) {  
            rez *= (1m + postoPorez.Value);  
        }  
        return rez;  
    }  
}
```

Ulančavanje referentnog podatka sa zaglavlja

❑ Primjer: FirmaWin – DokumentForm : PartnerForm

```
protected override void Zoom()
{
    if (InEditMode)
    {
        if (idPartneraComboBox.Focused)
        {
            PartnerForm f = new PartnerForm();
            f.StartPosition = FormStartPosition.CenterScreen;
            if (f.ShowDialog() == DialogResult.OK)
            {
                // Potrebno zbog bindinga na lookup novounesenih
                partnerInfoBindingSource.DataSource =
                    partnerBll.FetchAll();

                Partner p = (Partner)f.Selected;
                (dokumentBindingSource.Current
                    as Dokument).IdPartnera = p.IdPartnera;
            }
        }
    }
}
```

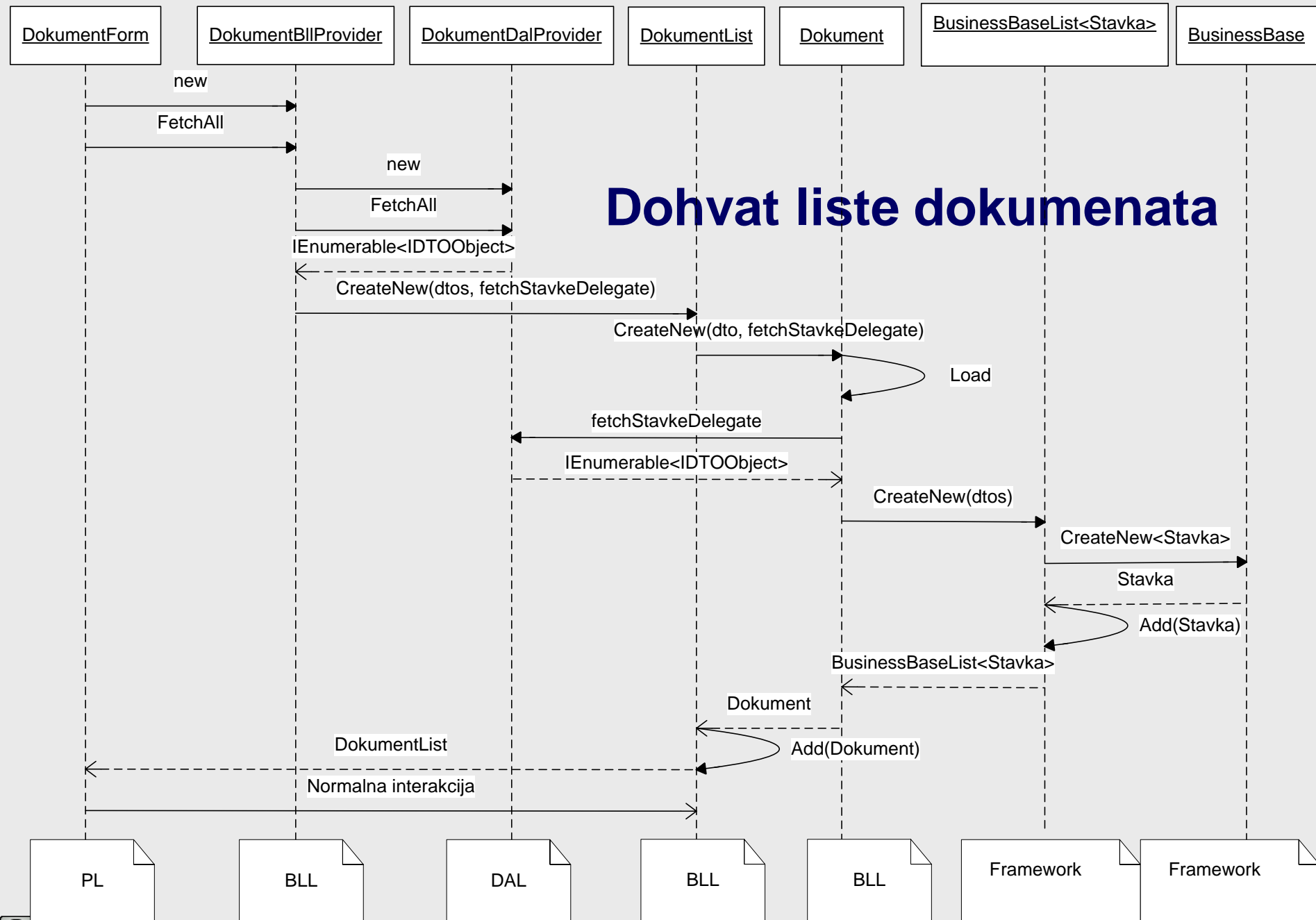
Ulančavanje referentnog podatka sa stavke

❑ Primjer: FirmaWin – DokumentForm : ArtiklForm

```
if (InEditMode &&
    stavkaDataGridView.SelectedCells[0].ColumnIndex == 0 && ...) {
ArtiklForm f = new ArtiklForm();
    if (f.ShowDialog() == DialogResult.OK) {
        artiklBindingSource.DataSource = artiklB11.FetchAll();
        ...
        if (stavkaDataGridView.Rows...IsNewRow) { // SKRAĆENO
            stavkaBindingSource.CancelEdit();
            Dokument dok = (Dokument)dokumentBindingSource.Current;
            s = new Stavka();
            dok.Stavke.Add(s);
        } else
            s = (Stavka)stavkaBindingSource.Current;

        s.SifArtikla = a.SifArtikla;
        s.NazArtikla = a.NazArtikla;
        ...
    }
```

Dohvat liste dokumenata



Dohvat složenog objekta

❏ Primjer: FirmaWin \ Firma.BLL \ DokumentBllProvider

- kreatoru liste predaje se i delegat na postupak koji koristeći DAL provider učitava stavke složenog objekta

```
private DokumentDalProvider dal = new DokumentDalProvider();

...

public DokumentList FetchAll() {
    var dalRecord = dal.FetchAll();
    return DokumentList.CreateNew(dalRecord, FetchStavke);
}

public BusinessBaseList<Stavka> FetchStavke(int idDokumenta) {
    var dalRecord = dal.FetchStavke (idDokumenta);
    return BusinessBaseList<Stavka>.CreateNew(dalRecord);
}
```

Dohvat stavki aktualnog zaglavlja (1)

❏ **Primjer:**  **FirmaWin \ Firma.BLL \ BusinessEntities \ DokumentList.cs**

```
internal static Dokument CreateNew(
    IEnumerable<IDTOObject> items,
    Func<int, BusinessBaseList<Stavka>> fetchStavkeDelegate) {
    DokumentList rez = new DokumentList();
    foreach (var dto in items) {
        rez.Add(Dokument.CreateNew(dto, fetchStavkeDelegate));
    }
    return rez;
}
```

❏ **Primjer:**  **FirmaWin \ Firma.BLL \ BusinessEntities \ Dokument.cs**

```
internal static Dokument CreateNew(IDTOObject dtoObject,
    Func<int, BusinessBaseList<Stavka>> stavkeDelegate)
{
    var item = new Dokument();
    item.Load(dtoObject);
    ...
    item.LoadStavke(item.IdDokumenta.Value, stavkeDelegate);
    ...
}
```

Dohvat stavki aktualnog zaglavlja (2)

❏ Primjer:  FirmaWin \ Firma.BLL \ BusinessEntities \ Dokument.cs

```
private void LoadStavke(int idDokumenta,
    Func<int, BusinessBaseList<Stavka>> stavkeDelegate) {
    stavke = stavkeDelegate(idDokumenta);
    foreach (var stavka in stavke)
    {
        stavka.BllProvider = this.BllProvider;
    }
    stavke.ListChanged += new
        ListChangedEventHandler(Stavke_ListChanged);
    SetState(BusinessObjectState.Unmodified);
    //SetState jer smo u međuvremenu mijenjali stavke
}
```


Dohvat stavki iz baze podataka

❑ Primjer: FirmaWin \ Firma.EF \ DokumentDalProvider.cs

- Za učitavanje stavki koristi se pohranjena procedura koja se veže na tablicu Artikl

```
public List<ap_StavkaList_R_Result> FetchStavke(int idDokumenta)
{
    using (FirmaEntities ctx = new FirmaEntities()) {
        var list = ctx.ap_StavkaList_R(idDokumenta).ToList();
        return list;
    }
}
```

```
CREATE PROCEDURE [dbo].[ap_StavkaList_R]
    @IdDokumenta int
AS BEGIN
    SELECT S.*, A.NazArtikla, A.JedMjere AS JedMjereArtikla
    FROM Stavka S
    INNER JOIN Artikl A ON S.SifArtikla = A.SifArtikla
    WHERE IdDokumenta = @IdDokumenta;
END
```

Spremanje složenog objekta (1)

❏ Primjer: FirmaWin \ Firma.BLL \ DokumentBllProvider.cs

- Poslovni objekt se kopira u DTO Dokument (EF.Dokument sa svojim stavkama)
- Specifičnost: vrijednost primarnog ključa je identity – dobit će se tek nakon pohrane
- DAL sloj (Entity Framework) ažurira vrijednosti identity PK-a nakon snimanja
- Dobivene vrijednosti ID-ova pridružiti novim dokumentima i stavkama
- Uspostavljena veza poslovnog i podatkovnog objekta (*dictUpdated*)

```
public void SaveChanges(IList changedItems) {  
    ...  
    //veza poslovnog i dal objekta za naknadno postavljanje id-a  
    var dictUpdated = new Dictionary<Dokument, DAL.Dokument>();  
    ...  
    var dtoObject = (DAL.Dokument)item.ToDtoObject();  
    dictUpdated[item] = dtoObject;  
    newItem.Add(dtoObject);  
    ...  
    dal.SaveChanges(editedItems, newItem, deletedItems);  
    foreach (var pair in dictUpdated) {  
        if (!pair.Key.IdDokumenta.HasValue)  
            pair.Key.IdDokumenta = pair.Value.IdDokumenta;  
    }  
}
```

Spremanje složenog objekta (2)

❏ Primjer: FirmaWin \ Firma.EF \ DokumentDalProvider.cs

- Nove dokumente dodati u kontekst (automatski se dodaju i njegove stavke)
- Označene za brisanje zakačiti za kontekst i označiti za brisanje
 - Isto učiniti i za njegove stavke

```
public void SaveChanges(IEnumerable<Dokument> changedItems,
IEnumerable<Dokument> newItem, IEnumerable<Dokument> deletedItems) {
    ...
    foreach (var item in newItem) {
        ctx.Dokument.Add(item);
    }

    foreach (var item in deletedItems) {
        var ctxDokument = ctx.Dokument.Attach(item);
        foreach (var stavka in ctxDokument.Stavke.ToList()) {
            ctx.Stavka.Attach(stavka);
            ctx.Entry(stavka).State = System.Data.EntityState.Deleted;
        }
        ctx.Entry(item).State = System.Data.EntityState.Deleted;
    }
    ctx.SaveChanges();
}
```

Spremanje složenog objekta (3)

❏ Primjer: FirmaWin \ Firma.EF \ DokumentDalProvider.cs

- Ažurirani dokumenti mogu imati nove, obrisane ili promijenjene stavke
- **SetValues** postavlja nove vrijednosti entitetu na osnovu vrijednosti nekog drugog ent.

```
public void SaveChanges(...) {  
    ...  
    foreach (var dokument in changedItems) {  
        var dbDokument = ctx.Dokument.Find(dokument.IdDokumenta);  
        ctx.Entry(dbDokument).CurrentValues.SetValues(dokument);  
        //uzmi postojeće stavke iz baze  
        foreach (var dbStavka in dbDokument.Stavke.ToList()) {  
            var stavka = dokument.Stavke.FirstOrDefault  
                (s => s.IdStavke == dbStavka.IdStavke);  
            //da li stavka još uvijek postoji ili ju treba obrisati  
            if (stavka != null)  
                ctx.Entry(dbStavka).CurrentValues.SetValues(stavka);  
            else  
                ctx.Entry(dbStavka).State = System.Data.EntityState.Deleted;  
        }  
        foreach (var stavka in dokument.Stavke.  
            Where(s => s.IdStavke == default(int)).ToList())  
            //nove stavke nemaju id  
            dbDokument.Stavke.Add(stavka);  
    }  
}
```

Povezivanje na tip specijalizacije

❑ Primjer: FirmaWin – PartnerForm

- automatski prikaz panela prikladnog tipu partnera
 - povezivanjem na kontrole osobaCheck, odnosno tvrtkaCheck
- kontrole mijenjaju vrijednost
 - navigacijom na zapis (partnerBindingSource_CurrentChanged)
 - interaktivno, tijekom uređivanja

```
osobaPanel.DataBindings.Add(  
    new Binding("Visible", osobaCheck, "Checked"));  
  
tvrtkaPanel.DataBindings.Add(  
    new Binding("Visible", tvrtkaCheck, "Checked"));
```

❑ Primjer: FirmaWin – TipPartnera

- enum TipPartnera
- class TipPartneraConverter

```
public enum TipPartnera  
{  
    Osoba,  
    Tvrtka,  
    Nedefinirano  
}
```

Prikaz tipa specijalizacije pri navigaciji

❏ Primjer: FirmaWin \ Firma \ Forme \ PartnerForm

```
private void partnerBindingSource_CurrentChanged(...)
{
    Partner p = partnerBindingSource.Current as Partner;

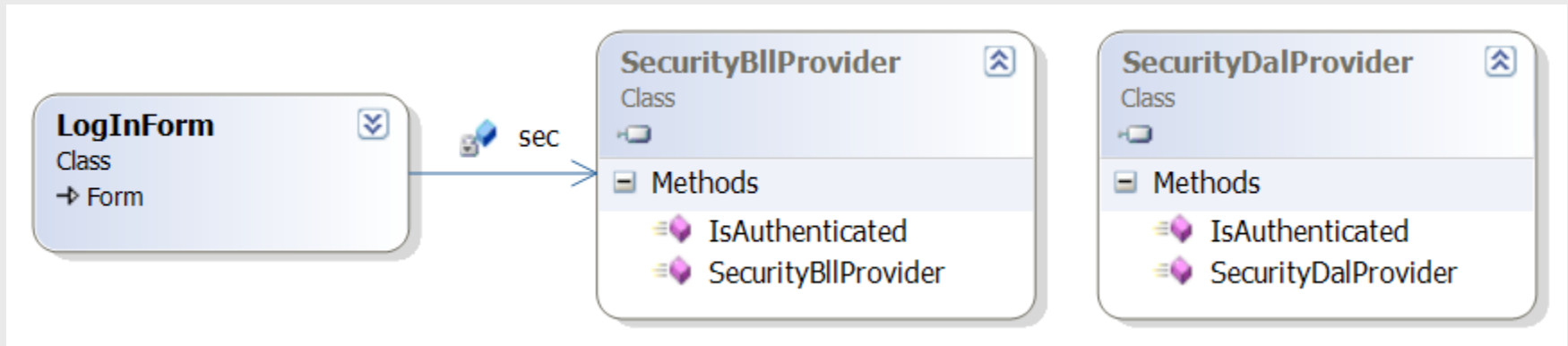
    if (p == null) // TipPartnera.Nedefinirano
    {
        osobaCheck.Checked = false;
        tvrtkaCheck.Checked = false;
    }
    else if (p.TipPartnera == TipPartnera.Tvrtka)
    {
        osobaCheck.Checked = false;
        tvrtkaCheck.Checked = true;
    }
    else // TipPartnera.Osoba
    {
        osobaCheck.Checked = true;
        tvrtkaCheck.Checked = false;
    }
}
```

Promjena tipa specijalizacije pri uređivanju

❑ Primjer:  FirmaWin \ Firma \ Forme \ PartnerForm

```
private void tvrtkaCheck_CheckedChanged(...)
{
    Partner p = partnerBindingSource.Current as Partner;
    if (p != null)
    {
        if (tvrtkaCheck.Checked)
        {
            p.TipPartnera = TipPartnera.Tvrtka;
        }
        else
        {
            p.TipPartnera = TipPartnera.Osoba;
        }
    }
}
```

Provjera korisnika



```
public class SecurityDalProvider
{
    public bool IsAuthenticated(string username, string password)
    {
        using (FirmaEntities ctx = new FirmaEntities()) {
            int count = ctx.Korisnik.
                Where(k => k.Username == username).
                Where(k => k.Password == password).
                Count();
            return count == 1;
        }
    }
}
```


Refleksija i korisnički atributi

Refleksija

- ❑ **Proces kojim program može pregledavati i modificirati vlastitu strukturu tijekom izvođenja**
- ❑ **Refleksija se upotrebljava za:**
 - Dohvat metapodataka (sadržanih u atributima)
 - Otkrivanje tipa podatka
 - Pristup informacijama o učitanim asemblijima i tipovima definiranim unutar njih (pregled interakcija i instanciranje tipova)
 - Dinamičko povezivanje na svojstva i postupke
 - Pozivanje svojstava ili postupaka dinamički instanciranog objekta na temelju otkrivenog tipa (*dynamic invocation*), npr. *bind* fonta ili boje
 - Stvaranje i korištenje novih tipova za vrijeme izvršavanja programa
- ❑ **Primjeri upotrebe**
 - Razvoj aplikacija za reverzno inženjerstvo
 - Razvoj preglednika razreda
 - Razvoj editora svojstava razreda (kao *Properties* prozor)
- ❑ **Prostor imena *System.Reflection* i razred *System.Type***

Prostor imena *System.Reflection*

❑ Prostor imena *System.Reflection*

- Sadrži razrede i sučelja za dohvat informacija o tipovima i članovima programskog koda pri izvođenju, kao i dinamičko kreiranje tipova

❑ Razredi

- `Assembly` – predstavlja asemblij
 - Postupci `Load`, `LoadFile`, `GetName`, `GetModules`, ...
- `MemberInfo` (apstraktni razred)
 - Omogućuje pristup metapodacima o članovima razreda
- `ConstructorInfo`, `PropertyInfo`, `MethodInfo`
 - Pristup metapodacima konstruktora, svojstava, odnosno postupaka
 - Nasljeđuju razred `MemberInfo`
 - `ConstructorInfo` i `MethodInfo` imaju još i postupke `IsPrivate`, `IsPublic`, `IsStatic`, ...
- ...

Razred System.Type

❑ Važnija svojstva:

- `IsAbstract`, `IsArray`, `IsClass`, `IsCOMObject`, `IsEnum`, `IsGenericTypeDefinition`, `IsGenericParameter`, `IsInterface`, `IsPrimitive`, `IsNestedPrivate`, `IsNestedPublic`, `IsSealed`, `IsValueType`

❑ Važniji postupci:

- `GetConstructors()`, `GetEvents()`, `GetFields()`, `GetInterfaces()`, `GetMembers()`, `GetMethods()`, `GetNestedTypes()`, `GetProperties()`
 - Svaki od ovih postupaka vraća polje koje predstavlja tražene informacije (npr. `GetMethods` vraća polja s članovima tipa `MethodInfo`)
- Postupci pandani prethodnima, ali s nazivima u jednini za dohvat konkretnog svojstva, postupka, ... npr.
 - `GetConstructor(type[])`, `GetProperty(string)`, `GetProperty(string, BindingFlags)`, `GetMethod(string)`, `GetMethod(string, BindingFlags)` ...

❑ `InvokeMember()` – Naknadno povezivanje za određeni element

❑ `GetType()` – statički postupak za vraćanje instance razreda `Type`

Stvaranje objekta razreda *System.Type* (1)

□ Ako je razred prisutan (referenciran) u trenutku kompilacije:

- Pozivom postupka `GetType` na nekom postojećem objektu:

```
MojRazred r = new MojRazred();
```

```
...
```

```
Type t = r.GetType();
```

- Korištenjem operatora `typeof`

```
Type t = typeof(MojRazred);
```

- Korištenjem statičkog postupka `GetType` iz razreda `Type` uz navođenje punog imena naziva razreda

```
Type t = Type.GetType("MojProjekt.MojRazred");
```

Stvaranje objekta razreda *System.Type* (2)


❑ Ako razred nije prisutan (referenciran) u trenutku kompilacije:

- Korištenjem statičkog postupka `GetType` iz razreda `Type` uz navođenje punog imena naziva razreda pripadajućeg asemblija (odvojen zarezom i razmakom)
 - `Type t = Type.GetType("MojProjekt.MojRazred, MojProjekt");`
 - Ako je razred generički iza naziva imena dodaje se znak ``` i broj tipova s kojima je razred parametriziran
`Type.GetType("System.Collection.Generic.Dictionary`2");`
 - Postupak `Type.GetType` je preopterećen te po želji omogućuje ignoriranje velikih i malih slova

❑ Asemblij se prethodno treba učitati unutar programa

- `Assembly.Load("naziv asemblija")`, ako je odgovarajući dll ili exe u mapi `bin\Debug` (ili sl.)
- Ili navođenjem pune putanje do asemblija
- `Assembly.LoadFrom("putanja do asemblija")`

Naknadno povezivanje (eng. late binding)

- ❑ Tehnika kojom se može stvoriti objekt nekog tipa i pozvati njegov postupak bez poznavanja tipa u trenutku kompilacije
- ❑ Postupak *CreateInstance* unutar razreda *Activator* stvara novi objekt određenog tipa
- ❑ Primjer:  Refleksija\Refleksija\Program.cs

```
Assembly asm = Assembly.LoadFrom
    (@"..\\..\\..\\ClassLibrary1\\bin\\Debug\\ClassLibrary1.dll");
Type type = asm.GetType("ClassLibrary1.Loto");

object obj = Activator.CreateInstance(type, 7, 39);
MethodInfo info = type.GetMethod("IzvuciBrojeve");
object result = info.Invoke(obj, new object[] {false});
string ispis = string.Join(", ", (List<int>)result);
Console.WriteLine(ispis);
```

Atributi

❑ Atributi su nadodane oznake tipovima, poljima, postupcima i svojstvima

- Definirani uglatim zagradama [] prije deklaracije entiteta koji opisuju
- Dostupni programski tijekom izvođenja programa

❑ Primjeri preddefiniranih atributa

- `Browsable` – vidljivost svojstva ili događaja u prozoru *Properties*
- `Category` – kategorija u kojoj se svojstvo ili događaj prikazuje u *Properties*
 - Neke preddefinirane kategorije: *Data*, *Behavior*, *Design*, *Action*, *Misc*
 - Moguće je definirati i vlastite kategorije
- `Description` - opis svojstva ili događaja
- `Obsolete` – označava da je neki postupak zastario (upozorenja pri kompilaciji)

❑ Razred ***System.Attribute*** – osnovni razred za attribute

- `GetCustomAttribute` – vraća atribut danog tipa *Type* primijenjenog na asemblij, član razreda, ...
- `GetCustomAttributes` – vraća polje atributa
- `IsDefined` – određuje da li je ijedan atribut zadanog tipa *Type* definiran
 - `npr. IsDefined(MemberInfo, Type)`

Atributi


- ❑ Vlastiti atributi nasljeđuju *System.Attribute*
- ❑ Naziv uobičajeno završava s *Attribute*, ali nije nužno
 - Ako završava s *Attribute*, taj se sufiks prilikom korištenja može se izostaviti

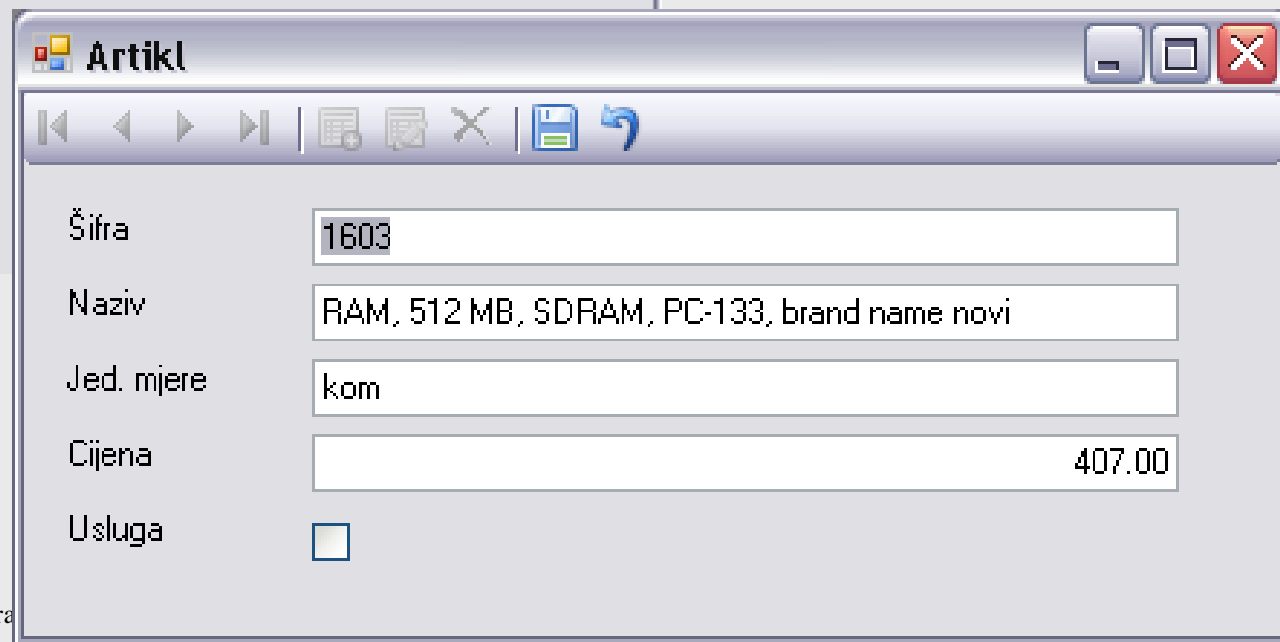
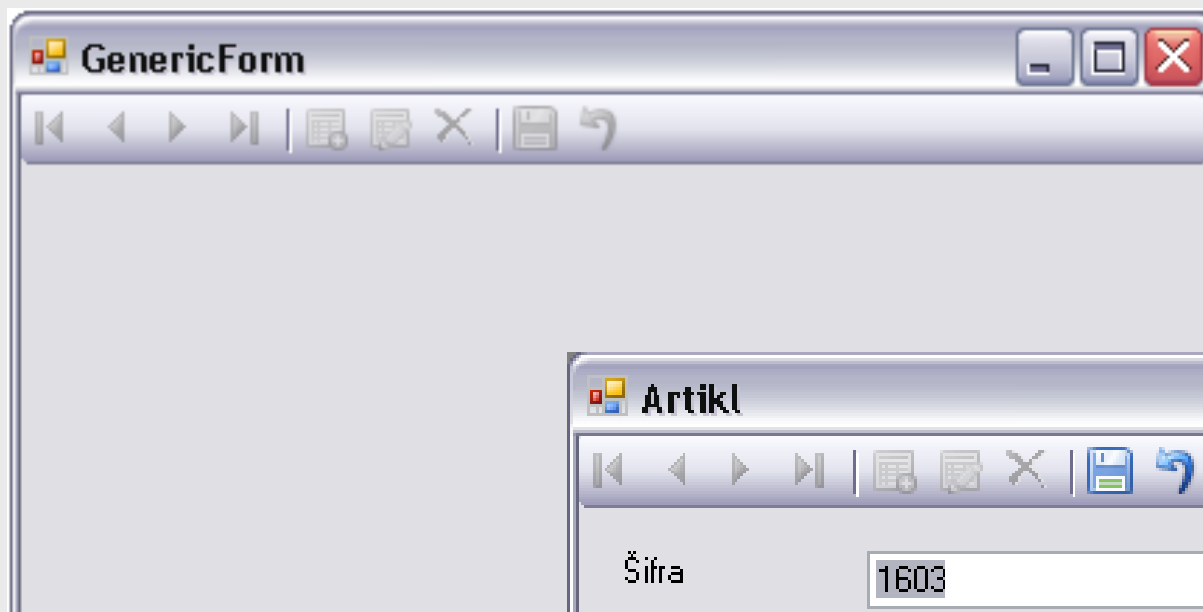
```
public class PrviAttribute : System.Attribute{  
    public string Naziv { get; set; }  
    public int Broj { get; set; }  
}  
  
public class DrugiAttribute : System.Attribute{  
    public string X { get; set; }  
}  
  
[Prvi(Broj=5, Naziv="Test")]  
[DrugiAttribute(X = "Proba")]  
public class MojRazred{...}
```

Univerzalni i samoprilagodljivi programski moduli

Primjena refleksije i korisničkih atributa

Univerzalna forma

- ❑ Preddefinirana forma koja se pri pokretanju prilagodi podacima koje treba obraditi.
- ❑ Primjer:  FirmaWin \ Firma \ Core \ GenericForm.cs

A screenshot of a software window titled "Artkl". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. Below the title bar is a toolbar with several icons: a double left arrow, a single left arrow, a single right arrow, a double right arrow, a document with a magnifying glass, a document with a pencil, a document with an 'X', a floppy disk, and a circular arrow. The main area of the window contains a form with the following fields:

Šifra	1603
Naziv	RAM, 512 MB, SDRAM, PC-133, brand name novi
Jed. mjere	kom
Cijena	407.00
Usluga	<input type="checkbox"/>

Pokretanje univerzalne forme

❑ Umjesto instanciranja namjenski oblikovane forme

- npr. `ArtiklForm f = new ArtiklForm();`
- koja pri pokretanju prikupi i poveže potrebne podatke

❑ Kreira se općenita forma

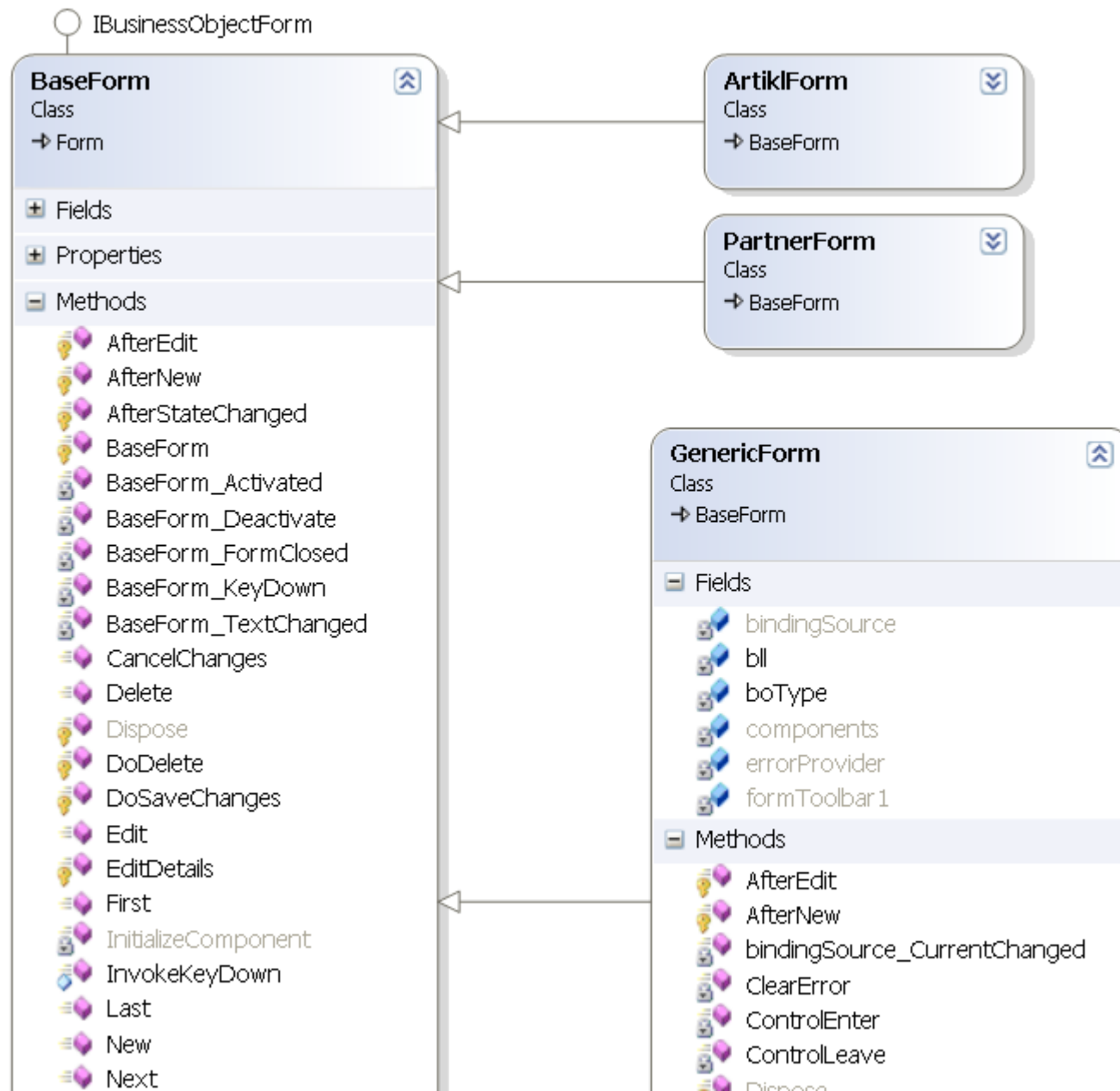
- koja se prilagodi predanom skupu podataka i odgovarajućem tipu

❑ Primjer: FirmaWin – MainForm - Šifrnici

```
private void artiklToolStripTablice_Click(...)
{
    using (new StatusBusy())
    {
        ArtiklBllProvider bll = new ArtiklBllProvider();
        GenericForm f = new GenericForm("Artikl",
            bll.FetchAll(), bll, typeof(Artikl));
        f.MdiParent = this;
        f.Show();
    }
}
```

Dizajn univerzalne forme

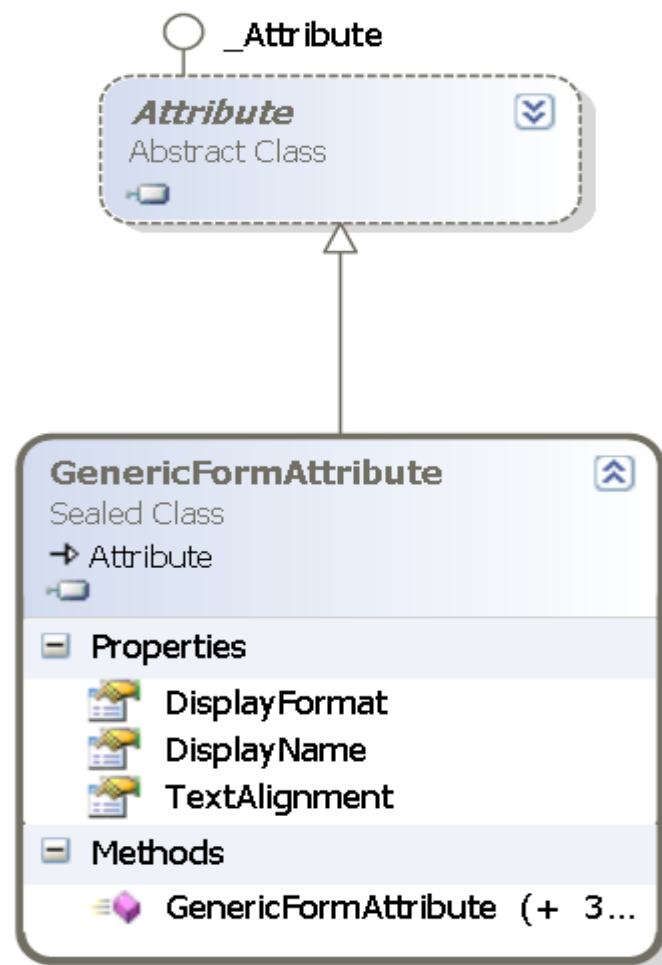
- ❑ Funkcionalnost izvedena iz osnovne forme, koju nasljeđuju i druge forme.



Prilagodba ponašanja

- ❑ Prilagodba korisnički definiranim atributima poslovnog entiteta
- ❑ Primjer:  FirmaWin \ Firma.BLL \ BusinessEntities \ Artikl.cs

```
[GenericForm("Šifra")]
public int? SifArtikla {
    get { return sifArtikla; }
    set { ... }
}
...
GenericForm("Cijena", "N2",
    HorizontalAlignment.Right)]
public decimal? CijArtikla {
    get { return cijArtikla; }
    set { ... }
}
```



Definiranje vlastitih atributa

❑ Primjer: FirmaWin \ Firma.Framework \ GenericFormAttribute.cs

- atribut za označavanje svojstava koja će se pokazati na univerzalnoj formi

```
[AttributeUsage(AttributeTargets.Property)]
public sealed class GenericFormAttribute : Attribute {
    private string displayName = string.Empty;
    private string displayFormat = string.Empty;
    private HorizontalAlignment ha = HorizontalAlignment.Left;
    ...
    public GenericFormAttribute(string displayName) {
```

❑ Primjer: FirmaWin \ Firma.BLL \ BusinessEntities \ Artikl.cs

```
[GenericForm("Cijena", "N2", HorizontalAlignment.Right)]
public decimal? CijArtikla {
    get { return cijArtikla; }
    set { ... }
}
```

Obrada atributa refleksijom

❑ Primjer: FirmaWin \ Firma \ Core \ GenericForm (SetupForm)

- Za pojedino svojstvo dodaje se prikladna kontrola na formu
- Dohvat svih javnih svojstava danog poslovnog objekta
- Provjera da li je svojstvo označeno atributom za prikaz na univerzalnoj formi

```
private Type boType;  
PropertyInfo[] props = boType.GetProperties(  
    BindingFlags.Instance | BindingFlags.Public);  
  
foreach (PropertyInfo prop in props) {  
    if (Attribute.IsDefined(prop, typeof(GenericFormAttribute))) {  
        GenericFormAttribute atr =  
            (GenericFormAttribute)Attribute.GetCustomAttribute(prop,  
                typeof(GenericFormAttribute));  
  
        ... //kreiranje kontrole za svojstvo  
    }  
    ...  
}
```


Kreiranje kontrole prema tipu svojstva

- ❑ Za tip svojstva *bool*, *DateTime* ili *string* kreira se kontrola *CheckBox*, *DateTimePicker*, odnosno *TextBox*

```
Control c = null;
if (prop.PropertyType.Equals(typeof(bool)) ||
    prop.PropertyType.Equals(typeof(bool?))) {
    c = new CheckBox();
    c.DataBindings.Add(new Binding(
        "CheckState", bindingSource, prop.Name, true));
}
else if (prop.PropertyType.Equals(typeof(DateTime)) ||
        prop.PropertyType.Equals(typeof(DateTime?))) {
    c = new DateTimePicker();
    c.DataBindings.Add(new Binding("Value", bindingSource, prop.Name));
}
else {
    c = new TextBox();
    ...
}
```

Dinamičko kreiranje kontrola na formi

```
// Kreiranje labele za svojstvo
Label l = new Label();
l.Text = atr.DisplayName;

// Postavljanje kontrole i labele na formu
this.Controls.Add(l);
l.Location = labelPos;
labelPos.Y += 26;

c.Width = 300;
this.Controls.Add(c);
c.Location = controlPos;
controlPos.Y += 26;

l.SendToBack();
c.Enter += new EventHandler(ControlEnter);
c.Leave += new EventHandler(ControlLeave);
```

Rukovanje svojstvom temeljem atributa

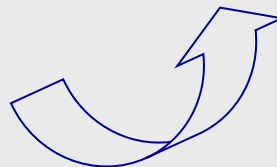
Artikl.cs

```
[GenericForm("Šifra")]
public int? SifArtikla
{
    get { return sifArtikla; }
    set
    {
        if (InEditMode)
        {
            sifArtikla = value;
            PropertyChanged(
                "SifArtikla");
        }
    }
}
```



BusinessBase.cs

```
protected void PropertyChanged(
    string propertyName)
{
    isDirty = true;
    // Osvježavanje data-bindingom
    OnPropertyChanged(propertyName);

    // Validacija
    DoValidation(propertyName);
}
```



Još jedan primjer primjene refleksije

- ❑ **Primjer:**  **FirmaWin \ Firma \ Core \ Utils.cs,**
 - Postupak `SetNull` – postavlja *null* na kontrole povezane na izvor podataka
- ❑ **Primjer:**  **FirmaWin \ Firma \ Core \ GenericForm(ControlLeave)**

```
public static class Utils {  
    public static void SetNull(Control c,  
        string bindedProperty, object businessObject)    {  
        Binding b = c.DataBindings[bindedProperty];  
        PropertyInfo p = businessObject.GetType().GetProperty(  
            b.BindingMemberInfo.BindingField,  
            BindingFlags.Instance | BindingFlags.Public);  
  
        if (p.PropertyType.Equals(typeof(string))) {  
            p.SetValue(businessObject, string.Empty, null);  
        }  
        else {  
            p.SetValue(businessObject, null, null);  
        }  
    }  
}
```

Oblikovanje izvješća

Izvješće

❑ Izvješće (*report*)

- Dokument za prezentaciju i organizaciju informacija o nekoj temi
- Izvješća mogu sadržavati tekst, brojeve, grafove, ilustracije

❑ Generator izvješća

- Aplikacija koja omogućava definiranje izgleda izvješća.
- Korisnik može:
 - odabrati koje podatke prikazati na izvješću (pojedinačne zapise, raspone zapisa, svojstva)
 - prilagoditi razmještaj podataka (zaglavlja, brojeve stranica, font, ...)

❑ *Reports*

- Paket za izradu izvješća, ugrađen u *VisualStudio*
- Samo izvješće je programska komponenta
- Ista struktura izvješća dinamički se može puniti različitim podacima
- Izvješće se može zapisati u .pdf, .xls, .doc, .rtf, .txt, .html i ispisati na pisaču

Primjer izvješća o dokumentima

□ Primjer: Izvjesca \ Dokumenti

- Popis dokumenta za odabrano vremensko razdoblje i odabranog poslovnog partnera, grupirano po datumu.

Popis dokumenata po datumu za razdoblje 10.1.2007 do 23.1.2007

10.1.2007



Broj	Datum	Partner	Iznos
267	10-sij-2007	Marija Jurinčić (1104982385036)	214.715.00 kn
269	10-sij-2007	Dino Stipanović (2602953335295)	234.098.00 kn

12.1.2007

Broj	Datum	Partner	Iznos
277	12-sij-2007	Miroslav Kolega (0108935330144)	266.061.00 kn
283	12-sij-2007	Hrvoje Ležaić (0503951335055)	1.212.067.00 kn
288	12-sij-2007	Marko Ležaić (0505952335183)	861.307.00 kn
290	12-sij-2007	Frane Lončar (0710951330113)	1.612.885.00 kn

Izvor podataka

❑ Izvor podataka za izvješće može biti:

- Baza podataka (ADO.NET DataSet, EntityFramework)
- Vlastiti razredi
 - Razredi iz poslovnog sloja (npr. iz primjera *Firma.Win*)
 - U projekt potrebno dodati reference na dll-ove nastale iz Firma.BLL
 - U slučaju da poslovni objekt nema sve traže podatke ili nisu potrebni svi podaci može se stvoriti novi razred
 - Primjer:  Izvjesca \ DokumentiStavke \ StavkaZalispis.cs
 Izvjesca \ Dokumenti \ Zalispis \ Dokument.cs
 - (Napomena: *Prilikom odabira izvora vlastiti razred možda neće biti vidljiv ako u međuvremenu nije napravljen Build*)

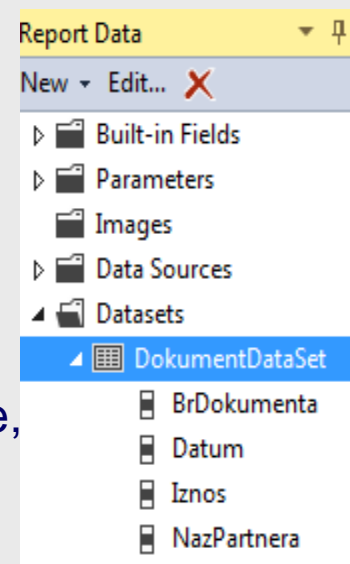
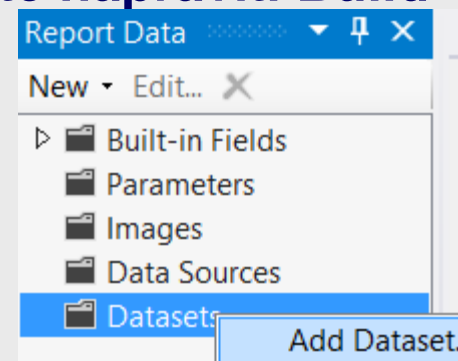
❑ Izvješću se mogu predati

- pojedinačni podaci, a grupiranje se obavlja u dizajnu
- već grupirani podaci
- kombinacija prethodna dva slučaja

❑ Filtriranje je moguće postaviti u dizajnu izvješća ili prilikom dohvata suziti količinu podataka

Stvaranje izvješća

- ❑ Napisati nove razrede i/ili dodati reference na prethodno napisane vlastite razrede (npr. *Firma.BLL* i *Firma.Framework*) te napraviti **Build**
- ❑ Dodati .rdlc datoteku
 - Project \ Add \ New Item ... Report (ili ReportWizard)
- ❑ Postaviti izvor podataka
 - Aktivirati prozor s podacima izvješća View \ Report Data
 - Dodati skup podataka za izvještaj
 - Datasets \ desni klik \ Add Dataset (*nema veze s ADO.NET datasetovima*)
 - Odabrati *Object* za vrstu izvora
 - Odabrati željeni razred
 - u našem primjeru Zalspis \ Dokument.cs
 - Preimenovati Dataset po želji (npr. u DokumentDataSet)
- ❑ S **Toolboxa** dovući kontrole po želji (opisano kasnije)
 - Pridružiti svojstva pojedinim elementima
 - *Built in Fields* – posebna polja npr. Page Number, Report Name,
 - *Parameter Fields* – polja za parametrizaciju izvješća
 - Grupiranja, agregatne vrijednosti, sortiranja...



Važnije kontrole za izradu izvješća

❑ Table

- prikaz podataka s fiksnim brojem stupaca i neograničenim brojem redaka
- za realizaciju se koristi tip *Tablix*

❑ Matrix

- varijabilni broj redaka i stupaca
- realizirano tipom *Tablix*
















❑ List

- spremnik za više elemenata koji se ponavljaju za svaki redak u izvoru podataka
- realizirano tipom *Rectangle*

❑ TextBox

- Prikaz slobodnog teksta, povezane vrijednosti ili proizvoljne formule

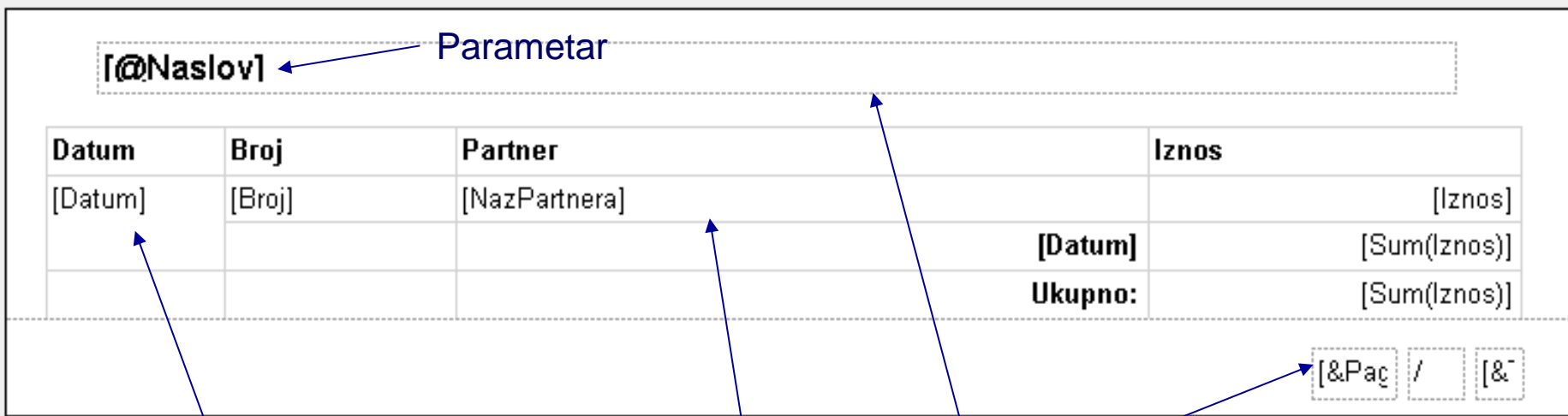
▲ Report Items

	Pointer
	Text Box
	Line
	Table
	Matrix
	Rectangle
	List
	Image
	Subreport
	Chart
	Gauge
	Map
	Data Bar
	Sparkline
	Indicator

Struktura izvješća

❑ Primjer: Izvjesca \ Dokumenti \ Dokumenti.rdlc

- Zaglavlje (izvještaja, stranice) i podnožje (izvještaja, stranice)
 - Desni klik na izvješće → Insert Page Header/Footer
- Stupci i retci
- Kontrole za prikaz pojedinačnih elemenata/tekstova (TextBox)
- Kontrole za prikaz lista (list/rectangle, table/tablix, matrix/tablix)
- Dodatni elementi – za grupiranje, izvješća unutar izvješća
- Parametri



The diagram illustrates the structure of an RDL report. It features a header section with a parameter placeholder `[@Naslov]`, a data table, and a footer section with a page number placeholder. Blue arrows point from labels to specific components: 'Parametar' points to the header parameter, 'Grupiranje' points to the first column of the table, 'Tablica' points to the table itself, 'TextBox' points to the footer page number, and an unlabeled arrow points to the table's data rows.

Datum	Broj	Partner	Iznos
[Datum]	[Broj]	[NazPartnera]	[Iznos]
			[Sum(Iznos)]
			[Sum(Iznos)]

Footer: [&Pač] / [&]

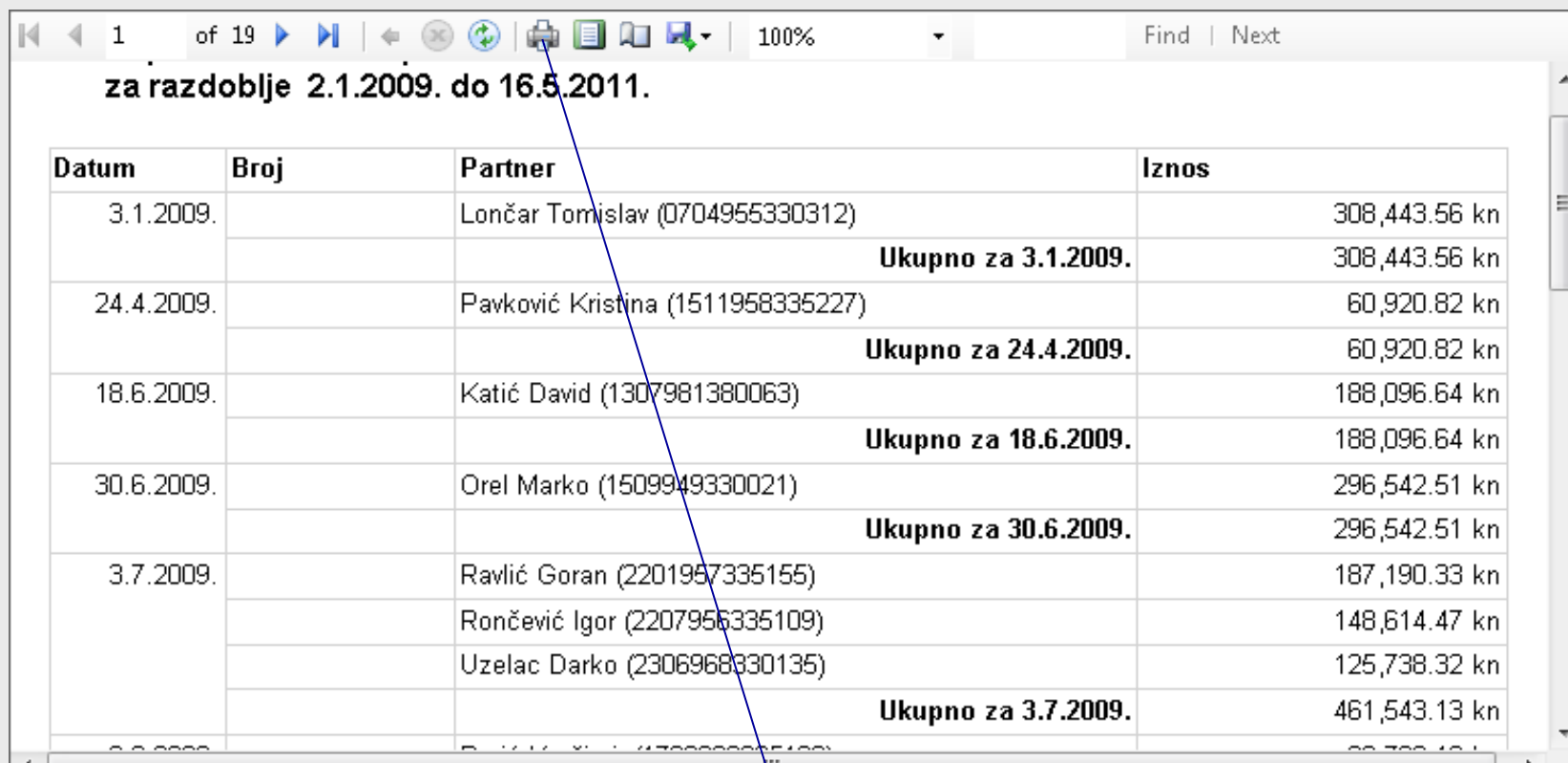
Grupiranje

Tablica

TextBox

Dodavanje izvješća na formi

- ❑ **ReportViewer – kontrola za prikaz izvješća**
 - dodaje se iz Toolboxa kao druge WinForms kontrole
- ❑ **Izvještaj se pridružuje u dizajnu ili programski**
 - Prikazano na sljedeća dva slajda
- ❑ **Primjer izgleda kontrole prilikom izvršavanja i dohvata podataka**



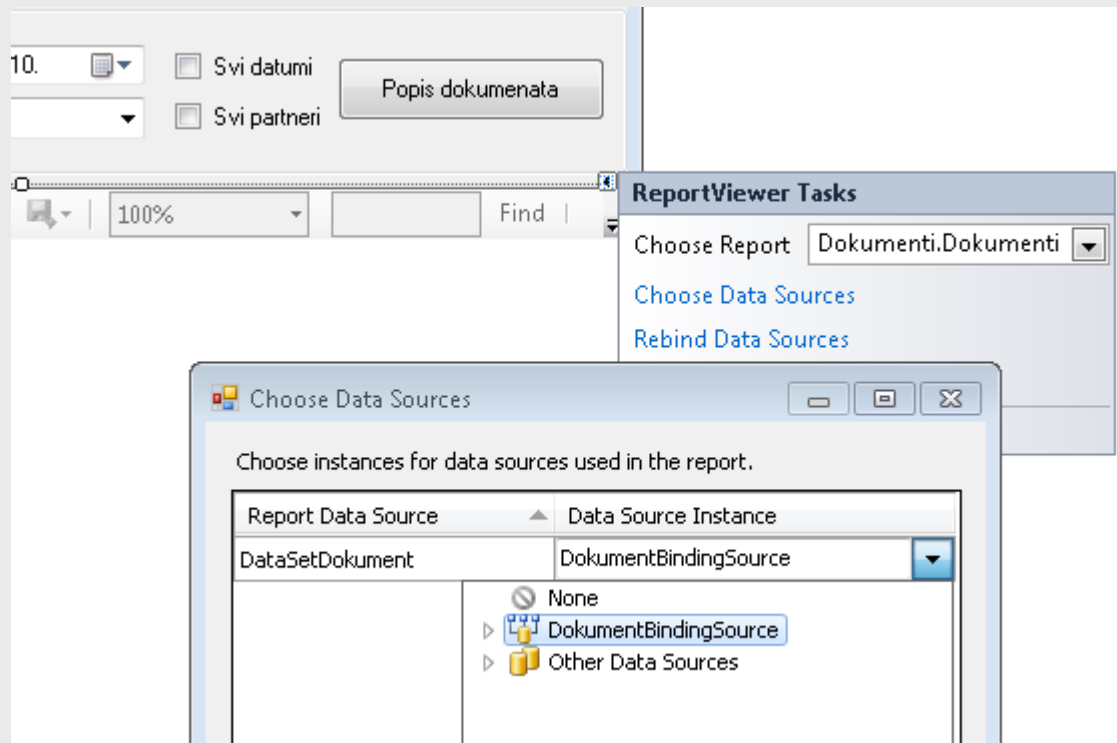
1 of 19 100% Find | Next

za razdoblje 2.1.2009. do 16.5.2011.

Datum	Broj	Partner	Iznos
3.1.2009.		Lončar Tomislav (0704955330312)	308,443.56 kn
		Ukupno za 3.1.2009.	308,443.56 kn
24.4.2009.		Pavković Kristina (1511958335227)	60,920.82 kn
		Ukupno za 24.4.2009.	60,920.82 kn
18.6.2009.		Katić David (1307981380063)	188,096.64 kn
		Ukupno za 18.6.2009.	188,096.64 kn
30.6.2009.		Orel Marko (1509949330021)	296,542.51 kn
		Ukupno za 30.6.2009.	296,542.51 kn
3.7.2009.		Ravlić Goran (2201957335155)	187,190.33 kn
		Rončević Igor (2207956335109)	148,614.47 kn
		Uzelac Darko (2306968330135)	125,738.32 kn
		Ukupno za 3.7.2009.	461,543.13 kn

Povezivanje izvješća u dizajnu

- ❑ Konkretna rdlc datoteka se postavlja kao izvještaj za kontrolu **ReportViewer**
- ❑ Kao izvor podataka izvješću postavlja se generirani ***BindingSource***



- ❑ Nakon punjenja podataka u generirani ***BindingSource*** pozvati ***RefreshReport*** za kontrolu **ReportViewer**

Povezivanje izvješća programski

❑ Primjer: Izvjesca\Dokumenti\DokumentForm

1. Kontrolu reportViewer povezati s konkretnim izvještajem
 - a) korištenjem svojstva *LocalReport.ReportEmbeddedResource* ili
 - b) korištenjem svojstva *LocalReport.ReportPath*
2. Pripremiti parametre
3. Dodati izvor podataka u kolekciju *DataSources* (koristi se ime koje je korišteno u dizajnu)
4. Osvježiti kontrolu *reportViewer* pozivom postupka *RefreshReport*

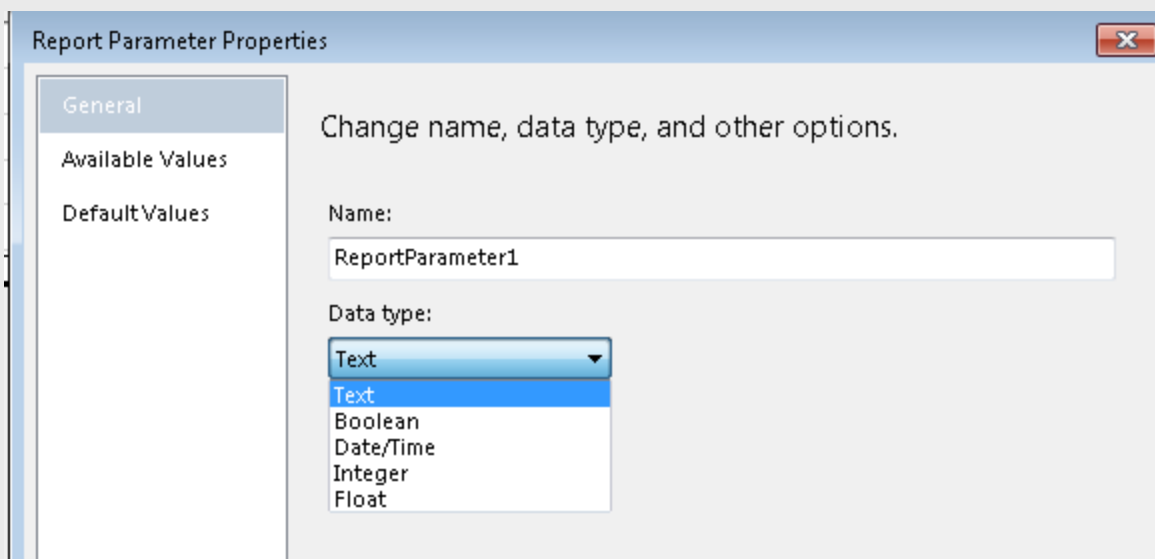
```
reportViewer.LocalReport.ReportEmbeddedResource =  
                                "Dokumenti.Dokumenti.rdlc";  
ReportParameter naslov = new ReportParameter("Naslov");  
naslov.Values.Add("tekst naslova")  
reportViewer.LocalReport.SetParameters(naslov);  
ReportDataSource reportSource =  
                                new ReportDataSource("DokumentDataSet", listaZaIspis);  
reportViewer.LocalReport.DataSources.Clear();  
reportViewer.LocalReport.DataSources.Add(reportSource);  
reportViewer.RefreshReport();
```

Parametrizacija

- ❑ U prozoru *Report Data (View → Report Data)* desni klik na *Parameters → Add Parameter*
 - Odabrati tip vrijednost i (opcionalno) moguće vrijednosti
- ❑ U kodu instancirati objekt tipa *ReportParameter* i postaviti mu naziv i vrijednost

```
ReportParameter naslov = new ReportParameter("Naslov");  
naslov.Values.Add("tekst naslova")  
reportViewer.LocalReport.SetParameters(naslov);
```

- Za naziv upotrijebiti naziv korišten u dizajnu
- Vrijednost parametra je tipa *StringCollection*

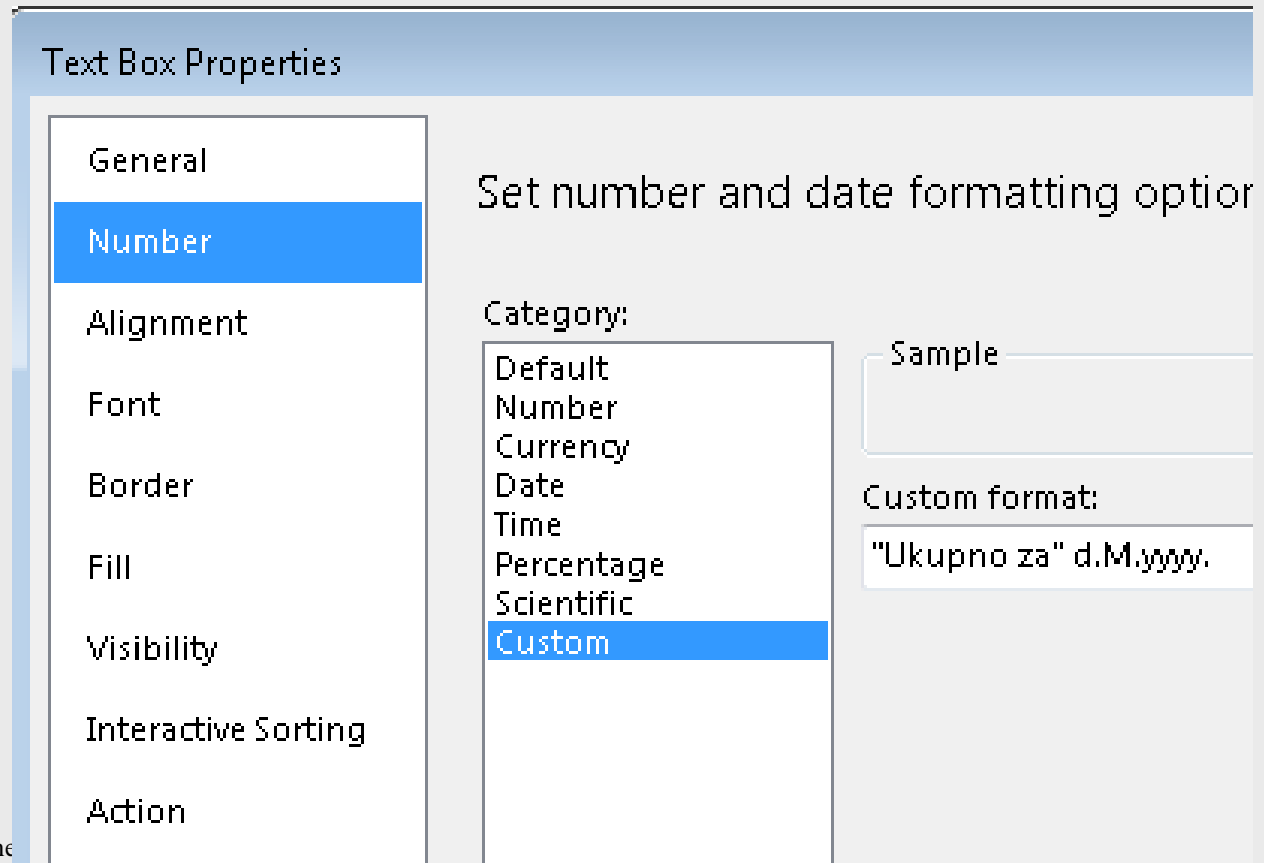


Formatiranje podataka

❑ Desni klik na element za prikaz podatka (npr TextBox)

→ **TextBoxProperties**

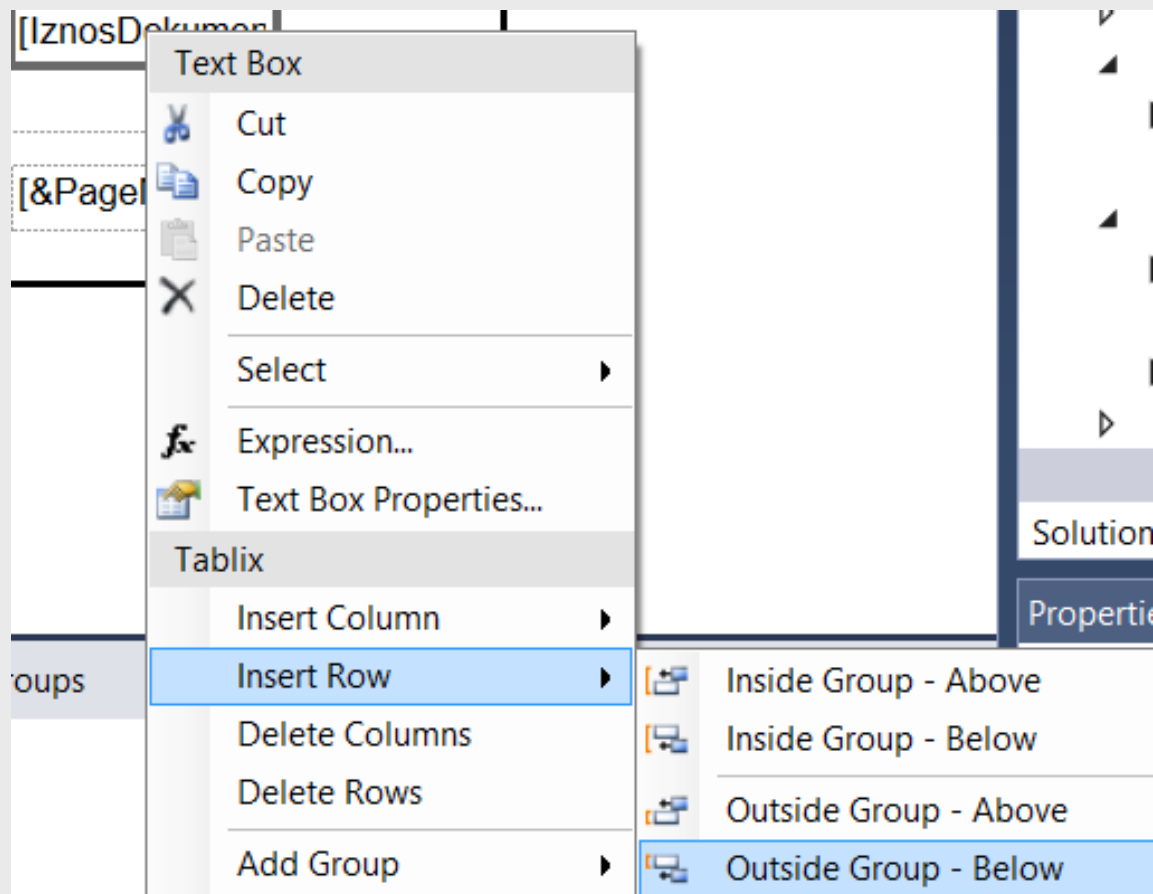
- Format datuma, brojeva, valute, ...
- Standardni formati za oblikovanje ispisa
- Konkretni tekst se upisuje unutar navodnika



Dodavanje retka u podnožje tablice ili grupe

❑ Desni klik na ćeliju u tablici → Insert Row → Outside Group - Below

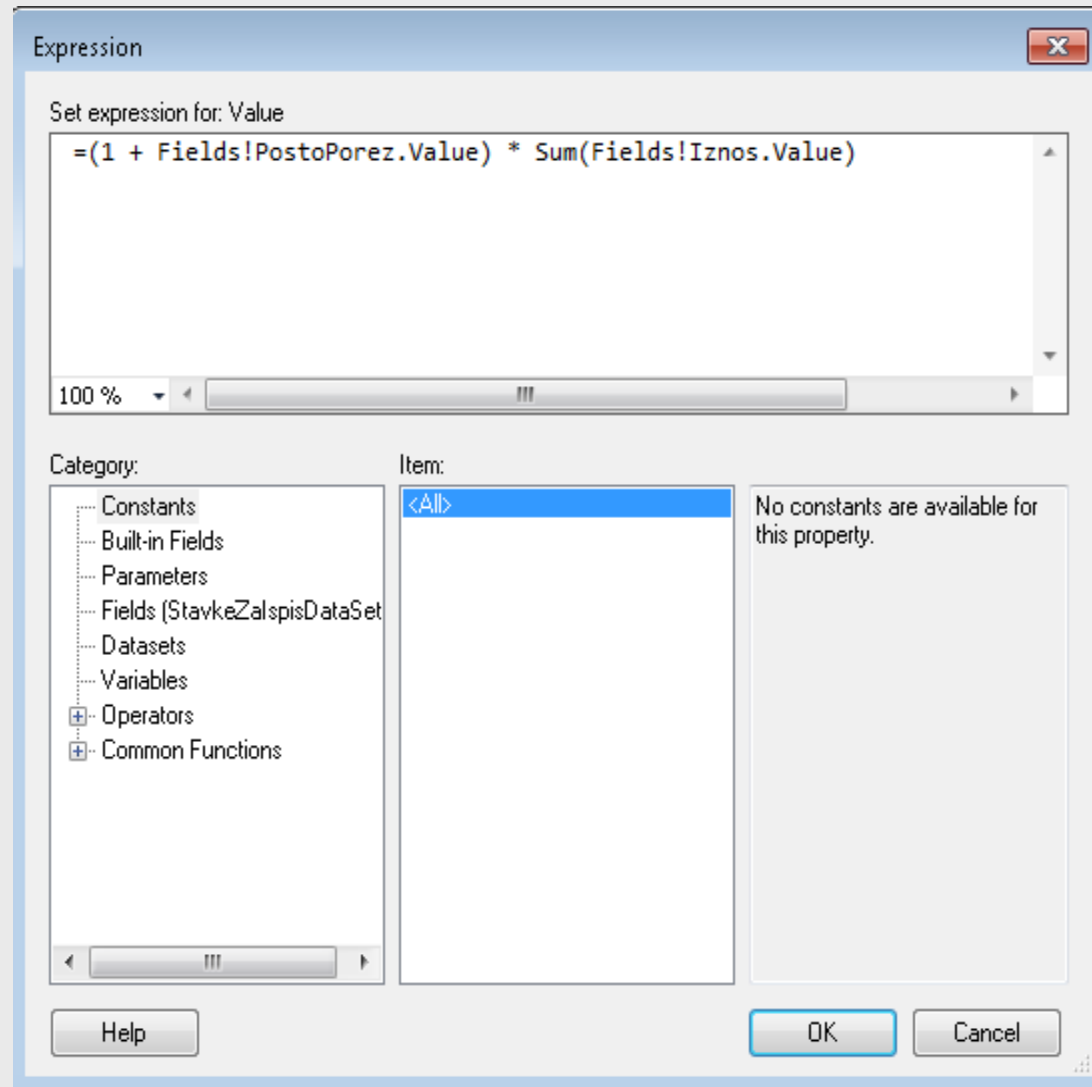
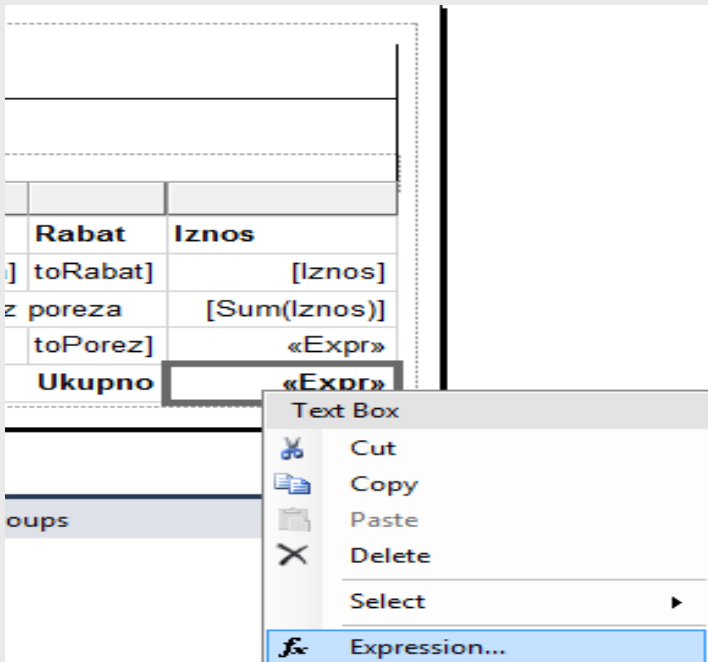
- Dodaje van trenutne grupe (na dno tablice ako nema grupiranja)
- Može sadržavati tekst ili formulu, npr. `=Sum(Fields!NazivStupca.Value)`
- Može i desni klik → Add Total




Formule

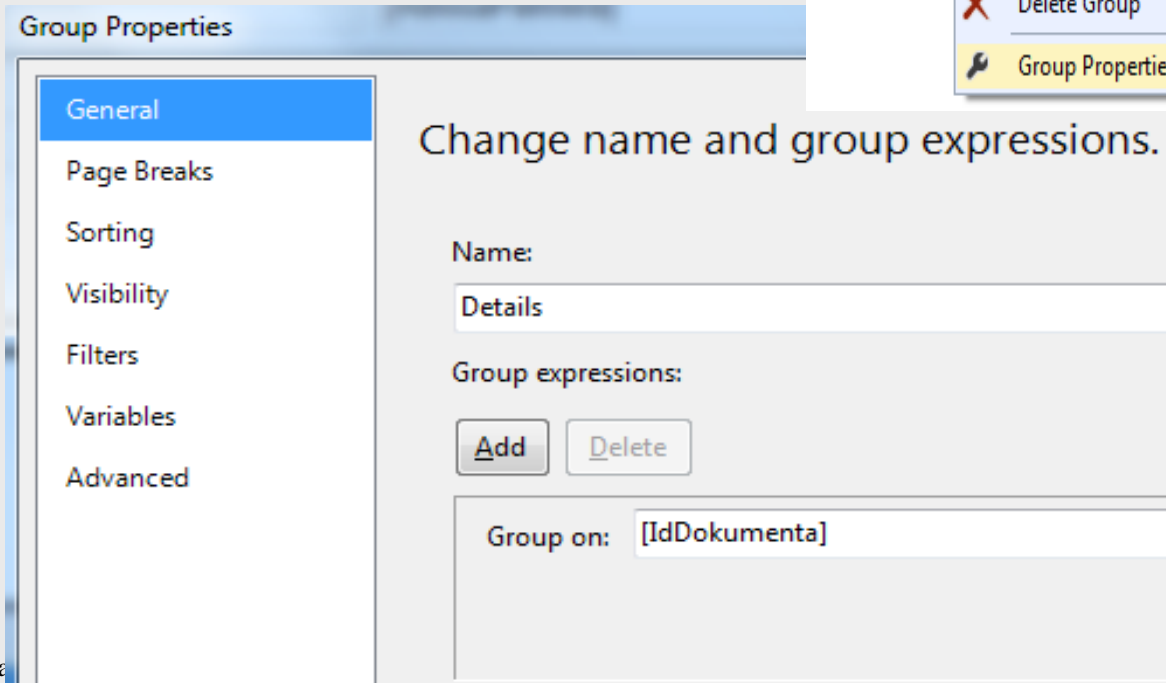
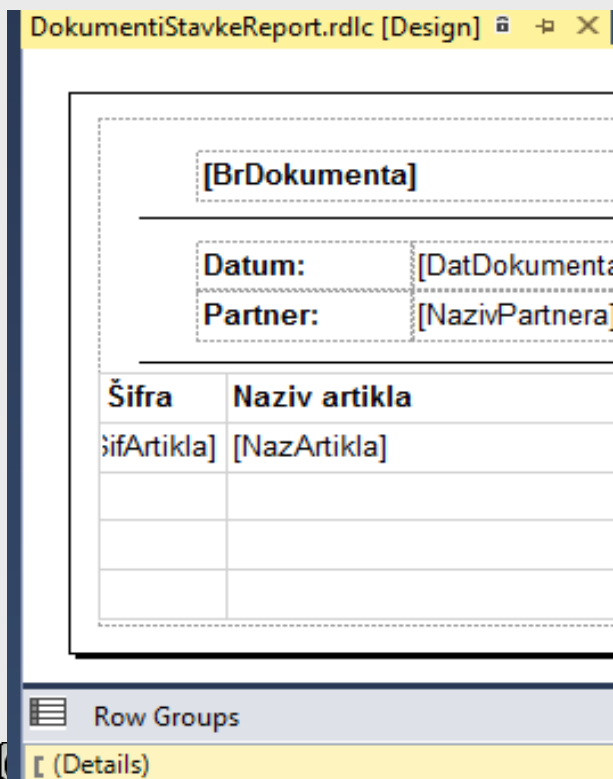
❑ Desni klik na element za prikaz podatka → Expression

- Standardne agregatne funkcije za oblikovanje ispisa
- Vrijednost pojedinog polja oblika
`Fields!Naziv.Value`



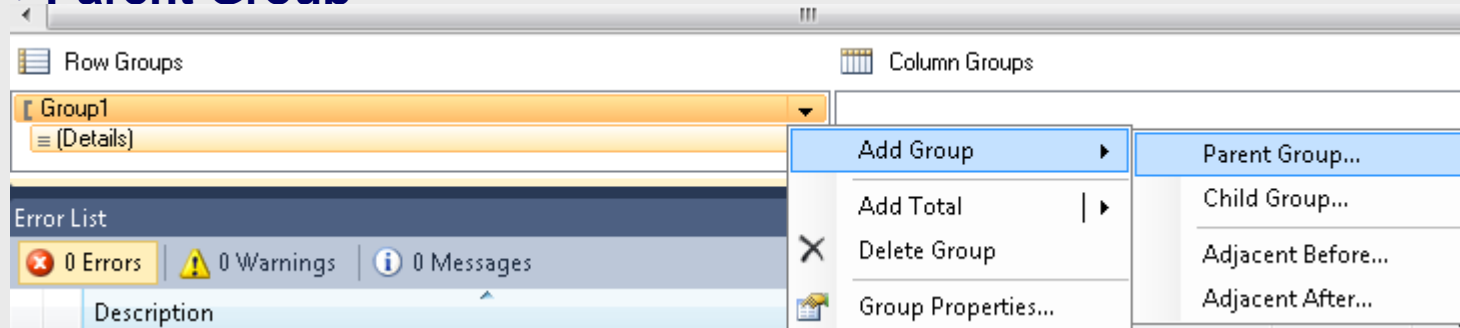
Grupiranje podataka (Bez izdvojenog stupca/retka)

- ❑ Na kontroli za koju se vrši grupiranje (npr. *List* ili *Table*)
Row Groups (ili Column Groups) \ Details \ Group Properties
 - odabrati izraze po kojima će se vršiti grupiranje
 - ovaj način ne stvara zasebni stupac (ili redak) za grupirane podatke
 - Primjer:  Izvjesca\DokumentiStavke\DokumentiStavkeReport.rdlc
 - Prikazuje se samo prvi podatak iz grupe
 - Koristi se kad su stupci/retci sa sumarnim podacima

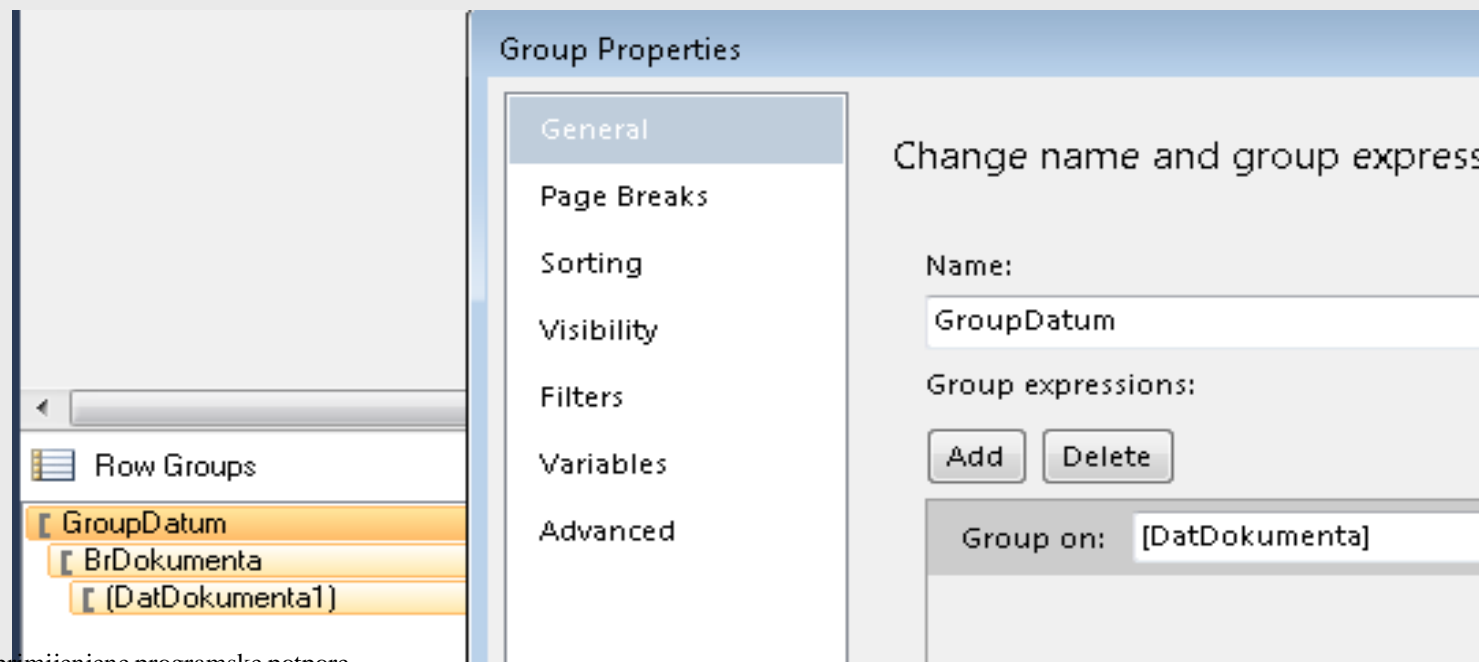


Grupiranje podataka s posebnim stupcem

- ❑ U okviru *RowGroups* (ili *ColumnGroups*) – desni klik na neki element – Add Group → Parent Group



- ❑ Odabir polja ili izraza za grupiranje (npr. *DatDokumenta*)



Primjer izvješća sa zaglavljem i stavkama

□ Primjer: Izvjesca \ DokumentiStavke

Dokument br: 30						
Datum: 12/29/2009 12:00:00 AM						
Partner: Krofak Mario (0305979335208) Kraljevački brijegi 20 10000 Zagreb						
Šifra	Naziv artikla	Količina	Jedinica mjere	Jedinična cijena	Rabat	Iznos
11943	Glazbena linija, GROUNDIG Varixx UMS 4401, 2 x 8 W RMS, micro, krom	8.0000	kom	785.00 kn	4.00%	6,028.80 kn
17692	Torba za notebook, BATACO, A/CCT 2V	1.0000	kom	209.00 kn	3.00%	202.73 kn
25264	MBO KINTEL, s. 775, D975XBX2KR, i975X, BUS 1333 MHz, serial ATA II, RAID, 7.1 zvuk, 1Gbps, SSSR 2, 2x FireWire, ATX 2, bulk	4.0000	kom	1,429.00 kn	5.00%	5,430.20 kn
24857	Knjiga "CCNA" + CD	1.0000	kom	259.00 kn	8.00%	238.28 kn
24894	Torba za notebook, KINGSINGTON Contour Terrain Messenger (63100EU)	0.0000	kom	249.00 kn	5.00%	0.00 kn
24837	Knjiga "Premiere Pro za 24 časa"	2.0000	kom	245.00 kn	8.00%	450.80 kn
				Ukupno bez poreza		181,015.24 kn
				Porez	22.00%	39,823.35 kn
				Ukupno		220,838.59 kn

Izrada izvješća sa zaglavljem i stavkama

1. **Napraviti razred koji sadrži sve pojedinačne podatke koji se trebaju prikazati**
 - Može biti razred iz poslovnog sloja ili novi razred koji sadrži sve potrebne podatke (denormalizirano)
2. **Dodati novi skup podataka (*DataSet iz Report Data*)**
 - Pogledati slajd *Stvaranje izvješća*
3. **Na izvješće dodati kontrolu *List* u koju će se dodati osnovni podaci iz zaglavlja**
4. **Postaviti grupiranje po primarnom ključu zaglavlja**
(Row Groups → Details → Group Properties → Add...)
5. **Unutar postojeće kontrole tipa *List* dodati kontrolu tipa *Table* u koju se dodaju podaci koji pripadaju pojedinoj stavki**
6. **Za sumiranje po pojedinom stupcu unutar stavki odabrati ćeliju s podatkom, a zatim desni klik → *Insert Row* → *Outside Group* → *Below* i odabrati željenu formulu**

Primjer izvješća sa zaglavljem i stavkama (2)

❑ Primjer: Izvjesca \ DokumentiStavke

•kontrola tipa
List/Rectangle

[BrDokumenta]						
Datum:		[DatDokumenta]				
Partner:		[NazivPartnera]		[AdresaPartnera]		
Šifra	Naziv artikla	Količi	Jedini	Jedinična	Rabat	Iznos
[ŠifArtikla]	[NazArtikla]	Artikla]	[JedMje	edCijArtikla]	toRabat]	[Iznos]
				Ukupno bez poreza		[Sum(Iznos)]
				Porez	toPorez]	«Expr»
				Ukupno		«Expr»

❑ **StavkaZalispis** - razred koji sadrži sve pojedinačne podatke koje treba prikazati

•kontrola tipa Table/Tablix

❑ Kontrola tipa **List/Rectangle** grupirana po polju **IdDokumenta**

- *Grupiranje podataka bez izdvojenog retka/stupca*

❑ Stupac **Iznos** sadrži sumu pojedinačnih iznosa ([Sum(Iznos)]) i izraze za izračun poreza i iznosa s porezom

- = Fields!PostoPorez.Value * Sum(Fields!Iznos.Value)
- =(1 + Fields!PostoPorez.Value) * Sum(Fields!Iznos.Value)