

# **Web aplikacije (ASP.NET MVC)**

---

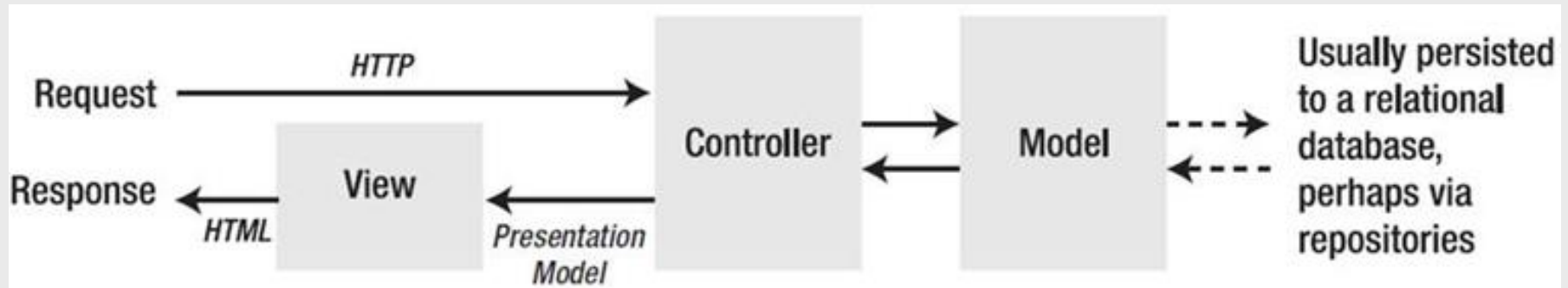
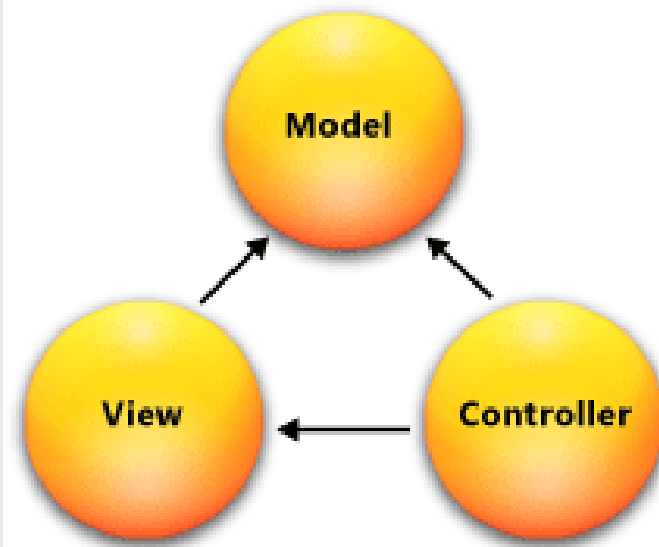
**2014/15.11**

# Nedostatci ASP.NET web formi

- ☐ ViewState proizvodi veliku količinu podataka pri svakom zahtjevu
- ☐ Kompliciran životni ciklus stranice
- ☐ Prevelike \*.cs datoteke uz često miješanje prezentacijske i aplikacijske logike
- ☐ Ograničena kontrola nad proizvedenim HTML-om
- ☐ Slaba mogućnost testiranja
  
- ☐ ASP.NET MVC kao rješenje navedenih problema
  - Objedinjena iskustva MVC implementacija u drugim jezicima
  - MVC kao koncept nastao u kasnim sedamdesetim godinama prošlog stoljeća (Smalltalk projekt unutar Xeroxa)

# ASP.NET MVC

- ❑ **Model - Pogled - Upravljač**
- ❑ **Detaljnija razrada prezentacijskog sloja**
  - Pogled definira izgled korisničkog sučelja
  - Obično vezan za objekt iz modela
- ❑ **Upravljač predstavlja prezentacijsku logiku**
  - Prima ulaz iz pogleda, obrađuje ga, puni i dohvaća model poziva niže slojeve i određuje redoslijed prikaza pogleda



- ❑ **U jednostavnim aplikacijama model objedinjava i poslovnu logiku i sloj pristupa podacima**
- ❑ **U složenijim kao model koristi se “pravi” poslovni model, a model unutar projekta služi za definiranje pomoćnih modela za lakši prikaz (“prezentacijski model”, a ne model u smislu poslovnog objekta)**

# Prednost MVC-a nad web formama

- ❑ Smanjena složenost podjelom aplikacije u model, pogled i upravljač
- ❑ Ne koristi se *ViewState* ni serverske kontrole čime se omogućava potpuna kontrola ponašanja aplikacije.
- ❑ Zahtjevi centralizirani na jedan upravljač
  - Front Controller pattern
  - bogata podrška za interpretiranje/usmjeravanje zahtjeva
- ❑ Podrška za test-driven development (TDD).
- ❑ Dobar okvir i za veće razvojne timove i za dizajnere

# Prednost web formi nad MVC-om

- ☐ **Velik broj serverskih kontrola i uobičajenih događaja**
  - nalik Windows formama
- ☐ **Page Controller pattern**
  - funkcionalnost vezana uz pojedinu stranicu
- ☐ **Jednostavno održavanje stanja web aplikacije**
  - Prikladno za Intranet aplikacije kad količina podataka nije od presudne važnosti
- ☐ **Prikladno za manje razvojne ekipe i dizajnere koji žele brzi razvoj pomoću ugrađenih komponenti**
- ☐ **Lakši (početni) razvoj zbog manje količine koda**

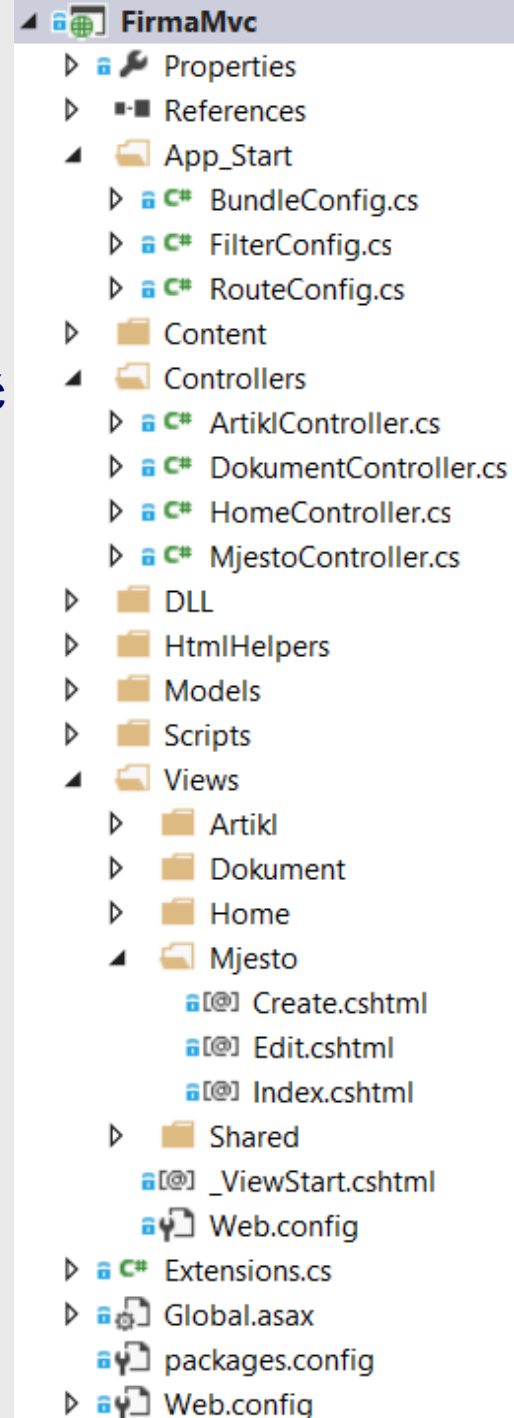
# Struktura ASP.NET MVC aplikacije

## □ Uobičajene mape i datoteke

- *Controllers*: upravljači unutar aplikacije
- *Views*: pogledi podijeljeni u podmape za svaki upravljač i Shared za zajedničke pogledе (npr. glavna stranica)
- *Models*: razred koji predstavljaju pomoćne ili domenske modele (u slučaju manje aplikacije)
- *Content*: statički sadržaj (css, teme...)
- *Scripts*: javascript biblioteke
- ...

## □ Specifično za primjer FirmaMvc

- *DLL*: sadrži dll datoteke iz primjera s višeslojnom aplikacijom
  - Poslovni i podatkovni sloj + *Firma.Framework*
  - Model za aplikaciju *FirmaMvc*
  - Dll-ovi referencirani unutar FirmaMvc
- *HtmlHelpers*: pomoćni razred za straničenje



# Inicijalne postavke MVC aplikacije

## ❑ Primjer: FirmaMvc – Global.asax.cs

- Pri prvom pokretanju aplikacije poziva se postupak *Application\_Start* iz datoteke *Global.asax.cs*
- Inicijaliziraju se
  - dijelovi (područja, eng. *areas*) aplikacije
  - filtri
  - usmjeravanja (rute)
  - paketi (grupira se više skriptnih datoteka i/ili datoteka sa stilovima)
- Kod pojedine inicijalizacije unutar mape *App\_Start*

```
namespace FirmaMvc {  
    public class MvcApplication : System.Web.HttpApplication {  
        protected void Application_Start() {  
            AreaRegistration.RegisterAllAreas();  
            FilterConfig.RegisterGlobalFilters(  
                GlobalFilters.Filters);  
            RouteConfig.RegisterRoutes(RouteTable.Routes);  
            BundleConfig.RegisterBundles(BundleTable.Bundles);  
            ...  
        }  
    }  
}
```

# Paketi datoteka

## ❑ Primjer: FirmaMvc – App\_Start \ BundleConfig.cs

- Više datoteka određenog tipa (stil, javascript, ...) pakira se u jednu datoteku
  - Poboljšanje performansi
  - Lakše promjene prilikom promjene verzije (npr. verzije jQuerya)

```
public class BundleConfig {  
    public static void RegisterBundles... {  
        bundles.Add(new ScriptBundle("~/bundles/jquery").Include(  
            "~/Scripts/jquery-{version}.js"));  
  
        bundles.Add(new StyleBundle("~/Content/css").Include(  
            ...,  
            "~/Content/Site.css"));  
    }  
}
```

- Prilikom korištenja navodi se ime paketa

```
@Styles.Render("~/Content/css")  
@Scripts.Render("~/bundles/jquery")
```



# Način rada MVC aplikacije

## ❑ Primjer: FirmaMVC – App\_Start \ RouteConfig.cs

- Korisnik od MVC aplikacije traži akciju upravljača, nakon čega aplikacija (tj. pozvani upravljač) popunjava model i vraća ažurirani pogled korisniku
- Kako prepoznati koja akcija i koji upravljač se zahtijeva?
  - Na osnovu URL-a i definiranih usmjeravanja

```
public class RouteConfig {  
    public static void RegisterRoutes(RouteCollection routes) {  
        routes.MapRoute( null, "Artikl/Page{page}",  
            new { Controller = "Artikl", action = "Index" }  
        );  
        routes.MapRoute( null, "Mjesto/Page{page}",  
            new { Controller = "Mjesto", action = "Index" }  
        );  
  
        routes.MapRoute( "Default", {controller}/{action}/{id}",  
            new { controller = "Home", action = "Index",  
                id = UrlParameter.Optional }  
        );  
  
        ...  
    }  
}
```

# Primjeri usmjeravanja

## ❑ **http://.../NazivUpravljača/Akcija/OpcionalniParametri...**

- Za traženi naziv upravljača mora postojati razred *NazivController.cs* u mapi *Controllers*
- Za traženu akciju u nekom upravljaču mora postojati postupak (s potrebnim parametrima) naziva jednakog traženoj akciji
  - Moguće definirati više istoimenih akcije za različite varijante zahtjeva (npr. GET i POST)

## ❑ **Primjer: FirmaMvc**

- **http://.../Artikl/Edit/3**
  - Upravljač Artikl(Controller) ima postupak Edit koji za parametar id ima 3
- **http://.../Artikl/Page10**
  - Upravljač Artikl izvršava pretpostavljeni postupak (Index) , a za vrijednost parametra page uzima vrijednost 10
- **http://.../Dokument/Details?page=7**
  - Upravljač Dokument(Controller) izvršava Details, a za vrijednost parametra page uzima vrijednost 7

# Uobičajeni rezultati akcije upravljača (izvedeni iz ActionResult)

Tip	Opis rezultata/povratne vrijednosti	Postupak u upravljaču
ViewResult	Prikazuje pogled	View
PartialViewResult	Prikazuje parcijalni pogled	PartialView
RedirectToRouteResult	Privremeno ili trajno preusmjerava zahtjev (HTTP kod 301 ili 302), stvarajući URL na osnovu postavki usmjeravanja	RedirectToAction RedirectToActionPermanent RedirectToRoute RedirectToRoutePermanent
RedirectResult	Privremeno ili trajno preusmjerava rezultat na određeni URL	Redirect RedirectPermanent
ContentResult	Vraća tekstualni sadržaj	Content
FileResult	Vraća binarni sadržaj	File
JsonResult	Vraća objekt serijaliziran u JSON format	Json
JavaScriptResult	Vraća JavaScript odsječak	JavaScript
HttpUnauthorizedResult	Vraća HTTP kod 401	-
HttpNotFoundResult	Vraća HTTP kod 404	HttpNotFound
HttpStatusCodeResult	Vraća određeni HTTP kod	-
EmptyResult	Bez povratne vrijednosti	-

# Glavna stranica

## ❑ Pretpostavljena glavna stranica za svaki pogled definirana u datoteci Views\\_ViewStart.cshtml

- Primjer:  FirmaMvc – Views \\_ViewStart.cshtml

```
@{  
    Layout = "~/Views/Shared/_Layout.cshtml";  
}
```

- Svaki pogled po potrebi može definirati svoju glavnu stranicu ili je uopće ne koristiti promjenom vrijednosti svojstva *Layout*

```
@{  
    Layout = null;  
}
```

- Ako se unutar pojedinog pogleda ne postavi vrijednost za *Layout* koristi se pretpostavljena glavna stranica

## ❑ Glavna stranica uključuje stil i javascript datoteke, definira navigaciju, prostor za javascript pojedinog pogleda...

- Konkretni sadržaj za neki zahtjev dobit će se pomoću *RenderBody*

# Primjer glavne stranice

## ❏ Primjer: FirmaMvc – Views\Shared\\_Layout.cshtml

```
<!DOCTYPE html>
<html lang="hr_HR" xml:lang="hr_HR"...>
<head>
  <title>@ViewBag.Title</title>
  @Styles.Render("~/Content/css")
  @Scripts.Render("~/bundles/jquery")
</head><body>
  <div id="header">
    <div class="title">RPPP - Firma.MVC</div>
  </div>
  <div id="navigation">
    @{Html.RenderAction("Navigation", "Home");}
  </div>
  <div id="content">@RenderBody()</div>
  @RenderSection("scripts", false)
  <script src="@Url.Content("~/Scripts/firma.js")"
    type="text/javascript"></script>
</body></html>
```

# Sintaksa pogleda

- ❑ Ako drugačije nije navedeno koristi se pogled čije ime odgovara pozvanoj akciji, a nalazi se u mapi Views\Pozvani upravljač
  - Ekstenzija cshtml
  - Pogled predstavlja mješavinu html i mvc koda
  - U pogledu se obično koristi jednostavni kod (npr. petlja, grananje i sl) vezan uz prikaz
  - MVC kod počinje oznakom @ iza kojeg slijedi naredba ili blok naredbi unutar vitičastih zagrada i html oznake
    - Komentari oblika @\* \*@
  - Početni redak pogleda (opcionalno) sadrži podatak koji se model koristi
    - *@model naziv razreda* koji se koristi za model
    - Konkretni vrijednosti predanog modela dobije se s *@Model*
  - Tekst izvan html oznake prefiksira se s @: ili se stavlja oznaka <text>
- ❑ Pogled može biti parcijalni
  - nema html zaglavlje niti nasljeđuje neku glavnu stranicu

# Osnovne postavke pogleda

- ❑ **Vlastiti prostori imena uključuju se u pojedinom pogledu s `@using`**
  - Alternativno mogu se uključiti za sve pogleda u `web.config`
- ❑ **Tip korištenog modela navodi se iza `@model`**
- ❑ **Naslov stranice postavlja se kroz `ViewBag.Title`**
  - proslijeđeno glavnoj stranici
- ❑ **Primjer:  FirmaMvc – Views \ Mjesto \ Index.cshtml**

```
@using FirmaMvc.HtmlHelpers
@model FirmaMvc.Models.MjestoListViewModel

@{
    ViewBag.Title = "Poštanski brojevi";
}
...
```

# Parcijalni pogled

- ❑ Istog oblika kao “obični” pogled, osim što ne koristi glavnu stranicu
  - Predstavlja dio sadržaja koji treba ubaciti unutar nekog drugog pogleda
- ❑ ***Html.RenderPartial***
  - Izvršava akciju (na određenom upravljaču) koja koristi parcijalni pogled.
    - Na upravljaču je umjesto *return View* potrebno koristiti *return PartialView*
  - Može se predati model nad kojim parcijalni pogled radi
  - Ispis parcijalnog pogleda ide direktno na izlazni tok
- ❑ ***Html.Partial***
  - Slično kao *Html.RenderPartial*, ali ne koristi direktno pisanje na izlazni tok, već vraća *string*
- ❑ **Slično: *Html.RenderAction* i *Html.Action***
  - U postojeći pogled uključuju rezultat poziva neke akcije
  - Razlika u odnosu na *RenderPartial* i *Partial*: novi zahtjev, ne predaje se model prilikom poziva



# Parcijalni pogled za navigaciju

## ❑ Primjer: FirmaMVC – Controllers \ HomeController.cs

```
public ActionResult Navigation()  
{  
    return PartialView();  
}
```

## ❑ Poveznice se stvaraju postupkom *Html.ActionLink*

- Automatski se formira konkretni URL na osnovu postavki usmjeravanja
- Nekoliko preopterećenih metoda za unos teksta poveznice, naziva akcije, naziva upravljača, vrijednosti parametara i html izgleda same poveznice
- Prefiksira se s @ ako nije prva naredba u bloku započetom s @

## ❑ Primjer: FirmaMvc – Views \ Home \ Navigation.cshtml

```
<li>@Html.ActionLink("Početna stranica", "Index", "Home")</li>  
  
<br /> Mjesta  
<li>@Html.ActionLink("Popis mjesta", "Index", "Mjesto",  
    new { page = 1 }, null)  
</li>  
<li>@Html.ActionLink("Unos novog mjesta", "Create", "Mjesto")</li>
```

# URL i model za prikaz svih mjesta

## ❑ Adresa: /Mjesto/Index

- Upravljač: Mjesto(Controller), Akcija: Index
  - Zbog postavki u Global.asax moglo je i bez navođenja akcije Index
- Opcionalno se predaje broj stranice i redoslijed sortiranja
  - Npr. /Mjesto/Page27?sort=4

## ❑ Primjer: FirmaMvc - Models \ MjestoListViewModel.cs

- Model za prikaz mjesta sadrži popis mjesta i podatke o trenutnoj stranici, ukupnom broju stranica i redoslijedu sortiranja ()
  - Mjesto je poslovni objekt iz Firma.BLL
  - Razred PagingInfo objašnjen naknadno

```
public class MjestoListViewModel {  
    public IEnumerable<Mjesto> MjestoList { get; set; }  
    public PagingInfo PagingInfo { get; set; }  
}
```

# Akcija dohvata svih mjesta

## ❑ Primjer: FirmaMVC – Controllers \ MjestoController.cs

- Upravljač dohvati podatke te napuni model i izazove prikaz pogleda istog imena.

```
public class MjestoController : Controller {  
    ...  
    public ActionResult Index(int page = 1, int sort = 1){  
        MjestoBllProvider bll = new MjestoBllProvider();  
        BusinessBaseList<Mjesto> mjesta = bll.FetchAll();  
        PagingInfo pagingInfo = new PagingInfo{  
            ...  
            TotalItems = mjesta.Count,  
        };  
        IEnumerable<Mjesto> list = mjesta.  
            OrderBy(m => m.NazMjesta);  
        list = list.Skip((page - 1) * pageSize).Take(pageSize);  
        MjestoListViewModel model = new MjestoListViewModel{  
            PagingInfo = pagingInfo,  
            MjestoList = list  
        };  
        return View(model);  
    }  
}
```

# Pregled svih mjesta (1)

## ❑ Primjer: FirmaMvc – Views \ Mjesto \ Index.cshtml

- Za svako mjesto definiran redak tablice s podacima o mjestu
  - Izgled retka definiran korištenjem css klase iz Content/Site.css
- Vrijednost modela može se dohvatiti preko svojstva Model

```
@model FirmaMvc.Models.MjestoListViewModel
...
<table class="zebra">
...
@foreach (var item in Model.MjestoList)
{
    <tr>
        <td>@item.PostBrMjesta</td>
        <td>@item.NazMjesta</td>
        <td>@item.PostNazMjesta</td>
        <td>@item.OznDrzave</td>
        ...
    }
    ...
</table>
```

## Pregled svih mjesta (2)

### ❑ Primjer: FirmaMvc – Views \ Mjesto \ Index.cshtml

- Omogućeno sortiranje i straničenje
  - Zaglavlje tablice s podacima sadrži poveznice na akciju *Index* na trenutnom upravljaču s vrijednostima za broj stranice i redoslijed sortiranja
- Poveznice koje koriste GET zahtjev stvaraju se postupkom *Html.ActionLink*

```
@model FirmaMvc.Models.MjestoListViewModel
...
<table class="zebra">
<thead>
<tr>
  <th>@Html.ActionLink("Poštanski broj", "Index",
    new { sort = 1, page = Model.PagingInfo.CurrentPage})</th>
  <th>@Html.ActionLink("Naziv mjesta", "Index",
    new { sort = 2, page = Model.PagingInfo.CurrentPage})</th>
  ...
</tr>
</thead>
</table>
```

# Poveznica za ažuriranje mjesta

## ❑ Primjer: FirmaMvc – Views \ Mjesto \ Index.cshtml

- Ažuriranje vodi na akciju *Edit* u upravljaču *Mjesto*
- Postupak *Edit* u *MjestoController* očekuje *id* kao identifikator mjesta
  - *id* se postavlja na konkretni *IdMjesta* prilikom stvaranja poveznice

```
@model FirmaMvc.Models.MjestoListViewModel
...
<table class="zebra">
...
@foreach (var item in Model.MjestoList)
{
    <tr>
        ...
        <td>@Html.ActionLink("Ažuriraj", "Edit",
            new { id = item.IdMjesta})</td>
        ...
    }
    ...
</table>
```

# Akcije ažuriranja podataka

## ❑ Primjer: FirmaMvc – Controllers \ MjestoController.cs

- Za ažuriranje zadužen postupak/akcija *Edit* u upravljaču *Mjesto*
- Potrebno razlikovati inicijalno otvaranje forme (GET zahtjev), od naknadnog slanja podataka (POST zahtjev)
  - Dva *Edit* postupka s odgovarajućim atributima *HttpGet* i *HttpPost*

```
public class MjestoController : Controller {
```

```
    [HttpGet]
```

```
    public ActionResult Edit(int id)
    {
        ...
    }
```

```
    [HttpPost]
```

```
    public ActionResult Edit(Mjesto mjesto)
    {
        ...
    }
```

# Ažuriranje podataka

## ❑ MVC sadrži skup postupaka koji stvaraju odgovarajuće html elemente za pojedino svojstvo modela

- `Html.EditorFor`, `Html.TextBoxFor`, `Html.HiddenFor`, `Html.CheckBoxFor`, ...
  - `Html.EditorFor` sam procjenjuje koju html kontrolu napraviti
  - `Html.HiddenFor` za vrijednosti koje treba sačuvati, a ne smiju se mijenjati i korisnik ih ne vidi na ekranu
- Ime html kontrole određeno imenom svojstva
- Primjer: Sljedeći odsječak iz `FirmaMvc – Views\Mjesto\Edit.cshtml`

```
@using (Html.BeginForm())  
{...  
    @Html.HiddenFor(m => m.IdMjesta)  
    @Html.EditorFor(m => m.PostBrMjesta)
```

stvara html odsječak

```
<form action="/Mjesto/Edit/6701" method="post">  
<input ... id="IdMjesta" name="IdMjesta" type="hidden"  
value="6701" />  
<input class="text-box single-line"... id="PostBrMjesta"  
name="PostBrMjesta" type="text" value="10000" />
```

- Alternativno, mogu se koristiti varijante bez sufiksa *For* navođenjem željenog imena



# Prijenos dodatnih vrijednosti pogledu

- ❑ Osim samog modela ponekad je potrebno prenijeti neke druge vrijednosti
  - poruke o pogrešci, izbor vrijednosti za padajuću listu država
- ❑ Mogu se koristiti svojstva upravljača ViewData ili ViewBag
  - ViewData (tipa ViewDataDictionary)
    - sadrži parove ključ (string) , vrijednost (objekt)
  - ViewBag (tipa dynamic)
    - Može sadržavati bilo koje svojstvo (nema sintaksne provjere prilikom kompilacije)
  - Primjer:  FirmaMvc – Controllers \ MjestoController.cs \ Edit (Get verzija)

```
public ActionResult Edit(int id)    {  
    MjestoBllProvider bll = new MjestoBllProvider();  
    Mjesto mjesto = bll.Fetch(id);  
  
    DrzavaBllProvider bllDrzava = new DrzavaBllProvider();  
    DrzavaList drzave = bllDrzava.FetchAll();  
    ViewBag.Drzave = drzave;  
    return View(mjesto);  
}
```

# Korištenje svojstva *ViewBag*/*ViewData*

- ❑ Nema sintaksne promjene te je potreban cast u (autoru) poznati tip
- ❑ Primjer:  **FirmaMvc – Views \ Mjesto \ Edit.cshtml**
  - U upravljaču (prethodna folija) unutar ViewBaga stavljen popis država (tip podatka DrzavaList)
  - Potrebno povezati državu s mjestom koje se upravo prikazuje
  - koristi se jedna od inačica postupka `Html.DropDownListFor`
    - *lamda* izraz za svojstvo koje se želi povezati
    - Kolekcija *SelectListItem* za padajuću listu

```
@Html.DropDownListFor(m => m.OznDrzave,
    ((Firma.BusinessEntities.DrzavaList) ViewBag.Drzave) .
    OrderBy(d => d.NazDrzave) .
    Select(d =>
        new SelectListItem{
            Text = d.NazDrzave,
            Value = d.OznDrzave
        }
    )
)
```

# Prihvat podataka

- ❑ Prilikom poziva akcije, na upravljaču se poziva istoimeni postupak
- ❑ Postupak može primiti:
  - cijeli model
  - FormCollection
    - parovi stringova s vrijednostima forme
  - bilo koji broj imenovanih parametara kojima se pokušava pridijeliti vrijednost
- ❑ Povezivanje s formom se vrši na osnovu svojstva *name* pojedine html kontrole
  
- ❑ Redoslijed povezivanja:
  1. Request.Form
  2. RouteData.Values
  3. Request.QueryString
  4. Request.Files

# Validacija


## ❑ Za prikaz validacijskih pogrešaka unutar pogleda koriste se:

- *Html.ValidationSummary* za pogreške na razini modela
- *Html.ValidationMessageFor* za pogreške na razini svojstva

## ❑ Validacija se provodi:

- automatski na osnovu metapodataka modela ako model ima definirane attribute iz prostora imena *System.ComponentModel.DataAnnotations*
- Eksplicitnim postavljanjem pogrešaka modela pozivom postupka *ModelState.AddModelError*
  - Predaje se naziv svojstva i opis validacijske pogreške
- MVC u slučaju validacijske pogreške postavlja odgovarajuću css klasu na html element vezan uz svojstvo s pogreškom
- Nazivi css stilova za validacijske pogreške:
  - *.input-validation-error*
  - *.field-validation-error*
  - *.validation-summary-errors*

# Proširenja (eng. extension methods)


- ❑ **Postupak koji se piše za neki postojeći tip (razred ili sučelje)**
  - Proširenje je statički postupak unutar statičkog razreda
  - Prvi parametar postupka prefiksa *this* i predstavlja tip koji se proširuje
- ❑ **Primjer:  FirmaMvc – Extensions.cs**

```
public static class Extensions {  
    public static void ValidateBusinessObject(this  
ModelStateDictionary ModelState, BusinessBase businessObject) {  
  
        businessObject.Validate();  
        foreach (var pair in businessObject.GetValidationErrors()) {  
            if (!string.IsNullOrEmpty(pair.Value))  
                ModelState.AddModelError(pair.Key, pair.Value);  
            ...  
        }  
    }  
}
```

- ❑ **Proširenje se poziva kao da je dio tog razreda ili sučelja**  
`ModelState.ValidateBusinessObject(objekt tipa BusinessBase);`
- ❑ **Proširenje ima pristup samo javnim svojstvima razreda koji proširuje!**

# Proširenja za straničenje

## ❑ Proširenje razreda *HtmlHelper* (zadužen za stvaranje html kontrola u pogledu) postupkom za prikaz stranica

- Konkretna adresa pojedine stranice parametrizirana delegatom tipa `Func<int, int, string>`
  - Za broj stranice *i* redoslijed sortiranja vraća konkretni URL
- Primjer:  FirmaMvc – HtmlHelpers \ PagingHelpers.cs

```
public static MvcHtmlString PageLinks(this HtmlHelper html,
    PagingInfo pagingInfo, Func<int, int, string> pageUrl)
{
    ...
    TagBuilder tag = new TagBuilder("a");
    tag.MergeAttribute("href", pageUrl(i, pagingInfo.Sort));
    ...
}
```

- Primjer poziva :  FirmaMvc – Views \ Mjesto \ Index.cshtml

```
@Html.PageLinks(Model.PagingInfo,
    (p, s) => Url.Action("Index", new {page = p, sort = s}))
```

# Forma za brisanje mjesta

## ❑ Za brisanje treba koristiti POST postupak

- GET postupak se preporuča za postupke koji ne mijenjaju stanja objekta ili aplikacije

## ❑ Potrebno stvoriti po jednu POST formu i jedan *submit* gumb za svako mjesto

- `Html.BeginForm`: nekoliko preopterećenih metoda za podešavanje postavki forme (akcija, upravljač, parametri...)
- U glavnoj stranici uključena `Firma.js` koja svakoj kontroli sa css stilom `delete` dodaje kod za potvrdu brisanja

## ❑ Primjer: **FirmaMvc – Views \ Mjesto \ Index.cshtml**

```
@using (Html.BeginForm("Delete", "Mjesto",
    new { id = item.IdMjesta, page = Model.PagingInfo.CurrentPage,
    sort = Model.PagingInfo.Sort })) {
    <input type="submit" value="Obriši" class="delete" />
}
```

# Prijenos podataka između zahtjeva

## ☐ **Brisanje nema vlastiti pogled**

- Nakon brisanja dolazi do preusmjerenja na akciju pregleda svih mjesta
- Tekst eventualne pogreške treba ispisati
- ViewBag ili ViewState nisu pogodni, jer se ne iscrtava pojedini pogled nego dolazi do preusmjerenja

## ☐ **Eventualna pogreška prilikom brisanja stavlja se u TempData**

- Podaci iz TempData se brišu nakon dovršetka http zahtjeva
- TempData staviti u pogled ili u glavnu stranicu



# Akcija brisanja mjesta

## ❑ Primjer: FirmaMvc – Controllers \ MjestoController.cs

- Brisanje nema vlastiti pogled
- Nakon odrađivanja posla, preusmjerava rezultat na neku akciju
- *TempData* sadrži eventualnu pogrešku

**[HttpPost]**

```
public ActionResult Delete(int id, int page = 1, int sort = 1)
{
    try{
        MjestoBllProvider bll = new MjestoBllProvider();
        Mjesto mjesto = bll.Fetch(id);
        mjesto.Delete();
        bll.Save(mjesto);
    }
    catch (Exception exc){
        TempData["Pogreska"] = exc.Message;
    }
    return RedirectToAction("Index",
        new { page = page, sort = sort });
}
```

# Prikaz artikala

## ❑ Primjer: FirmaMvc – Views \ Mjesto \ Index.cshtml

- Model sadrži kolekciju artikala i informacije o stranicama i broju elemenata
- Tablični prikaz svih artikala
- Za prikaz slike img tag s pozivom akcije *GetImage* na upravljaču *Artikl*
- Svaki redak sadrži poveznicu za ažuriranje i brisanje artikla
  - Klasično kao u primjeru mjesta ili unutar retka koristeći *jQuery* i *Ajax*

```
@model FirmaMvc.Models.ArtiklListViewModel
...
<table>
@foreach (var item in Model.ArtiklList)
{
    <tr>
        <td> </td>
        <td>@item.SifArtikla</td> <td>@item.NazArtikla</td> ...

        <td>@Html.ActionLink("Ažuriraj", "Edit", new { id =
item.SifArtikla, page = Model.PagingInfo.CurrentPage }) ...
```

# Prikaz slike artikla

## ❑ Primjer: FirmaMvc – Controllers \ ArtiklController.cs

- Slika artikla se dobije pozivom akcije *GetImage* na upravljaču *Artikl*
- Povratna vrijednost je binarni sadržaj (*FileContentResult*)
- Koristi se postupak *File*
  - Omogućava punjenje rezultata iz datoteke ili spremišta u memoriji (npr. iz polja bajtova)

```
public FileContentResult GetImage(int id)
{
    var bll = new ArtiklBllProvider();
    Artikl a= bll.Fetch(id);
    if (a != null && a.SlikaArtikla != null)
        return File(a.SlikaArtikla, "image/jpeg");
    else
        return null;
}
```

# Dodavanje nove slike

## ❑ Primjer: FirmaMvc – Views \ Artikl \ Edit.cs

- Forma mora biti tipa multipart/form-data

```
@using (Html.BeginForm("Edit", "Artikl",  
    new { page = ViewBag.CurrentPage, sort = ViewBag.Sort },  
    FormMethod.Post, new { enctype = "multipart/form-data" }))  
{ ...  
    <input type="file" name="slika" />
```

## ❑ Primjer: Controllers \ ArtiklController.cs

- Naziv parametra mora odgovarati nazivu html kontrole

```
[HttpPost]  
public ActionResult Edit(Artikl artikl,  
    HttpPostedFileWrapper slika, bool obrisiSliku, int page = 1){  
    var bll = new ArtiklBllProvider();  
    Artikl original = bll.Fetch(artikl.SifArtikla.Value);  
    ...  
    if (slika != null) {  
        original.SlikaArtikla = new byte[slika.ContentLength];  
        slika.InputStream.Read(original.SlikaArtikla, 0,  
                                slika.ContentLength);  
    }
```

# Poveznica za ažuriranje unutar retka

## ❑ Primjer: FirmaMvc – Views \ Artikl \ Index.cshtml

- Umjesto *Html.ActionLink* koristi se obična html kontrola
- Razlikuju se na osnovu css klase i vlastitog atributa *sifartikla*
  - Vlastiti atributi oblika data-naziv

```
<a class="editajax" data-sifartikla="@item.SifArtikla"
    href="#" >Ažuriraj (Ajax)</a>
```

- Nakon što se stranica učitava, *jQuery* kôd pronalazi sve takve poveznice i pridružuje odgovarajući događaj
  - *jQuery* kod koji se izvršava tek nakon učitavanja stranice počinje s `$(function() { ...`
  - Poziva se postupak *SetEditAjax* iz *firma.js* (objašnjeno na sljedećem slajdu) koji će promijeniti ponašanje poveznice

```
$(function () {
    $(".editajax").each(function () {
        SetEditAjax($(this), '@Url.Action("EditAjax",
            new { page = Model.PagingInfo.CurrentPage })');
    }); ...
});
```

# Zamjena html elementa rezultatom GET poziva

## ❑ Primjer: FirmaMvc – Scripts \ firma.js

- sprječava se uobičajeno ponašanje poveznice (*preventDefault*)
- Pozvat će se akcija predana parametrom *editAjaxUrl*
  - GET varijanta poziva
  - šifra artikla izvučena iz vlastitog atributa
- Pronalazi se nadređeni redak (redak u kojem se nalazi poveznica)
- Sadržaj retka se mijenja se sadržajem rezultata pozvane akcije

```
function SetEditAjax(btn, editAjaxUrl) {  
    $(btn).click(function (event) {  
        event.preventDefault();  
        var sifartikla = $(this).data('sifartikla');  
        var tr = $(this).parents("tr");  
        $.get(editAjaxUrl, { id: sifartikla }, function (data) {  
            tr.html(data);  
        });  
    });  
}
```

# Ažuriranje unutar retka (1)

## ❑ Primjer: FirmaMVC – Views \Artikl \ EditAjax.cshtml

- Sadržaj retka tablice s html kontrolama za unos i ispis rezultata validacije
- Svaka kontrola s atributom *sifartikla* kako bi se razlikovala od ostalih na stranici
- Namjena kontrole korištenjem css klasa

```
<td colspan="2">
    <span class="error">@TempData["Pogreska"]</span>
    @Html.ValidationSummary()
</td>
<td>
    <input type="text" value="@Model.NazArtikla"
        data-sifartikla="@Model.SifArtikla" class="nazartikla" />
</td>
<td>
    <input type="text" value="@Model.JedMjere"
        data-sifartikla="@Model.SifArtikla" class="jedmjere" />
</td>...
<input type="button" value="Spremi"
    class="saveajax" data-sifartikla="@Model.SifArtikla" />
```

## Ažuriranje unutar retka (2)

### ❑ Primjer: FirmaMVC – Views \Artikl \ EditAjax.cshtml

- Klikom na kontrolu s css klasom *saveajax* dolazi do slanja podataka (\$.post)
- Sadržaj trenutnog retka zamijeni se rezultatom akcije

```
$(function () {  
    $(".saveajax").click(function () {  
        var sifartikla = $(this).data('sifartikla');  
        var nazartikla = $(".nazartikla[data-sifartikla='" +  
            sifartikla + "']").val();  
  
        ...  
  
        var zastusluga = $(".zastusluga[data-sifartikla='" +  
            sifartikla + "']").is(':checked');  
        var url = '@Url.Action("EditAjax")';  
        var tr = $(this).parents("tr");  
  
        $.post(url, { SifArtikla: sifartikla, NazArtikla: nazartikla,  
JedMjere: jedmjere, CijArtikla: cijena, ZastUsluga: zastusluga, page :  
@ViewBag.CurrentPage }, function (data) {  
            tr.html(data);...  
        });  
    });  
});
```



# Ažuriranje unutar retka (3)

## ❑ Primjer: FirmaMVC - Controllers\ArtiklController.cs

- Ako je snimanje uspješno vraća se parcijalni pogled *Show*, a u protivnom iscrtava se parcijalni pogled *EditAjax* te se ispisuju validacijske pogreška

```
[HttpPost]
public ActionResult EditAjax(Artikl artikl) {
    Artikl original = ...dohvat artikla iz baze...
    try{
        ... izmjena vrijednosti originalnom objektu
        ModelState.ValidateBusinessObject(original);
        if (ModelState.IsValid) {
            bll.Save(original);
            return RedirectToAction("Show", ...
        }
    }
    catch (Exception exc) {
        TempData["Pogreska"] = exc.Message;
    }
    return PartialView(original);
}
```

# Poveznica za odustajanje od ažuriranja unutar retka

## ❑ Primjer: FirmaMVC – Views \ Artikl \ EditAjax.cshtml

- Klikom na gumb *Odustani* sadržaj retka se mijenja rezultatom akcije *Show*
- Potrebno postaviti povezivanja za novoučitane elemente

```
$(function () {  
    $(".cancelajax").click(function () {  
        var sifartikla = $(this).data('sifartikla');  
        var url = '@Url.Action("Show",  
                                new { page = ViewBag.CurrentPage })';  
        var tr = $(this).parents("tr");  
  
        $.get(url, { id: sifartikla }, function (data) {  
            tr.html(data);  
            SetEditAjax($(".editajax[data-sifartikla='" +  
                sifartikla + "']"),  
                '@Url.Action("EditAjax",  
                    new { page = ViewBag.CurrentPage })');  
            SetDeleteAjax(...);  
            SetConfirmDelete($(".delete[data-sifartikla='" +  
                sifartikla + "']"));  
        });  
    });  
});
```

# Brisanje artikla Ajax pozivom (1)

## ❑ Primjer: FirmaMvc – Controllers \ ArtiklController.cs

- Isključivo POST postupak koji prima šifra artikla (parametar id)
- Anonimni razred za rezultat serijaliziran u JSON oblik

```
[HttpPost]
public JsonResult DeleteAjax(int id) {
    var result = new { Successful = true,
                      ErrorMessage = string.Empty };

    try {
        ...
        artikl.Delete();
        bll.Save(artikl);
    }
    catch (Exception exc) {
        result = new { Successful = false,
                      ErrorMessage = exc.Message };
    }
    return Json(result);
}
```

# Brisanje artikla Ajax pozivom (2)

## ❑ Primjer: FirmaMvc – Scripts \ firma.js

- Klikom na gumb traži se potvrda brisanja
- POST varijanta poziva akcije iz parametra deleteAjaxUrl
  - šifra artikla izvučena iz vlastitog atributa
- Pronalazi se nadređeni redak (redak u kojem se nalazi poveznica)
- U slučaju uspješnog rezultata redak se uklanja iz strukture dokumenta

```
function SetDeleteAjax(btn, deleteAjaxUrl) {  
    $(btn).click(function () {  
        var sifartikla = $(this).data('sifartikla');  
        if (confirm('Obrisati artikl?')) {  
            $.post(deleteAjaxUrl, {id: sifartikla}, function (data) {  
                if (data.Successful) {  
                    var tr = $(btn).parents("tr");  
                    $(tr).remove();  
                }  
                else {  
                    alert(data.ErrorMessage);  
                }  
            });  
        }  
    });  
}
```

# Master-Detail primjer

## <http://.../Dokument>

- Prikazuje se pojedinačni dokument i sve njegove stavke + navigacija
- Moguće dodati ili obrisati stavku na početnom ekranu

1 2 3 4 5 6 7 8 9 10 **11** 12 13 14 15 16 17 18 19 20 21 ..100

Id dokumenta	2566	Vrsta dokumenta	R	Broj	1410	Datum	3.4.2012
Partner	ELSI (5267025)		Porez	25,00%			
Prethodni dokument			Iznos	2.183,75 kn			

[Ažuriraj](#)

Artikl	Količina	Jedinična cijena	Rabat	Iznos	
Knjiga "Visual Basic.NET"	1,00000	350,00 kn	0,00%	350,00 kn	<input type="button" value="Obriši stavku"/>
Univerzalni strujni adapter za notebook, TRUST 200NS/PW-1300p (14130)	2,00000	399,00 kn	0,00%	798,00 kn	<input type="button" value="Obriši stavku"/>
DVD/CD/MP3 player, linijski, DŽIVISI XV-N212S	1,00000	599,00 kn	0,00%	599,00 kn	<input type="button" value="Obriši stavku"/>
<input type="text" value="A/C kućni punjač za Palmu"/>	<input type="text" value="1"/>	<input type="text" value="0"/>			<input type="button" value="Dodaj stavku"/>

- Prilikom ažuriranja dokumenta moguće ažurirati zaglavlje dokumenta i svaku stavku.

Id dokumenta	2566	Vrsta dokumenta	R	Broj	1410	Datum	3.4.2012. 16:24:12
Partner	<input type="text" value="ELSI (5267025)"/>		Porez	<input type="text" value="0,25"/>			
Prethodni dokument	<input type="text" value="-----"/>		Iznos	2.183,75 kn			

Artikl	Količina	Jedinična cijena	Rabat	Iznos	
<input type="text" value="Knjiga 'Visual Basic.NET'"/>	<input type="text" value="1,00"/>		<input type="text" value="0,00"/>		Obriši stavku: <input type="checkbox"/>
<input type="text" value="Univerzalni strujni adapter za notebook"/>	<input type="text" value="2,00"/>		<input type="text" value="0,00"/>		Obriši stavku: <input type="checkbox"/>
<input type="text" value="DVD/CD/MP3 player, linijski, DŽIVISI X"/>	<input type="text" value="1,00"/>		<input type="text" value="0,00"/>		Obriši stavku: <input type="checkbox"/>

# Popis akcija i pogleda za master-detail primjer (1)

❑ **Primjer:**  **FirmaMvc – Controllers \ DokumentController.cs**

❑ **Index**

- preusmjerava zahtjev na Create ili Details

❑ **Create**

- sadrži GET i POST varijantu za unos novog dokumenta
- koristi pogled View \ Dokument \ Create
- postupak u POST varijanti prima *Dokument* kao parametar

❑ **Details**

- prikazuje jedan dokument i sve njegove stavke
- prima stranicu na osnovu koje se dohvaća konkretni dokument (1 po str.)
- pogled View \ Dokument \ Details

❑ **Edit**

- sadrži GET i POST varijante za ažuriranje
- GET varijanta prima id dokumenta kojeg treba ažurirati i trenutnu stranicu
- POST varijanta prima dokument i listu bool vrijednosti naziva BrisiStavku
- i-ti redak stavke u pogledu View \ Dokument \ Edit sadrži checkbox naziva BrisiStavku[i]

# Popis akcija i pogleda za master-detail primjer (2)

❑ **Primjer:**  **FirmaMvc – Controllers \ DokumentController.cs**

## ❑ **Delete**

- briše dokument (samo POST varijanta)
- rezultat preusmjeravanje na akciju Index (ili *Details* u slučaju pogreške)

## ❑ **DodajStavku**

- samo POST varijanta.
- prima podatke o novoj stavci i preusmjerava na akciju *Details*

## ❑ **ObrisiStavku**

- samo POST varijanta
- prima id dokumenta i id stavke koju treba obrisati
- rezultat je preusmjeravanje na akciju *Details*

❑ **Kod većine akcija koristi se parametar *page* da se nakon akcije može prikazati trenutni (nakon brisanja prethodni) dokument**

- Koristi se postupak *FetchFromPosition* iz BLL sloja
  - Dohvat 1 podatka počevši od neke stranice => dohvat n-tog dokumenta

# Inicijalni prikaz



## ❏ Primjer: FirmaMVC – Controllers \ DokumentController.cs

- Akcija *Index* u ovisnosti o postojanju dokumenata preusmjerava rezultat na prikaz prvog dokumenta ili na stranicu za stvaranje dokumenta

```
public ActionResult Index(int page = 1)
{
    var bll = new DokumentBllProvider();
    int count = bll.ItemsCount();
    if (count > 0)
    {
        return RedirectToAction("Details", new { page = 1 });
    }
    else
    {
        return RedirectToAction("Create");
    }
}
```



# Model za prikaz pojedinačnog dokumenta

- ❑ Za model se uzima pomoćni razred koji sadrži pojedinačni dokument i informacije o straničenju
- ❑ Primjer:  FirmaMVC – Models \ DokumentViewModel.cs  
 FirmaMVC – Models \ PagingInfo.cs

```
public class DokumentViewModel {  
    public Dokument Dokument { get; set; }  
    public PagingInfo PagingInfo { get; set; }  
}  
  
public class PagingInfo {  
    public int TotalItems { get; set; }  
    public int ItemsPerPage { get; set; }  
    public int CurrentPage { get; set; }  
    public int TotalPages {  
        get {  
            return (int)Math.Ceiling((decimal)TotalItems / ItemsPerPage);  
        }  
    }  
    public int Sort { get; set; }  
}
```

# Upravljač za prikaz pojedinačnog dokumenta

## ❏ Primjer: FirmaMvc – Controllers \ DokumentController.cs

- Dohvat dokumenta i njegovih stavki
- Priprema podataka za straničenje
- Priprema padajućih listi za stavki
  - Nije dio modela, već se prenosi kroz ViewBag (alternativno ViewData)

```
public ActionResult Details(int page) {  
    ...  
    var dokument = bllDokument.FetchFromPosition  
        (page-1, 1).SingleOrDefault();  
    ...  
    var bllArtikl = new ArtiklBllProvider();  
    ViewBag.Artikli = bllArtikl.FetchLookup();  
    ...  
    PagingInfo pagingInfo = new PagingInfo {  
        CurrentPage = page, ItemsPerPage = 1,  
        TotalItems = bllDokument.ItemsCount()  
    };  
    DokumentViewModel model = new DokumentViewModel {  
        Dokument = dokument, PagingInfo = pagingInfo  
    };  
    return View(model);  
}
```

# Pogled za prikaz pojedinačnog dokumenta

## ❑ Primjer: FirmaMvc – Views \ Dokument \ Details.cshtml

- Poveznice na pojedinačne dokumente (stranice) i odlazak na konkretni dokument
- Podaci dokumenta
- Poveznice za akciju ažuriranja dokumenta i brisanje dokumenta
- Za svaku stavku poveznica za brisanje
  - Ažuriranje se vrši istovremeno kad i ažuriranje dokumenta
- Redak ispod tablice za dodavanje nove stavke
  - Artikl se bira iz padajuće liste

```
@Html.DropDownList("SifArtikla",  
                    new SelectList(ViewBag.Artikli, "Key", "Text"),  
                    new { @class = "ddlartikl" }  
                    )
```

## ❑ Brisanje pojedine stavke, brisanje dokumenta i dodavanja stavke izvedeno slično kao u primjeru za Mjesto

# Ažuriranje dokumenta i stavki

## ❑ Primjer: FirmaMvc – Views \ Dokument \ Edit.cshtml

- Za vrijednosti koje se ne smiju promijeniti stvara se skrivena kontrola
  - Npr. `@Html.HiddenFor(m => m.Stavke[i].JedCijArtikla)`
- Padajuće liste pripremljene u upravljaču i pohranjene u ViewBag

```
@Html.DropDownListFor(m => m.Stavke[i].SifArtikla,  
    new SelectList(ViewBag.Artikli, "Key", "Text",  
        Model.Stavke[i].SifArtikla),  
    new { @class = "ddlartikl" })
```


- Za svaku stavku stvoren checkbox koji označa treba li obrisati stavku ili ne.
- Nazivi oblika `BrisiStavku[0...broj stavki]`
  - upravljač će primiti listu `BrisiStavku`

```
Obriši stavku: @Html.CheckBox("BrisiStavku[" + i + "]")
```

## ■ Primjer: FirmaMvc – Controllers \ DokumentController.cs

```
[HttpPost]  
public ActionResult Edit(int page,  
    Dokument dokument, List<bool> BrisiStavku) {  
    ...  
}
```

# Validacija u poslovnom modelu

- ❑ Validacija u poslovnom sloju
- ❑ za pogrešku nekog svojstva i-te stavke u zaglavlje se dodaje pogreška s nazivom **Stavka[i].NazivSvojstva**
  - Omogućava automatsko bojanje pogrešnih stavki
  - Primjer:  Firma.Win - Firma.Bll - BusinessEntities \ Dokument.cs

```
public override void Validate()
{
    ...
    foreach (var pair in Stavke[i].GetValidationErrors())
    {
        if (!string.IsNullOrEmpty(pair.Value))
        {
            SetError("Stavke[" + i + "]." + pair.Key, pair.Value);
        }
    }
    ...
}
```

# Reference

- Adam Freeman and Steven Sanderson: Pro ASP.NET MVC 3 Framework, Apress
- jQuery
  - <http://jQuery.com/>
  - jQuery UI: <http://jqueryui.com/demos/>
- CSS Tutorial
  - <http://www.w3schools.com/css/>
  - CSS selectors: [http://www.w3schools.com/cssref/css\\_selectors.asp](http://www.w3schools.com/cssref/css_selectors.asp)
- ASP.NET MVC
  - <http://www.asp.net/mvc>