ASP.NET MVC 3 REST
by mama
22.11.2011

Detaljne upute za izgradnju REST servisa i korisničkog sučelja koristeći ASP.NET MVC i C#.
Za početak je potrebno instalirati Visual Studio 2010 (Ultimate). Nakon instalacije VS2010 potrebno je s neta preuzeti ASP.NET MVC 3 i također ga instalirati. Nakon ova dva koraka može se započet s izgradnjom aplikacije. Aplikacija će se sastojati od 4 projekta reprezentativnih imena:
1. BaseModel
2. DataAccessLayer
4. Service
3. Server

i tim redom ćemo ih i graditi.
Prvo je potrebno u VSu napraviti **prazan solution**. Nazovimo ga npr. MVCREST. U MVCREST dodajmo novi projekt MVCREST.BaseModel koji ce biti tipa Class Library.

1. MVCREST.BaseModel

Projekt će sadržavati klase i sučelja naših resursa. Pretpostavimo da naša aplikacija radi s osobama i računima. Prvo je potrebno napraviti klasu Osoba i klasu Racun:
Osoba.cs:

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

namespace MVCREST.BaseModel
{
    public class Osoba
    {
        private long _id;
        public virtual long Id
        {
            get { return _id; }
            set { _id = value; }
        }

        private string _ime;
        public virtual string Ime
        {
            get { return _ime; }
            set { _ime = value; }
        }

        private string _prezime;
        public virtual string Prezime
        {
            get { return _prezime; }
            set { _prezime = value; }
        }

        private IList<Racun> _racuni;
        public virtual IList<Racun> Racuni
        {
          get { return _racuni; }
```

```
            set { _racuni = value; }
        }

        public Osoba()
        {
            _racuni = new List<Racun>();
        }

        public Osoba(long inId, string inIme,
            string inPrezime)
        {
            _id = inId;
            _ime = inIme;
            _prezime = inPrezime;
            _racuni = new List<Racun>();
        }
    }
}
```

Racun.cs:

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

namespace MVCREST.BaseModel
{
    public class Racun
    {
        private long _id;
        public virtual long Id
        {
            get { return _id; }
            set { _id = value; }
        }

        private string _brojRacuna;
        public virtual string BrojRacuna
        {
            get { return _brojRacuna; }
            set { _brojRacuna = value; }
        }

        private float _stanjeRacuna;
        public virtual float StanjeRacuna
        {
            get { return _stanjeRacuna; }
            set { _stanjeRacuna = value; }
        }

        private Osoba _vlasnik;
        public virtual Osoba Vlasnik
        {
          get { return _vlasnik; }
          set { _vlasnik = value; }
        }

        public Racun() { }

        public Racun(long inId, string inBrojRacuna,
            float inStanjeRacuna, Osoba inVlasnik)
        {
```

```
            _id = inId;
            _brojRacuna = inBrojRacuna;
            _stanjeRacuna = inStanjeRacuna;
            _vlasnik = inVlasnik;
        }
    }
}
```

Pretpostavimo da osoba ima više računa, a jedan račun pripada samo jednoj osobi. U buduće radimo sve samo sa osobama, a s računima ne, trivijalno je. Račune sam stavio samo zbog kasnijeg objašnjavanja problema. Svojstva u klasama su virtualna pošto će se u aplikaciji koristiti nHibernate za mapiranje u bazu podataka, a on zahtjeva sva svojstva koja se mapiraju budu **virtualna** i da za svaku klasu koja se mapira postoji **konstruktor bez argumenata**. Sad kad smo napravili klase naših resursa krenut ćemo u izgradnju sučelja repozitorija za CRUD operacije s našim klasama, tj. ovdje ćemo to napraviti samo za Osobu.

IOsobaRepository.cs:

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

namespace MVCREST.BaseModel
{
    public interface IOsobaRepository
    {
        void Dodaj(Osoba inOsoba);
        Osoba Dohvati(long inId);
        List<Osoba> DohvatiSve();
    }
}
```

Nakon izgradnje sučelja završili smo s projektom MVCREST.BaseModel. Potrebno je u naš solution dodati još jedan projekt tipa Class Library i nazvati ga MVCREST .DataAccessLayer.

2. MVCREST. DataAccessLayer

Ovaj projekt je samo za one koji ne znaju kako iz programa manipulirati podacima iz baze podataka i koji žele vidjeti drugačije načine od vlastitih!!! Ako ne radite ovaj projekt možete i IZBRISATI sučelje IOsobaRepository!!!

Projekt služi za mapiranje objekata u bazu podataka koristeći nHibernate mapiranje. NHibernate je skup dll datoteka s klasama koje omogućuju mapiranje. Kako bi smo to uspjeli prvo je potrebno nabaviti dll datoteke. Dll datoteke se nalaze u prilozenom projektu u folderu nHibernateBins. Bez razmišljanja što treba, a što ne, najlakše je sve dll datoteke referencirat u projekt. References->Add References->Browse i dodaju se. Osim tih dll datoteka potrebno je referencirati i MVCREST.BaseModel projekt. Ipak ćemo mapirati objekte prema klasama iz toga projekta koristeći sučelje repozitorija. References->AddReferences->Projects i odabere se željeni npr. MVCREST.BaseModel. Nakon pripreme potrebno je krenuti u izgradnju DALa. Prvo se naprave dva direktorija u DAL projektu, Mappings i Repos. Mappings će sadržavati definiciju mapiranja klasa u bazu podataka, dok će Repos sadržavati repozitorije za CRUD operacije nad resursima. Sada je potrebno na razini direktorija napraviti klasu nHibernateHelper. Koja će pomagati pri otvaranju veze s bazom. U ovom primjeru nHibernateHelper otvara komunikaciju s SQL server bazom podataka. Za drugačije konfiguracije potrebno je prosurfati internetom.

NHibernateHelper.cs

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using System.Reflection;
using NHibernate;
using NHibernate.Cfg;
using NHibernate.Driver;
using NHibernate.Dialect;
using NHibernate.Connection;
using FluentNHibernate.Cfg.Db;
using FluentNHibernate;
using NHibernate.Tool.hbm2ddl;
using MVCREST.DataAccessLayer.Mappings;

namespace MVCREST.DataAccessLayer
{
    public class NHibernateHelper
    {
        private static ISessionFactory _sessionFactory;
        public static ISession OpenSession()
        {
            if (_sessionFactory == null)
            {
                Configuration configuration = new Configuration();
                configuration.SetProperty("connection.provider",
                    "NHibernate.Connection.DriverConnectionProvider");
                configuration.SetProperty("connection.driver_class",
                    "NHibernate.Driver.SqlClientDriver");
                configuration.SetProperty("dialect",
                    "NHibernate.Dialect.MsSql2008Dialect");
                configuration.SetProperty("connection.connection_string",
                    @"Data Source=MARKO-PC\SQL2008DE;Initial Catalog=
                    RESTTEST;Integrated Security=True");
                configuration.SetProperty("query.substitutions",
                    "true=1;false=0");
                configuration.SetProperty("proxyfactory.factory_class",
                    "NHibernate.ByteCode.Castle.ProxyFactoryFactory,
NHibernate.ByteCode.Castle");
                configuration.AddMappingsFromAssembly(typeof(OsobaMap).Assembly);
                _sessionFactory = configuration.BuildSessionFactory();
            }
            return _sessionFactory.OpenSession();
        }
    }
}
```

Sta je potrebno promijeniti ako se i dalje koristi sql server:

1. linija `configuration.SetProperty("connection.connection_string", @"Data Source=MARKO-PC\SQL2008DE;Initial Catalog=RESTTEST;Integrated Security=True");`
String `@"Data Source=MARKO-PC\SQL2008DE;Initial Catalog=RESTTEST;Integrated Security=True"` zamijeniti s konekcijskim stringom na vašu bazu podataka.
2. linija `configuration.AddMappingsFromAssembly(typeof(OsobaMap).Assembly);`
klasu `OsobaMap` zamijeniti s bilo kojom map klasom. Izgradnja map klase će biti objašnjena u slijedećem koraku.

Nakon što je izgrađen helper sada treba napraviti i klase za mapiranje, u naše slućaju je to samo klasa Osoba. Pa unutar direktorija Mappings potrebno je dodati klasu OsobaMap (definira mapiranje svojstava klase Osoba u relaciju Osoba unutar baze podataka).
OsobaMap.cs:

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using MVCREST.BaseModel;
using FluentNHibernate.Mapping;

namespace MVCREST.DataAccessLayer.Mappings
{
    public class OsobaMap : ClassMap<Osoba>
    {
        public OsobaMap()
        {
            Not.LazyLoad();
            Table("Osoba");

            Id(x => x.Id)
                .Column("Id")
                .GeneratedBy
                .Native();

            Map(x => x.Ime)
                .Column("Ime")
                .Not.Nullable();

            Map(x => x.Prezime)
                .Column("Prezime")
                .Not.Nullable();

            /*Bilo kad bih mapirali i racune*/
            /*HasMany<Racun>(x => x.Racuni).
                Table("Racun").
                KeyColumn("Vlasnik").
                Inverse().
                Cascade.Delete().
                Not.LazyLoad();*/
        }
    }
}
```

Općenito linije:

```
Not.LazyLoad(); -> prilikom učitavanja kao listu ne želimo kasno učitavanje učitaj odmah!
Table("Osoba"); -> klasu mapiraj u tablicu Osoba
```

```
Id(x => x.Id)
.Column("Id")
.GeneratedBy
.Native(); -> svojstvo Id se mapira u stupac Id, to je primarni kljuc i generira se kako je to definirano u tablici (nek u tablici bude definicija auto uvećavanje da ne brinemo mi o tome)
```

```
Map(x => x.Ime)
.Column("Ime")
.Not.Nullable(); -> svojstvo Ime iz klase se mapira u stupac Ime u bazi podataka. Ne smije biti null.
```

```
Map(x => x.Prezime)
.Column("Prezime")
.Not.Nullable(); -> trivijalno imenu.
```

Mapiranje višestruke veze, kad bismo ju mapirali bi se prikazala na slijedeći način (Ako osoba ima više računa):

```
HasMany<Racun>(x => x.Racuni).
Table("Racun").
KeyColumn("Vlasnik").
Inverse().
Cascade.Delete().
Not.LazyLoad();
```

Na žalost mi to ovdje nećemo radit, ali objašnjenje bih bilo: Svojstvo Racuni se mapira u tablicu Racun, gdje je stupac stranog kljuca prema osobi stupac Vlasnik, o listi racuna se brine inverzna strana od trenutne, a inverzna se nalazi u klasi RacunMap, samo tijekom brisanja osobe izbrisi i sve racune od te osobe.

Ostale stvari se mogu naći na googleu, traži se pod „fluent nHibernate“.

Nakon izgradnje klasa za mapiranje (i baze podataka s odgovarajućim relacijama u sql serveru) može se pristupiti izgradnji repozitorija. U direktoriju Repos dodaje se klasa OsobaRepository koja implementira sučelje IOsobaRepository iz projekta MVCREST.BaseModel. Nek objekt tipa OsobaRepository u memoriji može postojat samo jedan (tj. nad klasom primjenimo singleton pattern, privatna static instanca i privatni konstruktor, javno static svojstvo za dohvat instance, prije vraćanja instance ako nije instancirana instanciraj je). Također u klasu dodajmo i izgradimo tijela metoda iz implementiranog sučelja.
OsobaRepository.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using MVCREST.BaseModel;
using NHibernate;
using NHibernate.Linq;

namespace MVCREST.DataAccessLayer.Repos
{
    public class OsobaRepository : IOsobaRepository
    {
        #region Singleton Pattern

        private static OsobaRepository _instance = null;
        public static IOsobaRepository Instance
        {
            get
            {
                if (_instance == null)
                    _instance = new OsobaRepository();
                return _instance;
            }
        }

        private OsobaRepository() { }

        #endregion

        public void Dodaj(Osoba inOsoba)
        {
            using (ISession session = NHibernateHelper.OpenSession())
            using (ITransaction transaction = session.BeginTransaction())
            {
                session.Save(inOsoba);
                transaction.Commit();
            }
        }
```

```
        public Osoba Dohvati(long inId)
        {
            using (ISession session = NHibernateHelper.OpenSession())
            using (ITransaction transaction = session.BeginTransaction())
            {
                return (from osoba in session.Linq<Osoba>()
                        where osoba.Id == inId
                        select osoba).FirstOrDefault();
            }
        }

        public List<Osoba> DohvatiSve()
        {
            using (ISession session = NHibernateHelper.OpenSession())
            using (ITransaction transaction = session.BeginTransaction())
            {
                return session.Linq<Osoba>().ToList();
            }
        }
    }
}
```

Jednostavne R operacije koristeći LINQ upite, i CUD je u potpunosti podržan od strane nHibernatea. Nakon izgradnje repozitorija završili smo s projektom MVCREST .DataAccessLayer i potrebno je napraviti servis koji će raditi C i R operaciju nad resursima. Za to je potrebno napraviti novi projekt tipa ASP.NET MVC 3 Web Application i nazvati ćemo ju MVCREST .Service. Nek projekt bude prazan i nek koristi Razor view engine. HTML5 ili bez njega, projekt uopće neće imati HTML.

3. MVCREST. Service

Projekt s aplikacijskim sučeljem na servis i servisom kao takvim. Prima zahtjeve, upravlja resursima i vraća odgovor (kao JSON rezultat). Ne sadržava HTML, osim možda kao opis apija. Imace u **MVC**u su vazni slijedeci direktoriji: **M**odels, **V**iews i **C**ontrollers, te datoteka Global.asax. Direktorij Controllers sadržava klase kontrolere (C#) koji generiraju podatke za poglede koji se nalaze u direktoriju Views (ASP.NET), koji se generiraju uz pomoć objekata iz raznih modela u direktoriju Models. Datoteka Global.asax sadržava ulaznu točku aplikacije i definicije ruta koje su važne kako bi se adrese ispravno rutirale. Primjer adrese u MVCu je http://www.mojadomena.com/imeKontrolera/imeMetode/prametar1/.../parametarn, gdje je ime controlera Controller klasa (treba postojati istoimeni poddirektorij u direktoriju Views), ime metode je neka metoda u kontroleru (treba postojati istoimeni view uveć navedenom poddirektoriju unutar direktorija Views), a parametri su argumenti metode unutar kontrolera. Npr. za adresu http://www.mojadomena.com/Osoba/Dohvati/1 bih direktoriji izgledali ovako:
Contollers -> OsobaController.cs
Views -> Osoba -> Dohvati,
a u klasi OsobaController.cs bi postojala metoda Dohvati koja prima parametar npr. long? id (u slučaju iznad s vrijednošću 1), gdje ? znači da može biti null. E pa mi u servisu nećemo imati nikakve poglede, već samo modele i kontrolere...uz mali mod datoteke Global.asax, servis će proraditi...

Prvo je u direktorij Controllers potrebno dodati novi direktorij api, a zatim u direktorij api novi kontroler imena OsobeController (prazan), gdje je nastavak **Controller** **OBVEZAN**! Nek već postojeća metoda Index služi za dohvat jednog, svih i unos novog podatka. Trebamo imati dvi verzije metode Index. Prva je za dohvat jedne osobe ili svih osoba i prima long? id (ako je id nije null vrati osobu s id, inače vrati sve osobe), a druga je za unos nove osobe i prima model osobe preko httppost metode. Prvo ćemo napraviti model osobe u direktoriju Models, nek se klasa zove ViewModelOsoba.

ViewModelOsoba.cs:

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

namespace MVCREST.Service.Models
{
    public class ViewModelOsoba
    {
        public string Ime { get; set; }
        public string Prezime { get; set; }
    }
}
```

Nakon toga popunimo prethodno kreirane metode u OsobeController klasi. Prethodno je potrebno referencirati projekte BaseModel i DataAccessLayer, te sve dll datoteke nHibernate-a. Tijekom punjenja cemo napraviti JSON klase za Osobu u direktoriju Models.

Odgovor.cs:

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

namespace MVCREST.Service.Models
{
    public class Odgovor
    {
        public const string USPJEH = "Uspjeh";
        public const string GRESKA = "Greška";

        private string _tip;
        public string Tip
        {
            get { return _tip; }
            set { _tip = value; }
        }

        private string _text;
        public string Text
        {
            get { return _text; }
            set { _text = value; }
        }

        private string _kljuc;
        public string Kljuc
        {
            get { return _kljuc; }
            set { _kljuc = value; }
        }

        public Odgovor()
        {
            _text = null;
            _tip = null;
            _kljuc = null;
        }
    }
}
```

Jedan veoma jednostavan odgovor servisa na zahtjev. Koristit ćemo ga pri unosu.
JSONOdgovorOsoba.cs

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

namespace MVCREST.Service.Models
{
    public class JSONOdgovorOsoba
    {
        public Odgovor Odgovor
        {
            get;
            set;
        }

        private Osoba _osoba;
        public Osoba Osoba
        {
            get
            {
                return _osoba;
            }
            set
            {
                _osoba = value;
                if (_osoba != null)
                {
                    if(_osoba.Racuni!=null)
                    {
                        foreach (Racun r in _osoba.Racuni)
                        {
                            r.Vlasnik = null;
                        }
                    }
                }
            }
        }

        public JSONOdgovorOsoba() { }
    }
}
```

Kombinacije odgovora i osobe koristi se za dohvat jedne osobe. Napomena je na tome da dvosmjerne veze u odgovoru ne smiju biti prisutne jer serijalizacija dvosmjernih veza dovodi do beskonačnog JSON stringa...zbog toga vlasnika u racunima u osobi postavljamo na null, objekt osoba je već taj vlasnik (ove i buduće ovakve izmjene ne spremati u repozitorij!).

JSONOdgovorOsobe.cs

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

namespace MVCREST.Service.Models
{
    public class JSONOdgovorOsobe
    {
```

```
        public Odgovor Odgovor
        {
            get;
            set;
        }

        private List<Osoba> _osobe;
        public List<Osoba> Osobe
        {
            get
            {
                return _osobe;
            }
            set
            {
                _osobe = value;
                if (_osobe != null)
                {
                    foreach (Osoba o in _osobe)
                    {
                        o.Racuni=null;
                    }
                }
            }
        }

        public JSONOdgovorOsobe()
        {
            _osobe = new List<Osoba>();
        }
    }
}
```

Kombinacija liste osoba i odgovora koristi se kao odgovor pri dohvatu svih osoba. Dvosmjerna veza je trivijalna gornjoj.

Zatim popunimo naš kontroler OsobeController.
OsobeController.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using MVCREST.BaseModel;
using MVCREST.DataAccessLayer.Repos;
using MVCREST.Service.Models;

namespace MVCREST.Service.Controllers.api
{
    public class OsobeController : Controller
    {
        public JsonResult Index(long? id)
        {
            //Dohvat jedne osobe
            if (id != null)
            {
                JSONOdgovorOsoba odgovor = new JSONOdgovorOsoba();
                odgovor.Osoba = OsobaRepository.Instance.Dohvati((long)id);
                if (odgovor.Osoba != null)
                {
                    odgovor.Odgovor = new Odgovor()
                    {
```

```
                        Tip = Odgovor.USPJEH,
                        Text = "Uspješan dohvat jedne osobe."
                    };
                }
                else
                {
                    odgovor.Odgovor = new Odgovor()
                    {
                        Tip = Odgovor.GRESKA,
                        Text = "Osoba ne postoji."
                    };
                }
                return Json(odgovor, JsonRequestBehavior.AllowGet);
            }
            else//Dohvat svih osoba
            {
                JSONOdgovorOsobe odgovor = new JSONOdgovorOsobe();
                odgovor.Odgovor = new Odgovor()
                {
                    Tip = Odgovor.USPJEH,
                    Text = "Uspješan dohvat svih osoba."
                };
                odgovor.Osobe = OsobaRepository.Instance.DohvatiSve();
                return Json(odgovor, JsonRequestBehavior.AllowGet);
            }
        }

        //Unos nove osobe
        [HttpPost]
        public JsonResult Index(ViewModelOsoba model)
        {
            Odgovor odgovor = new Odgovor();
            if (!string.IsNullOrEmpty(model.Ime))
            {
                if (!string.IsNullOrEmpty(model.Prezime))
                {
                    Osoba osoba = new Osoba(0, model.Ime, model.Prezime);
                    OsobaRepository.Instance.Dodaj(osoba);
                    odgovor.Tip = Odgovor.USPJEH;
                    odgovor.Text = "Osoba uspješno dodana.";
                }
                else
                {
                    odgovor.Tip = Odgovor.GRESKA;
                    odgovor.Text = "Prezime ne smije biti prazno!";
                    odgovor.Kljuc = "Prezime";
                }
            }
            else
            {
                odgovor.Tip = Odgovor.GRESKA;
                odgovor.Text = "Ime ne smije biti prazno!";
                odgovor.Kljuc = "Ime";
            }

            return Json(odgovor, JsonRequestBehavior.AllowGet);
        }
    }
}
```

I za kraj modirajmo Global.asax.

Global.asax:

```
using System;
```

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

namespace MVCREST.Service
{
    // Note: For instructions on enabling IIS6 or IIS7 classic mode,
    // visit http://go.microsoft.com/?LinkId=9394801

    public class MvcApplication : System.Web.HttpApplication
    {
        public static void RegisterGlobalFilters(GlobalFilterCollection filters)
        {
            filters.Add(new HandleErrorAttribute());
        }

        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            //Ruta za prikaz i objašnjenja APIja
            //moguće adrese:
            //http://domena
            //http://domena/api
            //http://domena/api/Index
            routes.MapRoute(
                "Default",
                "{controller}/{action}",
                new { controller = "api", action = "Index" } // Parameter defaults
            );

            //Ruta za osobe (dohvat i izmjena)
            //POST api/Osobe/Index -> dodavanje
            //GET api/Osobe/Index -> dohvat svih
            //GET api/Osobe/Index/1 -> dohvat s id 1
            routes.MapRoute(
                "OsobaRes", // Route name
                "api/Osobe/Index/{id}", // URL with parameters
                new { controller = "Osobe", action = "Index", id =
UrlParameter.Optional } // Parameter defaults
            );
        }

        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();

            RegisterGlobalFilters(GlobalFilters.Filters);
            RegisterRoutes(RouteTable.Routes);
        }
    }
}
```

Važna metoda je RegisterRoutes. U njoj se rute registriraju za korištenje. Prva ruta je default i ona se prikazuje kada se ukuca adresa domene, ostale rute se dodatne. Druga ruta koja se mapira je naša ruta za dodavanje/dohvat Osobe/Osoba. Više o rutiranju se može naći na internetu**. Napomena da ovdje treba napraviti dnevnik korištenja, tj. metode ovih kontrolera trebaju zapisivati dnevnik korištenja u neku datoteku, na način (u formatu) koji se traži u zadatku labosa!**

Nakon modiranja naš servis je gotov...i vraća JSON stringove, ali gdje je korisničko sučelje. Jednostavno je potrebno napraviti još jedan projekt MVCREST.Server tipa ASP.NET MVC 3 Web Application, Razor i InternetApplication, ne empty...

4. MVCREST.Server

Projekt koji služi za prikaz korisničkog sučelja...u obliku web stranice :). Servisu salje zahtjev preko web klijenta, prima odgovor i generira html koji vraća browseru...za početak je potrebno u ovaj projekt referencirati projekte MVCREST.Service (korištenje modela) i MVCREST.BaseModel (korištenje modela). Iz web.config izbrisati slijedeće stvari:

```
<connectionStrings>
    <add name="ApplicationServices"
         connectionString="data source=.\SQLEXPRESS;Integrated
Security=SSPI;AttachDBFilename=|DataDirectory|aspnetdb.mdf;User Instance=true"
         providerName="System.Data.SqlClient" />
  </connectionStrings>
```

```
<authentication mode="Forms">
      <forms loginUrl="~/Account/LogOn" timeout="2880" />
    </authentication>

    <membership>
      <providers>
        <clear/>
        <add name="AspNetSqlMembershipProvider"
type="System.Web.Security.SqlMembershipProvider"
connectionStringName="ApplicationServices"
             enablePasswordRetrieval="false" enablePasswordReset="true"
requiresQuestionAndAnswer="false" requiresUniqueEmail="false"
             maxInvalidPasswordAttempts="5" minRequiredPasswordLength="6"
minRequiredNonalphanumericCharacters="0" passwordAttemptWindow="10"
             applicationName="/" />
      </providers>
    </membership>

    <profile>
      <providers>
        <clear/>
        <add name="AspNetSqlProfileProvider"
type="System.Web.Profile.SqlProfileProvider"
connectionStringName="ApplicationServices" applicationName="/" />
      </providers>
    </profile>

    <roleManager enabled="false">
      <providers>
        <clear/>
        <add name="AspNetSqlRoleProvider" type="System.Web.Security.SqlRoleProvider"
connectionStringName="ApplicationServices" applicationName="/" />
        <add name="AspNetWindowsTokenRoleProvider"
type="System.Web.Security.WindowsTokenRoleProvider" applicationName="/" />
      </providers>
    </roleManager>
```

Izbrisati datoteke AccountController.cs i HomeController.cs, te AccountModels.cs. Također izbrisati direktorije Home i Account unutar direktorija Views! Ostala editiranja ćemo učiniti kasnije, kad bude potrebe. Sad trebamo napraviti stranicu za dodavanje osobe. Prvo napravimo klasu

OsobaController.cs u direktoriju Controllers i poddirektorij Osoba u direktoriju Views. Pravila ćemo primijenit slijedeća:
nek metoda Index bude za prikaz svih osoba, metoda Prikazi za prikaz određene osobe, te na kraju metoda Dodaj za dodavanje. Nakon izgradnje klase OsobaController.cs bih izgledala ovako:
OsobaController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using MVCREST.Service.Models;
using System.Net;
using System.Text;
using System.Web.Script.Serialization;
using System.Collections.Specialized;

namespace MVCREST.Server.Controllers
{
    public class OsobaController : Controller
    {
        //izmijeniti ovisno o portu na kojem se servis pokrece!!!
        private string domenaServisa = "http://localhost:38550";

        //prikaz svih osoba
        public ActionResult Index()
        {
            WebClient webClient = new WebClient();
            webClient.Headers["User-Agent"] = Request.UserAgent;
            string responseData = Encoding.UTF8.GetString(
                webClient.DownloadData(domenaServisa + "/api/Osobe/Index"));
            JavaScriptSerializer deserializer=new JavaScriptSerializer();
            JSONOdgovorOsobe odgovor =
deserializer.Deserialize<JSONOdgovorOsobe>(responseData);

            return View(odgovor);
        }

        //prikaz određene osobe
        public ActionResult Prikazi(long? id)
        {
            if (id == null)
                return RedirectToAction("Index");
            WebClient webClient = new WebClient();
            webClient.Headers["User-Agent"] = Request.UserAgent;
            string responseData = Encoding.UTF8.GetString(
                webClient.DownloadData(domenaServisa + "/api/Osobe/Index/" + id));
            JavaScriptSerializer deserializer = new JavaScriptSerializer();
            JSONOdgovorOsoba odgovor =
deserializer.Deserialize<JSONOdgovorOsoba>(responseData);

            return View(odgovor);
        }

        //prikazi dodavanje osobe
        public ActionResult Dodaj()
        {
            return View();
        }

        //izvrsi dodavanje osobe
        [HttpPost]
```

```
        public ActionResult Dodaj(ViewModelOsoba model)
        {
            NameValueCollection postData = new NameValueCollection();
            postData["ime"] = model.Ime;
            postData["prezime"] = model.Prezime;

            WebClient webClient = new WebClient();
            webClient.Headers["User-Agent"] = Request.UserAgent;
            string responseData = Encoding.UTF8.GetString(
                webClient.UploadValues(domenaServisa + "/api/Osobe/Index", "POST",
postData));
            JavaScriptSerializer deserializer = new JavaScriptSerializer();
            Odgovor odgovor = deserializer.Deserialize<Odgovor>(responseData);

            if (odgovor.Tip == Odgovor.GRESKA)
            {
                ModelState.AddModelError(odgovor.Kljuc, odgovor.Text);
                return View(model);
            }
            else
            {
                return RedirectToAction("Index");
            }
        }
    }
}
```

Važno je znati da tijekom komunikacije u zaglavlju poruke namijenjene servisu treba biti zapisan http agent kako bi servis ispravno zapisao korišteni preglednik. Razlika između metode Dodaj() i metode Dodaj(Argument model) je u tome što prva prikazuje stranicu sa inputima za unos osobe prvi put (HTTP GET), a druga metoda poziva se aktivacijom submita (tj. tijekom HTTP POST metode), a model je popunjen vrijednostima iz inputa.
Sada još treba napraviti viewove (3 u konačnici), a sve ih treba strpati u poddirektorij Osoba u direktoriju Views...
Index.cshtml

```
@model JSONOdgovorOsobe
@using MVCREST.Service.Models
@using MVCREST.BaseModel;
@{
    ViewBag.Title = "Lista osoba";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<fieldset>
<legend>Osobe</legend>
@if (Model.Osobe.Count == 0)
{
    <div class="editor-label">
        Nema osoba za dohvat!
    </div>
}
else
{
    foreach (Osoba o in Model.Osobe)
    {
        <fieldset>
            <legend>@o.Ime @o.Prezime (ID:@o.Id)</legend>

            <div class="editor-label">
                Id: <font style="font-weight:bold">@o.Id</font>
            </div>
```

```
            <div class="editor-label">
                Ime: <font style="font-weight:bold">@o.Ime</font>
            </div>

            <div class="editor-label">
                Prezime: <font style="font-weight:bold">@o.Prezime</font>
            </div>

            <div class="editor-label">
                <a title="Prikaži detalje" href="@Url.RouteUrl("PrikaziOsobu", new {
id = o.Id })">Prikaži detalje</a>
            </div>
        </fieldset>
    }
}
</fieldset>
```

Prikazi.cshtml

```
@model JSONOdgovorOsoba
@using MVCREST.Service.Models
@using MVCREST.BaseModel;
@{
    ViewBag.Title = "Prikaz osobe";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

@if (@Model.Osoba == null)
{
    <p>@Model.Odgovor.Text</p>
}
else
{
    <fieldset>
        <legend>@Model.Osoba.Ime @Model.Osoba.Prezime (ID:@Model.Osoba.Id)</legend>

        <div class="editor-label">
            Ime
        </div>
        <div class="editor-field">
            @Model.Osoba.Ime
        </div>

        <div class="editor-label">
            Prezime
        </div>
        <div class="editor-field">
            @Model.Osoba.Prezime
        </div>

        <div class="editor-label">
            Broj računa
        </div>
        <div class="editor-field">
            @Model.Osoba.Racuni.Count
        </div>
    </fieldset>
}
```

Dodaj.cshtml

```
@model ViewModelOsoba
```

```
@using MVCREST.Service.Models
@{
    ViewBag.Title = "Unos osobe";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

@using (Html.BeginForm())
{
    @Html.ValidationSummary(true, "Greška tijekom unosa osobe. Popravite pogreške i
pokušajte ponovo.")
    <div>
        <fieldset>
            <legend>Unos osobe</legend>

            <div class="editor-label">
                Ime
            </div>
            <div class="editor-field">
                @Html.TextBoxFor(m => m.Ime)
                @Html.ValidationMessageFor(m => m.Ime)
            </div>

            <div class="editor-label">
                Prezime
            </div>
            <div class="editor-field">
                @Html.PasswordFor(m => m.Prezime)
                @Html.ValidationMessageFor(m => m.Prezime)
            </div>

            <p>
                <input type="submit" value="Dodaj osobu" />
            </p>
        </fieldset>
    </div>
}
```

I za kraj je potrebno modirati Global.asax datoteku za ovaj projekt...
Global.asax:

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

namespace MVCREST.Server
{
    // Note: For instructions on enabling IIS6 or IIS7 classic mode,
    // visit http://go.microsoft.com/?LinkId=9394801

    public class MvcApplication : System.Web.HttpApplication
    {
        public static void RegisterGlobalFilters(GlobalFilterCollection filters)
        {
            filters.Add(new HandleErrorAttribute());
        }

        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
```

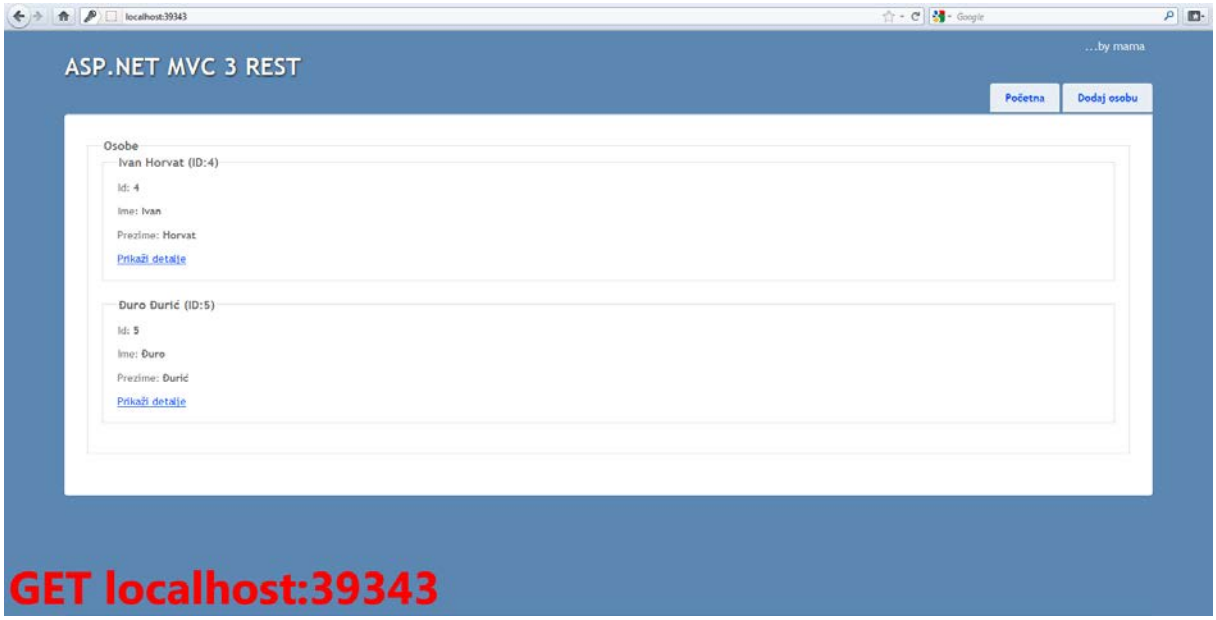
```
            routes.MapRoute(
                "Default", // Route name
                "{controller}/{action}", // URL with parameters
                new { controller = "Osoba", action = "Index" } // Parameter defaults
            );

            routes.MapRoute(
                "PrikaziOsobu", // Route name
                "Osoba/Prikazi/{id}", // URL with parameters
                new { controller = "Osoba", action = "Prikazi", id =
UrlParameter.Optional } // Parameter defaults
            );
        }

        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();

            RegisterGlobalFilters(GlobalFilters.Filters);
            RegisterRoutes(RouteTable.Routes);
        }
    }
}
```

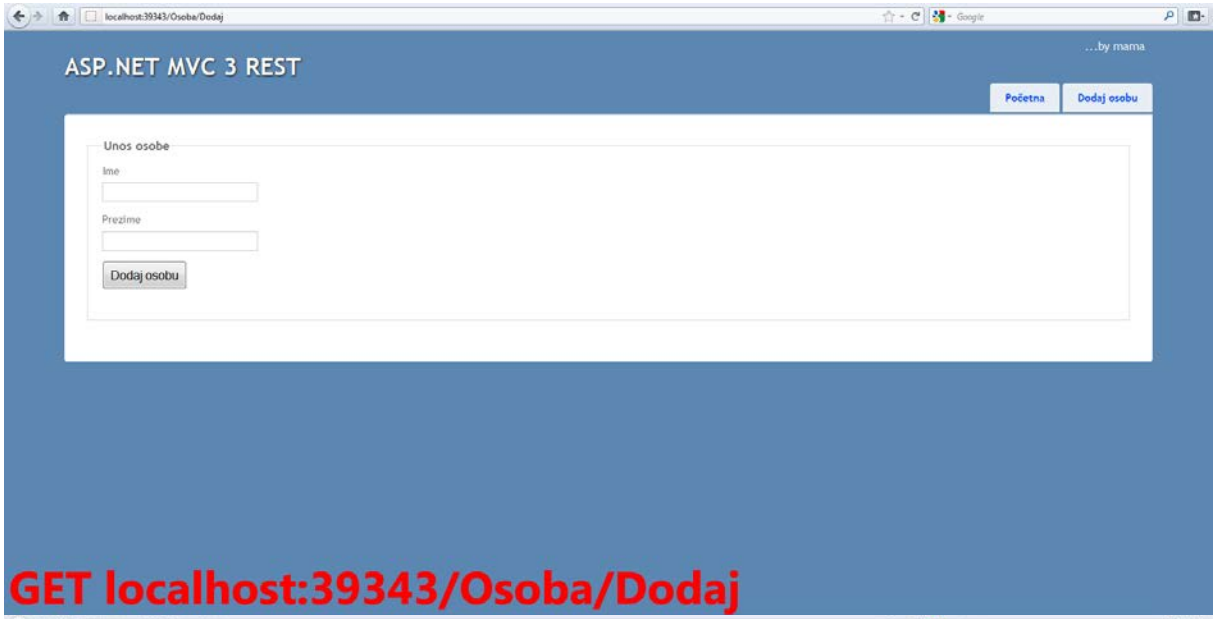
Rutu Osoba/Dodaj ne treba mapirati jer je građena na način kao defaultna ({controller}/{action}) ruta.

Za kraj se aplikacija pokreće. Važno je naglasiti da je potrebno posebno pokrenuti Service projekt i Server projekt, a pokreću se na različitim virtualnim serverima. Pokretanje je bolje i brže bez debaga. Za sve ostale nejasnoće možete pogledati solution u prilogu! Eventualne dodatke, autentifikaciju, autorizaciju, fotografije...potražite na netu. Neke dijelove koda je potrebno promijeniti kako bi aplikacija radila, a sve je obijašnjno gdje i kako...Pozdrav

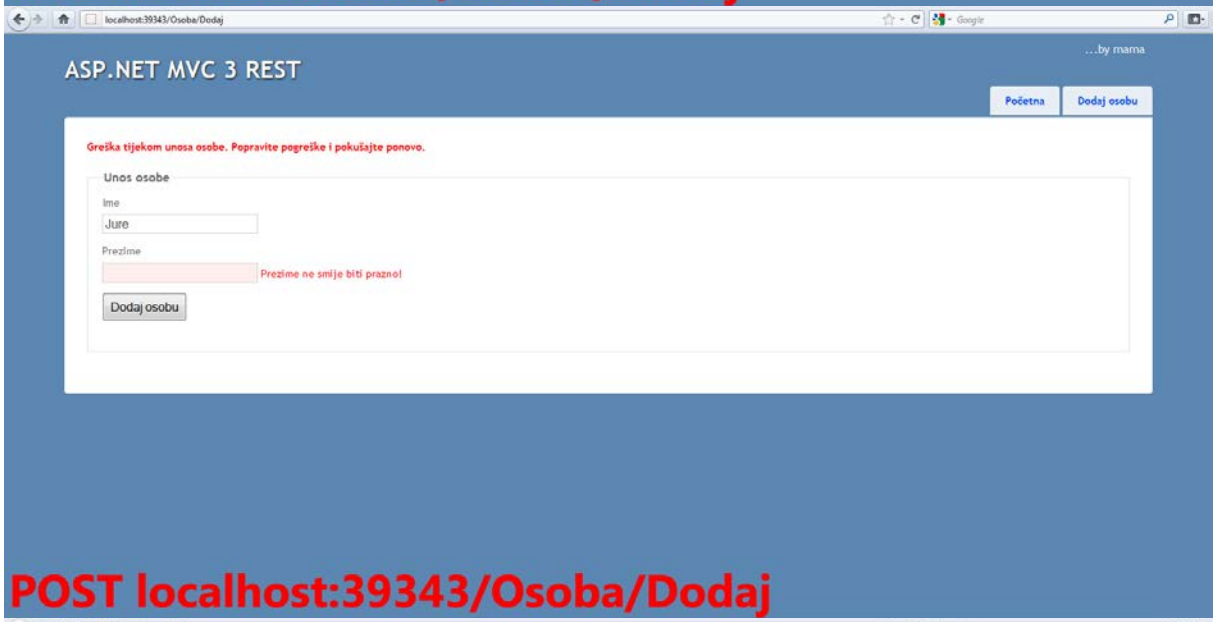
by mama

Izgled:

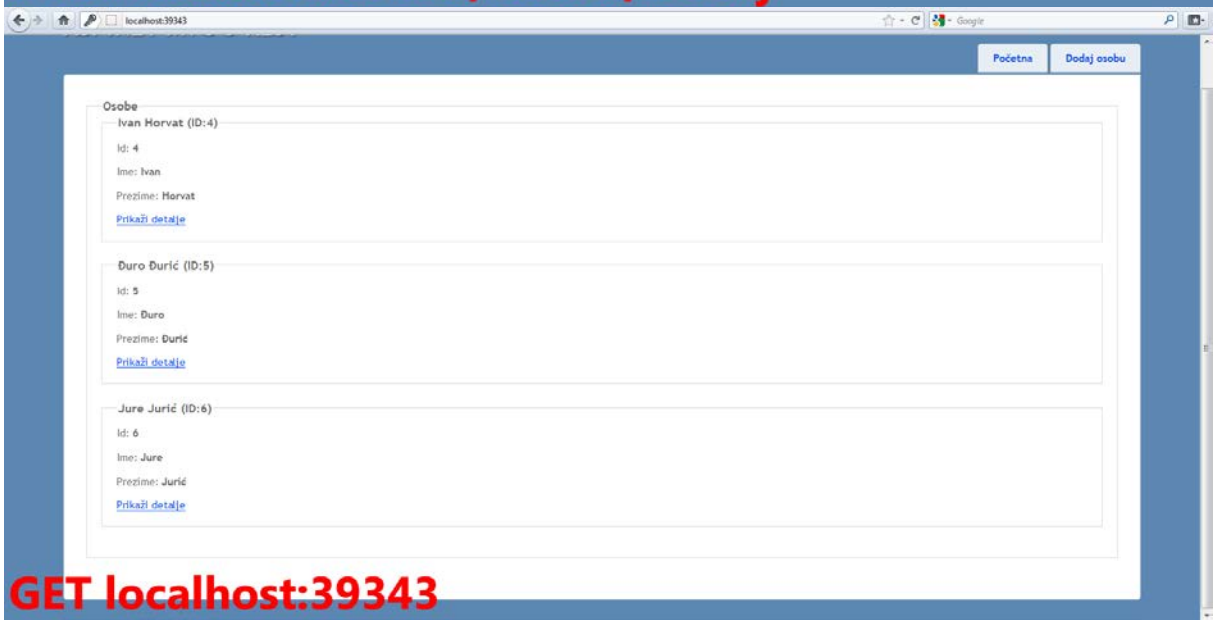




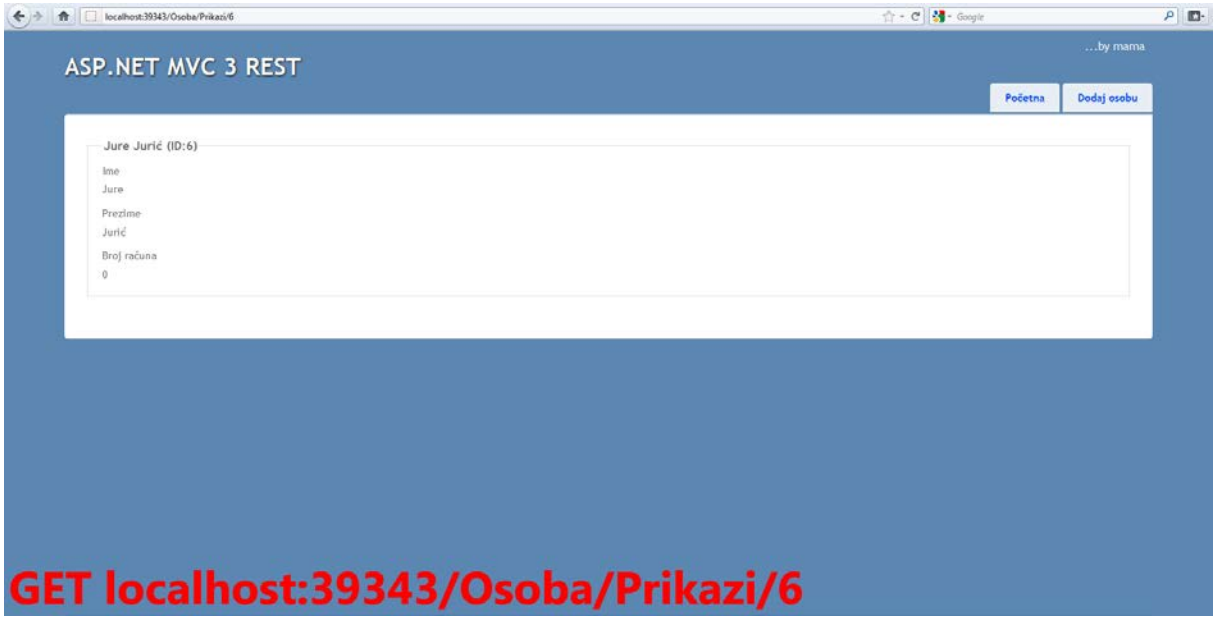
GET localhost:39343/Osoba/Dodaj



POST localhost:39343/Osoba/Dodaj



GET localhost:39343



Odgovor servisa na zahtjev: **GET http://localhost:38550/api/Osobe/Index/6**


```
{"Odgovor":{"Tip":"Uspjeh","Text":"Uspješan dohvat jedne
osobe.","Kljuc":null},"Osoba":{"Id":6,"Ime":"Jure","Prezime":"Jurić","Racun
i":[]}}
```


EDIT: Također se moguće potruditi i napraviti modifikacije tako da pozivna adresa za resurse (npr. dohvat osobe) unutar api-ja bude (bez Index) npr. http://localhost:38550/api/Osob**e**/6 ili http://localhost:38550/api/Osob**a**/6, a jedan od načina je da se **ne kreira** novi kontroler za svaki resurs unutar direktorija api, već da se metode unutar **apiController** klase nazovu po imenu resursa :)