

Web servisi

2014/15.12

Servis

- ☐ Jedna ili više funkcionalnih komponenti (ili cijeli sustav) s kojim se komunicira putem javno objavljenih i precizno definiranih sučelja
- ☐ Prima jedan ili više zahtjeva i vraća jedan ili više odgovora
- ☐ Osigurava mehanizam za pozivanje udaljenih postupaka
- ☐ Princip crne kutije
- ☐ Heterogeni klijenti
- ☐ Koristi otvorene web standarde: HTTP, XML, SOAP ...

Standardi za web servise

❑ WS-*

- Skup standarda za sigurnost, podatke i opise web servisa

❑ SOAP – Simple Object Access Protocol

- laki protokol za razmjenu informacija u distribuiranim, heterogenim okruženjima \approx HTTP + XML (zapravo XML preko HTTP-a)
- Vođen akcijama (eng. action driven)

❑ REST = Representational State Transfer

- Alternativa SOAP protokolu
- Orijentiran na resurse (eng. resource driven)

❑ WSDL - Web Services Description Language

- XML shema za opis Web servisa
- definira format postupaka koje pruža Web servis

❑ UDDI - Universal Description, Discovery, and Integration

- Discovery – postupak lociranja Web servisa koji se nalaze na nekom URL-u
- Web site objavljuje DISCO dokumente, koji vraćaju URL i WSDL opise
- <http://uddi.xml.org> - Registracijska baza Web servisa

SOA

- ❑ **SOA = Servisno orijentirana arhitektura**
 - eng. Service Oriented Architecture

- ❑ **Skup precizno definiranih, međusobno neovisnih servisa povezanih u logički jedinstvenu aplikaciju**
 - Objektno orijentirana aplikacija povezuje objekte
 - Servisno orijentirana aplikacija povezuje servise

- ❑ **Distribuirani sustav u kojem sudjeluje više autonomnih servisa međusobno šaljući poruke preko granica**
 - Granice mogu biti određene procesom, mrežom, ...

Osnovna pravila servisno orijentirane arhitekture

❑ Jasno određene granice

- Jasno iskazana funkcionalnost i struktura podataka
- Implementacija = crna kutija

❑ Neovisnost servisa

- Servis ne ovisi o klijentu, nekom drugom servisu, lokaciji i vrsti instalacije
- Verzije se razvijaju neovisno o klijentu. Objavljene verzije se ne mijenjaju.

❑ Ugovor, a ne implementacija

- Korisnik servisa i implementator servisa dijele samo listu javnih postupaka i definiciju struktura podataka
 - Dijeljeni podaci trebaju biti tipovno neutralni
 - Tipovi specifični za pojedini jezik moraju se moći pretvoriti u neutralni oblik i obrnuto
- Implementacijski postupci ostaju tajna

❑ Semantika, a ne samo sintaksa

- Logička kategorizacija servisa, smisleno imenovanje postupaka

Problemi i preporuke prilikom izradu servisa

☐ Sigurnost komunikacije

☐ Konzistentnost stanja

- Neuspjeh prilikom izvršavanja servisa ne smije ostaviti sustav u stanju pogreške

☐ Problem višenitnosti

☐ Pouzdanost i robusnost servisa

- Klijent treba znati je li servis primio poruku.
- Pogreške u servisu treba obraditi

☐ Interoperabilnost

- Tko sve može pozvati servis?

☐ Skalabilnost

☐ Brzina obrade postupka

Struktura primjera WCF

☐ Mapa *DLL*

- Smještaj dll-ova koji se naknadno referenciraju (*Firma.BLL* i *Firma.Framework*)

☐ *KlasicniWebServis* i *PozivKlasicnogWebServisa*:

- Klasični, “stari”, ASP.NET web servis (ekstenzija servisa .asmx) i konzolna aplikacija za demonstraciju poziva

☐ *Ugovor*

- WCF ugovor (struktura WCF servisa)

☐ *ImplementacijaUgovora*

- Konkretna implementacija sučelja iz projekta *Ugovor*

☐ *SmjestajWebApp* i *SmjestajWinApp*

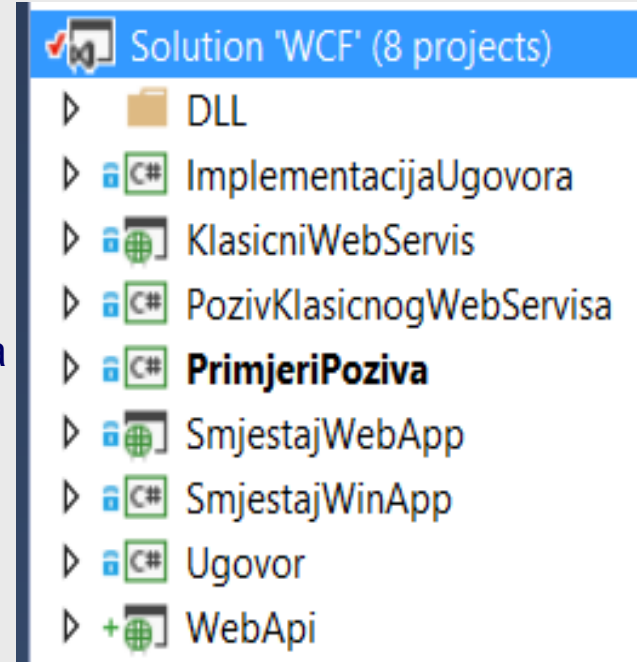
- Izlaganje WCF servisa unutar web, odnosno windows aplikacije

☐ *PrimjeriPoziva*

- Korištenje WCF servisa kao da se radi o uobičajenom web servisu
- Demonstracija korištenja WCF servisa iz .NET klijenta
- demonstracija REST poziva iz .NET klijenta


☐ *WebApi*

- Primjer izrade REST servisa kroz ASP.Net WebAPI



Primjer web servisa

❑ Primjer: Web servis za popis država

- Primjer:  WCF \ KlasicniWebServis
- Unutar web aplikacije odabrati Add New Item → Web Service
 - Stvaraju se *.asmx i *.asmx.cs datoteke
- Po potrebi dodati razrede za razmjenu podataka
 - U primjeru dodan novi razred Drzava (naziv, oznaka, iso 3 oznaka)

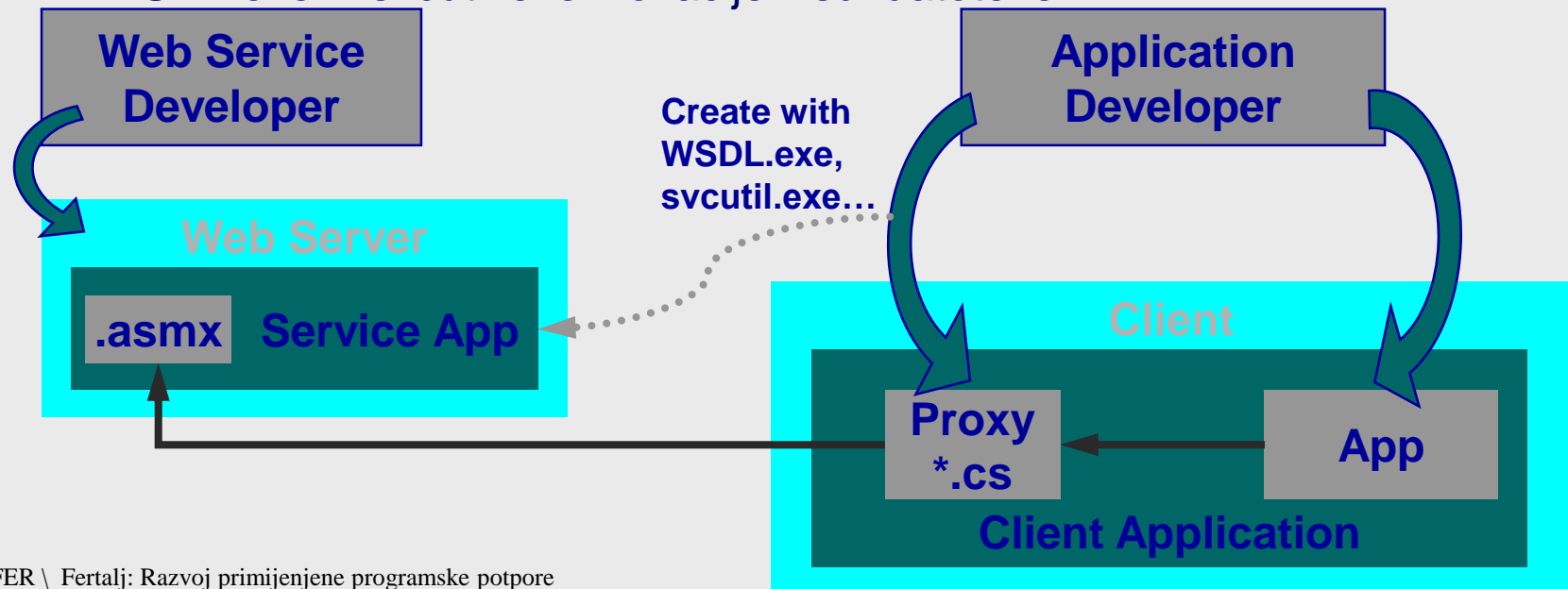
❑ Postupci koji se objavljuju ispred imena metode imaju atribut [WebMethod]

[WebMethod]

```
public List<Drzava> PopisDrzava() {  
    List<Drzava> list = new List<Drzava>();  
    var bll = new Firma.DrzavaBllProvider();  
    foreach (var d in bll.FetchAll()) {  
        list.Add(new Drzava {  
            Naziv = d.NazDrzave,  
            Oznaka = d.OznDrzave,  
            Iso3Oznaka = d.ISO3Drzave  
        });  
    }  
    return list;  
}
```


Korištenje Web servisa

- ❑ **Servis dostupan na adresi oblika <http://.../naziv.asmx>**
 - Otvaranjem *.asmx stranice u pregledniku dobiju se informacije o servisu
 - ...naziv.asmx?wsdl vraća wsdl datoteku
 - Potrebno pozvati konkretni postupak servisa (SOAP poruka)
 - Za korištenje web servisa potrebno generirati *proxy* razrede
 - Razredi koji sadrže (skrivaju) detalje komunikacije s nekim drugim objektom
 - Proxy se generira kroz razvojno sučelje ili kroz naredbeni redak pomoću WSDL.exe ili svcutil.exe i lokacije wsdl datoteke



Referenciranje web servisa iz projekta

- ❑ Desni klik na projekt → Add Web Reference → Add Service Reference

The screenshot shows the 'Add Service Reference' dialog box in Visual Studio. The 'Address' field contains 'http://localhost:44835/FirmaWebService.asmx'. The 'Services' list shows four services: 'FirmaService.svc', 'FirmaWebService.asmx', 'FirmaWebService', and 'FirmaWebServiceSoap'. The 'Namespaces' field shows 'FirmaServiceReference'.

The 'Solution Explorer' on the right shows the project structure. The 'Service References' folder is highlighted, and 'FirmaServiceReference' is listed as a service reference.

- ❑ Dodavanjem reference na web servis u projekt se dodaju
FirmaServiceReference.wsdl
FirmaServiceReference.disco
...
- ❑ U *app.config* zapisuju se postavke (način komunikacije, adresa servisa, maksimalna veličina poruke...)


Poziv web servisa

❑ Primjer: WCF \ PozivKlasicnogWebServisa

- Razredi za komunikaciju definirani su unutar web servisa (a ne klijenta).
- Dodavanjem web reference *Visual Studio* će automatski generirati omotač (*proxy*) onih razreda koje se koriste u komunikaciji.
- Generirani razredi na klijentu sadržavat će sve varijable i svojstva koja su unutar odgovarajućeg razreda na web servisu javna i namijenjena i za čitanje i za pisanje.
- Postupci definirani u razredima na web servisu ne pojavljuju se u generiranim razredima.
- Primjer pozivanja referenciranog web servisa iz kôda

```
var client = new FirmaServiceReference.FirmaWebServiceSoapClient();
foreach (var drzava in client.PopisDrzava())
{
    Console.WriteLine("{0}-{1}, {2}", drzava.Ozn,
        drzava.Naziv, drzava.ISO3);
}
```

Struktura povratnog XML-a web servisa

- ❑ **Svako svojstvo se posprema kao element u povratnom xml-u**
 - naziv elementa jednak nazivu svojstva
 - promjena strukture moguća korištenjem atributa
- ❑ **Primjer:**  **WCF\KlasicniWebServis\Drzava.cs**

```
[XmlType("D")]
public class Drzava{
    [XmlElement("Ozn")]
    public string Oznaka { get; set; }
    public string Naziv { get; set; }
    [XmlAttribute("ISO3")]
    public string Iso3Oznaka { get; set; }
}
```

- Iso3Oznaka će se umjesto kao element pospremiti kao atribut elementa oznake D (umjesto Drzava), a naziv elementa koji sadrži Oznaku bit će Ozn
- Generirani omotač na klijentu ima razred D, a ne razred Drzava
- ❑ **Ako se neko svojstvo želi isključiti iz serijalizacije potrebno je upotrijebiti atribut [XmlIgnore]**
 - Nužno ako se neko svojstvo ne može serijalizirati u xml

Nedostatci klasičnih web servisa

❑ Razredi definirani na strani web servisa

- Nepraktično kad su i klijent i servis unutar .NET okruženja i mogu imati zajednički skup razreda i postupaka
 - Generirani razredi sadrže samo javna svojstva i javne varijable
 - Postupci se ne serijaliziraju, pa se moraju ponovo pisati na klijentu

❑ Web servis (i razredi na web servisu) mora postojati da bi se klijent mogao početi razvijati

❑ Slabija mogućnost serijalizacije

- Npr. razredi koji nasljeđuju `IDictionary` ne mogu se serijalizirati (npr. `Hashtable`)

❑ Ograničenje na HTTP protokol

- Značajan pad performansi naspram alternativa u situaciji kad su i klijent i web servis na istom računalu

WCF servisi

- ❑ **WCF = Windows Communication Foundation**
- ❑ **Ne mora nužno biti web servis**
 - Mogu se koristiti različiti protokoli: HTTP, TCP, Net Pipes...
- ❑ **ABC princip : *Address, Binding, Contract***
 - Adresa i način povezivanja odvojeni od podataka
 - Zapisano u datotekama *web.config/app.config*
 - Ugovor: skup sučelja, razreda i postupaka
 - sadrže ga obje strane u komunikaciji
 - neovisan o adresi i načinu povezivanja
- ❑ **Jasnije odvajanje razreda za komunikaciju**
- ❑ **Jednostavnija promjena razreda, implementacije i smještaja**
- ❑ **Koraci razvoja WCF servisa**
 - Definirati ugovor
 - Implementirati ugovor servisa
 - Ugostiti / izložiti servis (adresa, ponašanje i način povezivanja za pristupne točke)

Protokoli povezivanja

❑ HTTP

- Uobičajeni web protokol, omogućava integraciju s otvorenim standardima i rad s različitim arhitekturama

❑ TCP

- Brzi protokol u kojem se koristi binarni format, pogodan za komunikaciju WCF – WCF posebno u lokalnoj mreži.

❑ Imenovani cjevovodi (eng. named pipes)

- Brz i pouzdan način prijenosa između WCF klijenta i WCF servisa na istom računalu. Koristi dio dijeljene memorije za komunikaciju.

❑ MSMQ (Microsoft Message Queue)

- Omogućava stavljanje poruka u red, koristan ako servis nije uvijek dostupan, a poruku može preuzeti i naknadno

❑ Vlastiti protokoli


- WCF po potrebi dozvoljava korištenje vlastitog protokola povezivanja.

Definirani tipovi povezivanja

Naziv povezivanja	Kratki opis
basicHttp(Context)Binding	Način povezivanja starih asmx servisa prema standardu WS-I Basic Profile 1.1
wsHttp(Context)Binding	Napradniji WS-* profili (Preporuča se u odnosu na basic)
wsDualHttpBinding	Isto kao i wsHttpBinding, ali za dvosmjernu komunikaciju
webHttpBinding	Koriste se za izlaganje servisa kroz različite Http zahtjeve (za REST servise koji vraćaju XML ili JSON)
wsFederationHttpBinding	WS-* profili + federated identity
netTCP(Context)Binding	Za povezivanje WCF klijenta i servisa TCP-om
netNamedPipeBinding	Optimiziran za komunikaciju klijenta i servisa na istom računalu
netPeerTcpBinding	P2P komunikacija u kojoj svaki čvor je ujedno klijent i server prema ostalim sudionicima
netMsmqBinding	Povezivanje za asinkronu komunikaciju između klijenta i servisa
msmqIntegrationBinding	Omogućava komunikaciju sa postojećim sustavima koji komuniciraju koristeći MSMQ

- ❑ **Varijante s kontekstnim povezivanjem (**ContextBinding*) omogućavaju dugotrajne servise rekonstrukcijom konteksta pomoću *cookiea* ili poruke u zaglavlju SOAP poruke**

Definiranje sučelja s postupcima u ugovoru

- ❑ **Primjer:**  **WCF \ Ugovor \ IFirmaServis.cs**
 - Opisuje postupke koji će se koristiti u komunikaciji
 - Ne određuje vrstu komunikacije i način implementacije
- ❑ **Prostor imena `System.ServiceModel`**
 - Add Reference -> **`System.ServiceModel.dll`**
- ❑ **Sučelje dekorirano atributom `[ServiceContract]`**
- ❑ **Postupak dekoriran atributom `[OperationContract]`**

```
[ServiceContract]
public interface IFirmaServis {
    [OperationContract]
    List<Drzava> PopisDrzava();
    [OperationContract]
    List<Osoba> PopisOsoba(); ...
}
```

- ❑ ****WCF podržava naprednije načine serijalizacije, pa možemo iskoristiti razred `Drzava` iz poslovnog sloja***
 - *(nije moguće kod klasičnog web servisa zbog svojstva `BllProvider` tipa `IBllProvider`)*

Implementacija sučelja


❑ **Primjer:**  WCF \ ImplementacijaUgovora \ FirmaServis.cs

❑ **Zaseban projekt**

- Referencira projekt s ugovorom (i ostale potrebne dll-ove)
- Implementira sučelja iz ugovora
- Ne ovisi o načinu povezivanja
 - Može se koristiti i kao obični dll (ne mora nužno biti izložena kao servis)

```
public class FirmaServis : Ugovor.IFirmaServis {  
    public List<Firma.BusinessEntities.Drzava> PopisDrzava() {  
        DrzavaBllProvider bll = new DrzavaBllProvider();  
        return bll.FetchAll().ToList();  
    }  
    ...  
    public List<Ugovor.Osoba> PopisOsoba() {  
        var bllProvider = new Firma.PartnerBllProvider();  
        var osobe = bllProvider.FetchAll()  
            .Where(p => p.TipPartnera == TipPartnera.Osoba)  
            .Select(p => new Osoba {  
                IdPartnera = p.IdPartnera.Value,  
                Ime = p.ImeOsobe, ...  
            })  
    }  
}
```

Izlaganje WCF servisa (1)

- ❑ Izlaže se / udomljuje (eng. *host*) konkretna implementacija nekog WCF ugovora
- ❑ Primjer:  WCF \ SmjestajWebApp \ web.config
 - Udomljavanje WCF servisa unutar web aplikacije
 - Prazna web aplikacija (ne mora nužno biti korištena samo za WCF servis)
 - Referencira projekt koji sadrži implementaciju ugovora
 - U datoteci *web.config* navodi se relativna putanja servisa i naziv konkretne implementacije

```
<system.serviceModel>
    ...
    <serviceHostingEnvironment>
        <serviceActivations>
            <add relativeAddress="Firma.svc"
                service="ImplementacijaUgovora.FirmaServis"/>
        </serviceActivations>
    </serviceHostingEnvironment>
</system.serviceModel>
```

- Servis je sad aktivan na adresi <http://.../Firma.svc>

Izlaganje WCF servisa (2)

- ❑ **Primjer:**  WCF \ SmjestajWebApp
- ❑ **Bez dodatnih postavki** WCF servis je dostupan kao web servis na adresi <http://.../Firma.svc> uz pretpostavljeni način povezivanja (**basicHttpBinding**)
 - Dostupan svima koji imaju njegovu definiciju (wsdl dokument) ili dll s ugovorom
- ❑ **Za prikaz wsdl-a treba omogućiti *Metadata exchange (MEX)***

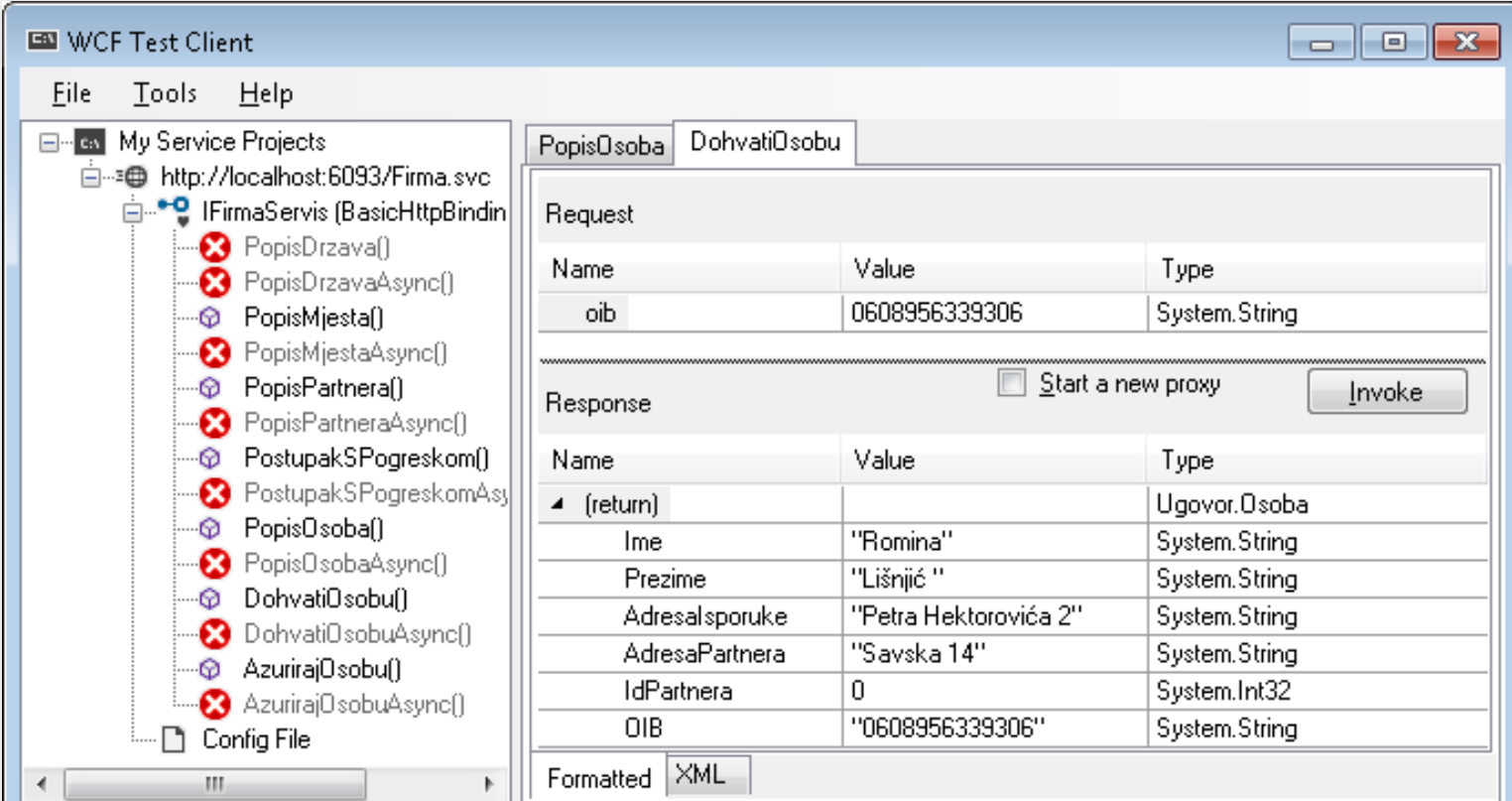
```
<system.serviceModel>
  <behaviors>
    <serviceBehaviors>
      <behavior>
        <serviceMetadata httpGetEnabled="true"/>/>
      </behavior>
    </serviceBehaviors>
  </behaviors>
</system.serviceModel>
```

- Wsdl dostupan na adresi <http://.../Firma.svc?wsdl>
- Proxy kod za poziv servisa može se dobiti sa
`svcutil.exe http://.../Firma.svc?wsdl`

Program WCF Test Client

❑ Služi za testiranje WCF servisa

-  C:\Program Files (x86)\Microsoft Visual Studio 12.0\Common7\IDE\WcfTestClient.exe
- File → Add Service → adresa servisa
- Omogućava unos ulaznih parametara, poziv i pregled rezultata servisa



WCF Test Client


File Tools Help

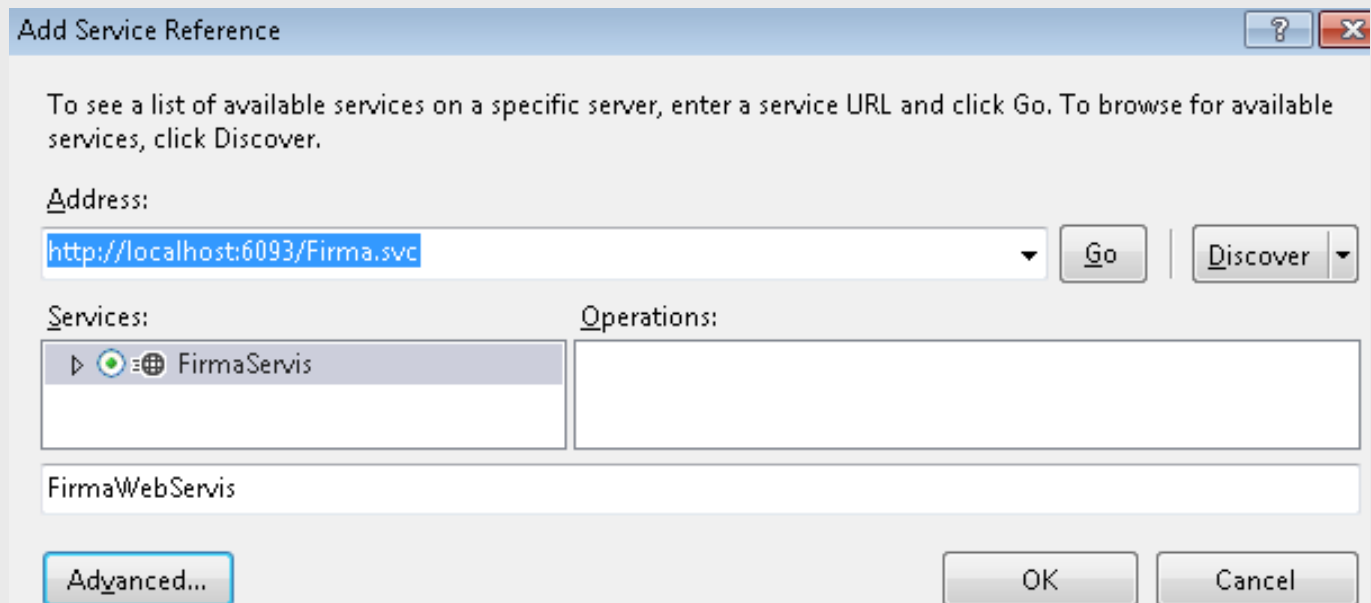
My Service Projects

- http://localhost:6093/Firma.svc
 - IFirmaServis (BasicHttpBinding)
 - PopisDrzava()
 - PopisDrzavaAsync()
 - PopisMjesta()
 - PopisMjestaAsync()
 - PopisPartnera()
 - PopisPartneraAsync()
 - PostupakSPogreskom()
 - PostupakSPogreskomAsy
 - PopisOsoba()
 - PopisOsobaAsync()
 - DohvatiOsobu()
 - DohvatiOsobuAsync()
 - AzurirajOsobu()
 - AzurirajOsobuAsync()
 - Config File

Korištenje WCF servisa dodavanjem reference (1)

❑ Korištenje WCF servisa kao da se radi o klasičnom web servisu


- Primjer:  WCF \ PrimjeriPoziva
- Desni klik na projekt → Add Web Reference → Add Service Reference
- Unijeti adresu servisa → Go
 - ili odabrati Discover ako je servis dio istog rješenja
- Unijeti naziv prostora imena (eng. *namespace*) za razrede proxy-a koji će se generirati

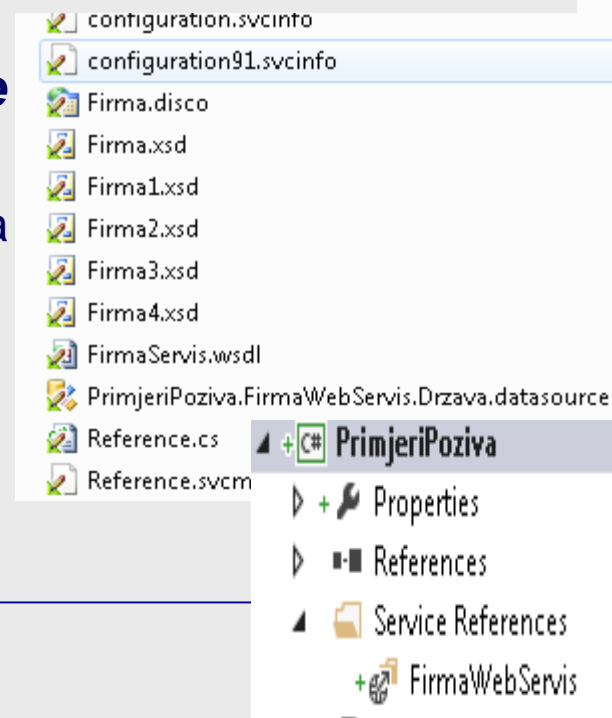


- Alternativa je koristiti svcutil kako bi se iz wsdl-a generirali razredi

Korištenje WCF servisa dodavanjem reference (2)

❑ Referencirani servis pojavljuje se pod *Service References*

- Na disku automatski stvoreno nekoliko datoteka za generirani proxy
- U datoteku app.config automatski dodana adresa servisa i način povezivanja
- Primjer:  WCF \ PrimjeriPoziva \ app.config



```
<system.serviceModel>
...
<binding name="BasicHttpBinding_IFirmaServis" />
...
<client>
  <endpoint address="http://localhost:6093/Firma.svc"
    binding="basicHttpBinding"
    bindingConfiguration="BasicHttpBinding_IFirmaServis"
    contract="FirmaWebServis.IFirmaServis"
    name="BasicHttpBinding_IFirmaServis" />
</client>
```

Korištenje WCF servisa dodavanjem reference (3)

❑ Primjer: WCF \ PrimjeriPoziva \ FormKlasicno

- Automatski generiran klijent za poziv web servisa
- Sadrži sve postupke iz ugovora i njihove asinkrone varijante

```
private async void btnLoad_Click(...) {  
    var client = new FirmaWebServis.FirmaServisClient();  
    var osobe = new List<Ugovor.Osoba>(  
        await client.PopisOsobaAsync());  
    ...  
}
```

❑ Prilikom dodavanja reference na servis uobičajeno je uključena opcija ***Reuse types in all referenced assemblies***

- Ako projekt referencira dll koji sadrži razrede iz WCF servisa onda ne dolazi do stvaranja novih razreda
 - Npr. u primjeru je referenciran Ugovor.dll, pa se koristi Ugovor.Osoba
- Inače kreiraju se razredi istih imena, ali različitog prostora imena NazivProjekta.NazivReference (npr. PrimjeriPoziva.FirmaWebServis)
 - Npr. FirmaWebServis.Drzava ≠ Firma.Drzava !

Korištenje WCF servisa bez dodavanje reference


❑ Primjer:  WCF \ PrimjeriPoziva \ FormWCF

❑ Za poziv WCF servisa korištenjem WCF mehanizama potrebno je

- referencirati ugovor
- definirati pristupnu točku u konfiguracijskoj datoteci (vidi sljedeći slajd)
- koristiti *ChannelFactory* za stvaranje komunikacijskog kanala
 - stvara klijenta za komunikaciju koji sadrži sve metode iz ugovora
 - u primjeru klijent je takav da implementira sučelje *Firma.IFirmaServis*

```
private void btnLoad_Click(...) {  
    string endPointName = ...  
    using (ChannelFactory<Ugovor.IFirmaServis> factory =  
        new ChannelFactory<Ugovor.IFirmaServis>(endPointName))  
    {  
        Ugovor.IFirmaServis client = factory.CreateChannel();  
        var osobe = client.PopisOsoba();  
        ...  
    }  
}
```

Definicija pristupne točke u klijentu

- ❑ Podaci o pristupnoj točki definiraju se unutar konfiguracijske datoteke
- ❑ Primjer:  WCF \ PrimjeriPoziva \ app.config
 - Puni naziv razreda iz ugovora (atribut *contract*)
 - Naziv pristupne točke (atribut *name*)
 - Adresu servisa (atribut *address*)
 - Način povezivanja (atribut *binding*)
 - basicHttpBinding, wsHttpBinding, netNamePipeBinding, customBinding...

```
<system.serviceModel>
  <client>
    <endpoint contract="Ugovor.IFirmaServis"
              name="FirmaEndPointWebServis"
              address="http://localhost:6093/Firma.svc"
              binding="basicHttpBinding" />
    <endpoint contract="Ugovor.IFirmaServis"
              name="FirmaEndPointLokalno"
              address="net.pipe://localhost/FirmaServis"
              binding="netNamedPipeBinding" />
  </client>
</system.serviceModel>
```

Konfiguriranje WCF postavki

❑ Postavke WCF klijenta navode se programski ili u konfiguracijskim datotekama

- Može se izdvojiti u posebnu konfiguracijsku datoteku koja se uključi iz datoteke web.config
 - Npr. `<system.serviceModel> <bindings configSource="ime.config" />`

❑ Grafički alat za pregled i definiranje postavki unutar VS-a

- Izbornik Tools → WCF Service Configuration Editor

C:\TFS_RPPP\RPPP\RPPP-primjeri\WCF\PrimjeriPoziva\App.config - Microsoft Service Configuration Editor

File Help

Configuration

- Services
- Client
 - Metadata
 - Endpoints
 - FirmaEndPointWebServis**
 - FirmaEndPointLokalno
 - BasicHttpBinding_IFirmaServis
 - Bindings
 - Prilagodjeno (basicHttpBinding)
 - BasicHttpBinding_IFirmaServis


Client Endpoint

General Identity Headers

▲ (Configuration)	
Name	FirmaEndPointWebServis
▲ (General)	
Address	http://localhost:6093/Firma.svc
BehaviorConfiguration	
Binding	basicHttpBinding
BindingConfiguration	Prilagodjeno
Contract	Ugovor.IFirmaServis
EndpointConfiguration	

Povezivanje WCF servisa korištenjem cjevovoda

❑ Smještaj implementacije ugovora unutar Windows aplikacije

- Demonstrira mogućnost izlaganja WCF servisa korištenjem drugačijih protokola
- Koristi se razred *ServiceHost*
- Komunikacija korištenjem cjevovoda
 - Cjevovod prikladan ako su i servis i klijent na istom računalu
- Primjer:  WCF \ SmjestajWinApp \ Form1

```
ServiceHost host =  
    new ServiceHost(typeof(ImplementacijaUgovora.FirmaServis),  
        new Uri[] { new Uri("net.pipe://localhost") });  
  
NetNamedPipeBinding binding = new NetNamedPipeBinding();  
  
host.AddServiceEndpoint(typeof(Ugovor.IFirmaServis),  
    binding, "FirmaServis");  
  
host.Open();
```

❑ Na klijentu je potrebno samo promijeniti parametre pristupne točke

WCF i serijalizacija podataka

- ❑ WCF za serijalizaciju u XML koristi *DataContractSerializer*
 - Serijaliziraju se sve javne varijable i javna svojstva za koja je omogućeno čitanje i pisanje
- ❑ Izostavljanje pojedinog svojstva ili varijable vrši se upotrebom atributa *IgnoreDataMember* (opt-out princip)
- ❑ Ako se za razred postavi atribut *DataContract* tada se serijaliziraju samo oni članovi koji imaju atribut *DataMember* (opt-in princip)
- ❑ Primjer:  WCF\Ugovor\Mjesto.cs

```
public class Mjesto{  
    public int Pbr { get; set; }  
    [IgnoreDataMember]  
    public string NazMjesta  
        { get; set; }  
    public string NazDrzave  
        { get; set; }  
    public string PostNazMjesta  
        { get; set; }  
}
```

≡

```
[DataContract]  
public class Mjesto {  
    [DataMember]  
    public int Pbr { get; set; }  
    public string NazMjesta  
        { get; set; }  
    [DataMember]  
    public string NazDrzave  
        { get; set; }  
    [DataMember]  
    public string PostNazMjesta  
        { get; set; }  
}
```

Unaprijed nepoznati tipovi povratnih vrijednosti

- ❑ Stvarni tip povratne vrijednosti nije poznat u sljedećim slučajevima
 - Postupak vraća sučelje
 - Postupak vraća bazni razred
 - Postupak vraća *Object*
- ❑ Za navedene primjere potrebno je najaviti moguće povratne tipove (osim ako se ne radi o primitivnim tipovima)
- ❑ Koriste se atributi *KnownType* i *ServiceKnownType*
 - Navode se svi poznati izvedeni razrede nekog razreda ili sučelja
 - *KnownType* - koristi se kao atribut nekog tipa
 - *ServiceKnownType* - koristi se kao atribut određenog postupka
- ❑ Alternativa navođenju tipa unutar atributa *(Service)KnownType*
 - Navođenje u konfiguracijskoj datoteci
 - Navodi se naziv postupka koji vraća tip izvedenog razreda

Primjer navođenja izvedenih razreda

❑ Primjer: WCF \ Ugovor \ IFirmaServis.cs

- Osoba nasljeđuje Partnera
- Povratna vrijednost tipa Partner

```
[OperationContract]  
[ServiceKnownType(typeof(Osoba))]  
[ServiceKnownType(typeof(Tvrtka))]  
List<Partner> PopisPartnera();
```

❑ Alternativno, može se navesti i unutar konfiguracijske datoteke (potrebno i na serveru i na klijentu)

```
<system.runtime.serialization>  
  <dataContractSerializer>  
    <declaredTypes>  
      <add type="Ugovor.Partner,Ugovor">  
        <knownType type="Ugovor.Osoba,Ugovor" />  
        <knownType type="Ugovor.Tvrtka,Ugovor" />  
      </add>  
    </declaredTypes>  
  </dataContractSerializer>  
</system.runtime.serialization>
```

Iznimke (pogreške) u radu s WCF servisima

❑ Pogreška u postavkama klijenta

- npr. ne postoji definicija pristupne točke => *InvalidOperationException*

❑ Pogreška u komunikaciji (***CommunicationException***)

- Neusklađeni načini povezivanja
 - npr. servis izložen kroz *basic http* način povezivanja, a klijent pretpostavlja WS-standard => *CommunicationException*
- Servis nije dostupan => *EndpointNotFoundException*
- Neuhvaćena (ili nenajavljena) iznimka u servisu => *FaultException*

❑ Za najavu (tipa) pogreške koja se može očekivati koristi se atribut ***FaultContract***

- Primjer:  WCF\Ugovor\FirmaServis.cs

```
[OperationContract]
[FaultContract(typeof(Pogreska))]
string PostupakSPogreskom(VrstaPogreske vrsta);
```

❑ Za bacanje iznimke s predviđenom pogreškom koristi se ***FaultException***

- Primjer:  WCF\ImplementacijaUgovora\FirmaServis.cs

```
throw new FaultException<Pogreska>(
    new Pogreska { OpisPogreske = "Neki opis pogreške..." });
```



Primjer klijenta koji ispravno rukuje iznimkama

❏ Primjer: WCF \ PrimjeriPoziva \ FormWCF.cs

```
try{
    using (ChannelFactory<Ugovor.IFirmaServis> factory = new
        ChannelFactory<Ugovor.IFirmaServis>(endPointName)) {
        Ugovor.IFirmaServis client = factory.CreateChannel();
        ...
    }
}
catch (FaultException<Pogreska> fep) {
    ... fep.Detail.OpisPogreske ...
}
catch (FaultException fexc) {
    ...
}
catch (EndpointNotFoundException epexc) {
    ...
}
catch (CommunicationException commexc) {
    ...
}
catch (InvalidOperationException invexc) {
    ...
}
```

Postavke povezivanja

❑ Postavke povezivanja moguće je modificirati

- Primjer:  WCF \ PrimjeriPoziva \ app.config
- Unutar elementa *serviceModel* -> *bindings* -> *tippovezivanja* definiraju se imenovane postavke povezivanja s promijenjenim vrijednostima
- Inicijalnim postavkama količina podataka je ograničena na 65536 okteta

❑ U definiciji pristupne točke osim tipa povezivanja navodi se i naziv konfiguracije (*bindingConfiguration*)

```
<system.serviceModel>
  <bindings>
    <basicHttpBinding>
      <binding name="Prilagodjeno"  maxBufferSize="5000000"
        maxReceivedMessageSize="5000000"/>
    </basicHttpBinding>
  </bindings>
  <client>
    <endpoint name="FirmaEndPointWebServis"
      contract="Ugovor.IFirmaServis"
      address="http://localhost:6093/Firma.svc"
      binding="basicHttpBinding"
      bindingConfiguration="Prilagodjeno"
      />
  </client>
</system.serviceModel>
```

REST

- ❑ **REST = Representational State Transfer**
 - Alternativa SOAP protokolu
 - Nije protokol, već obrazac izrade i poziva web servisa
- ❑ **Slanjem SOAP poruke na pristupnu točku provjerava se sadržaj i na osnovu sadržaja se određuje postupak koji se treba izvršiti**
- ❑ **Korištenjem REST-a postupak se određuje na osnovu url-a i http metode (GET, POST, DELETE, PUT)**
- ❑ **Prednosti REST-a u odnosu na SOAP**
 - Veći broj potencijalnih klijenata
 - Dovoljna je podrška za Http i XML ili JSON
 - Nije potrebno implementirati složene WS-* standarde
 - Lakše *cacheiranje*
 - Manje poruke
 - POX (Plain old XML) – XML poruke bez SOAP zaglavlja
 - JSON – (JavaScript Object Notation)
- ❑ **Nedostatak**
 - Nema wsdl-a, veća mogućnost nepravilne poruke

REST i vrste HTTP poziva

❑ GET

- Služi za dohvat podataka
- Vraća rezultat ili HTTP status 404

❑ POST

- Služi za kreiranje novog podatka
- Za uspješno stvoreni podatak vraća HTTP status 201

❑ PUT

- Služi za ažuriranje podatka
- Nakon uspješnog ažuriranja vraća HTTP status 200 ili 204

❑ DELETE

- Služi za brisanje podatka
- Ako je podatak uspješno obrisani ili je već ranije bio obrisani vraća HTTP status 200 ili 204

REST i WCF


- ❑ Podrška unutar prostora imena *System.ServiceModel.Web*
- ❑ Koristi se način povezivanja *webHttpBinding*
- ❑ Atributi *WebGet* i *WebInvoke*

- Primjer:  WCF \ Ugovor \ IFirmaServis.cs

```
[OperationContract]
[WebGet(UriTemplate = "GetOsoba/{idOsobe}",
        BodyStyle=WebMessageBodyStyle.Wrapped) ]
Osoba DohvatiOsobu(string idOsobe);
```

- Definira putanju do postupka i način zadavanja parametara
 - Parametri navedeni u putanji moraju biti tipa *string*
- Uz pristupnu točku na adresi *rest* (vidi slajd *Definicija pristupne točke*) pozivom url-a `http://localhost:6093/Firma.svc/rest/GetOsoba/10` izvršit će se postupak *DohvatiOsobu* za osobu s id-om 10
- Svojstvo *BodyStyle* određuje da li će rezultat biti dodatno ugniježđen unutar *xml* ili *json* elementa

Definicija pristupne točke

- ❑ Ukoliko se u konfiguracijskoj datoteci ne navede niti jedna pristupna točka podrazumijeva se pristupna točka s načinom povezivanja *basicHttpBinding*
 - Navođenjem bilo koje druge, ova se mogućnost gubi
- ❑ Ista implementacija servisa može biti izložena na više načina povezivanja, ali na različitim adresama
- ❑ Navodi se naziv razreda implementacije te sve pristupne točke
 - Za svaku točku se navodi relativna adresa, razred ugovora i način povezivanja te opcionalno naziv postavki ponašanja (*behaviorConfiguration*)
 - Za pristupnu točku za koju adresa nije navedena koristi se prazni string
 - Primjer  WCF \ SmjestajWebApp \ web.config

```
<services>
  <service name="ImplementacijaUgovora.FirmaServis">
    <endpoint binding="basicHttpBinding"
      contract="Ugovor.IFirmaServis"/>
    <endpoint address="rest" binding="webHttpBinding"
      contract="Ugovor.IFirmaServis"
      behaviorConfiguration="restBehaviour" />
  </service>
</services>
```

Ponašanje (eng. behavior)

- ❑ **Omogućava podešavanje rada sa sjednicom, istovremenim pristupom, transakcijama, ...**
- ❑ **Na razini servisa (Service behavior)**
 - npr. objavljivanje meta podataka, životni vijek servisa (po sjednici, po pozivu, jedan za sve pozive), ...
- ❑ **Na razini postupka (Operation behavior)**
 - npr. impersonacija, transakcije, životni vijek objekta
- ❑ **Na razini ugovora (Contract behavior)**
- ❑ **Na razini pristupne točke (Endpoint behavior)**
 - npr. upravljanje vjerodajnicama klijenta, način serijalizacije,...

Primjer upotrebe ponašanja pristupne točke

❑ Primjer: WCF\SmjestajWebApp\web.config

- Moguće je definirati izlazni format REST servisa (Json ili Xml)

```
<endpointBehaviors>
  <behavior name="restBehaviour">
    <webHttp defaultOutgoingResponseFormat="Json" />
  </behavior>
</endpointBehaviors>
```

- Bolja varijanta: odabrati automatski format na osnovu vrijednosti *Accept(Header)* u zaglavlju pozivajuće poruke


```
<endpointBehaviors>
  <behavior name="restBehaviour">
    <webHttp automaticFormatSelectionEnabled="true" />
  </behavior>
</endpointBehaviors>
```


Pružanje podataka o REST servisima

❑ Primjer: WCF \ SmjestajWebApp \ web.config

- Postavljanjem vrijednost *helpEnabled* na *true* u postavkama ponašanja servisa automatski se generira stranica s popisom postupaka servisa
- sufiks *"/help"* na adresi pristupne točke

```
<endpointBehaviors>  
  <behavior name="restBehaviour">  
    <webHttp helpEnabled="true"/>  
  </behavior>  
</endpointBehaviors>
```

→ ↻ 🏠  localhost:6093/Firma.svc/rest/help

Operations at http://localhost:6093/Firma.svc/rest

This page describes the service operations at this endpoint.

Uri	Method	Description
GetOsoba/{oib}	GET	Service at http://localhost:6093/Firma.svc/rest/GetOsoba/{OIB}
PopisDrzava	POST	Service at http://localhost:6093/Firma.svc/rest/PopisDrzava
PopisMjesta	POST	Service at http://localhost:6093/Firma.svc/rest/PopisMjesta
PopisOsoba	GET	Service at http://localhost:6093/Firma.svc/rest/PopisOsoba
PopisPartnera	POST	Service at http://localhost:6093/Firma.svc/rest/PopisPartnera
PostupakSPogreskom	POST	Service at http://localhost:6093/Firma.svc/rest/PostupakSPogreskom
UpdateOsoba	PUT	Service at http://localhost:6093/Firma.svc/rest/UpdateOsoba

Fiddler

❑ Alat za inspekciju http prometa i stvaranje http zahtjeva

- Za vrijeme dok je Fiddler aktivan pristupne točke pozivamo na `ipv4.fiddler` umjesto na `localhost`

The screenshot displays the Fiddler application window. The top menu bar includes Statistics, Inspectors, AutoResponder, Composer, Filters, Log, and Timeline. Below this is a sub-menu with Headers, TextView, WebForms, HexView, Auth, Cookies, Raw (selected), JSON, and XML. The main pane shows the details of a POST request to `http://127.0.0.1:6093/FirmaServis.svc`. The request headers include `Content-Type: text/xml; charset=utf-8`, `SOAPAction: "http://tempuri.org/IFirmaServis/PopisOsoba"`, `Host: 127.0.0.1:6093`, `Content-Length: 135`, `Expect: 100-continue`, `Accept-Encoding: gzip, deflate`, and `Connection: Keep-Alive`. The body is an XML SOAP envelope. Below the request, a search bar and a 'View in Notepad' button are visible. The bottom pane shows the response headers, including `HTTP/1.1 200 OK`, `Server: ASP.NET Development Server/10.0.0.0`, `Date: Wed, 04 Apr 2012 13:44:19 GMT`, `X-AspNet-Version: 4.0.30319`, `Content-Length: 12012`, `Cache-Control: private`, `Content-Type: text/xml; charset=utf-8`, and `Connection: Close`. The response body is also an XML SOAP envelope containing a `PopisOsobaResponse` with various fields like `AdresaPartnera`, `Ime`, `JMBG`, `Prezime`, and `AdresaIsporuke`.

Chrome Postman

❑ Postman – REST Client

- Proširenje za Chrome
- Omogućava jednostavno kreiranje različitih oblika zahtjeva (GET, PUT, POST, DELETE)

The screenshot displays the Chrome Postman extension interface. The browser address bar shows the URL `chrome-extension://fdmmgilgnpjigdojojpjoooidkmcomcm/index.html`. The Postman logo is visible in the top left corner. The interface is divided into several sections:

- Left Sidebar:** Contains the "POSTMAN" logo, download and help icons, and tabs for "History" and "Collections". The "History" tab is active, showing a list of recent requests. Two requests are visible, both labeled "GET" and pointing to `http://localhost:6093/Firma.svc/rest/PopisOsoba`.
- Main Panel:** Displays the details of the selected request.
 - Request Bar:** Shows the URL `http://localhost:6093/Firma.svc/rest/PopisOsoba` and the method `GET`. It also includes buttons for "URL params" and "Headers (1)".
 - Auth Section:** Includes tabs for "Normal", "Basic Auth", "Digest Auth", and "OAuth 1.0". The "Normal" tab is selected, and it shows "No environment".
 - Accept Section:** Shows the "Accept" header set to `application/json`. There is a "Manage presets" button.
 - Header Section:** A table with columns "Header" and "Value".
 - Action Buttons:** "Send", "Preview", "Add to collection", and "Reset".
 - Response Section:** Shows the "Body" tab selected. It displays the "STATUS" as `200 OK` and the "TIME" as `1650 ms`. Below this are tabs for "Pretty", "Raw", and "Preview". The "Pretty" tab is active, showing a JSON response in a formatted view. The response is an array with one object containing fields like `"AdresaIsporuke"`, `"AdresaPartnera"`, `"IdPartnera"`, `"OIB"`, `"Ime"`, and `"Prezime"`.

At the bottom of the interface, there is a banner for "Upgrade to Postman packaged app" and a "Supporters" link.

Atribut *WebInvoke*

- ❑ Koristi se za postupke koji se ne pozivaju načinom GET
- ❑ Primjer:  WCF\Ugovor\IFirmaServis.cs

```
[OperationContract]  
[WebInvoke(UriTemplate="UpdateOsoba/{idOsobe}",  
Method = "PUT", BodyStyle = WebMessageBodyStyle.Bare)]  
Osoba AzurirajOsobu(string idOsobe, Osoba o);
```

❑ Može se definirati

- *UriTemplate* - putanja do postupka i parametri koji se prenose kroz url
- *Method* - vrsta metode http zahtjeva (pretpostavljena je POST)
- *RequestFormat* i *ResponseFormat* – format ulaza i izlaza servisa (xml ili json)
- *BodyStyle* – određuje da li je podatak dodatno omotan ili ne

Primjer poziva REST servisa iz .NET klijenta (1)

❑ Primjer: WCF \ PrimjeriPoziva \ FormREST

- Razred *HttpClient* za pozivanje REST servisa
- Proširenja iz assembly-a *System.Net.Http.Formatting* za automatsku deserijalizaciju iz xml-a i json-a u željeni razred i obrnuto
 - *.ReadAsAsync<T>*

```
private async void btnLoad_Click...    {
    string serviceUrl = tbServiceUrl.Text + "/PopisOsoba";
    HttpClient client = new HttpClient();
    client.DefaultRequestHeaders.Accept.Clear();
    ...
    client.DefaultRequestHeaders.Accept.Add(new
        MediaTypeWithQualityHeaderValue("application/json"));

    var response = await client.GetAsync(serviceUrl);
    var osobe = await response.Content
        .ReadAsAsync<IEnumerable<Ugovor.Osoba>>();
    osobaBindingSource.DataSource = osobe;
    ...
}
```

Primjer poziva REST servisa iz .NET klijenta (2)

❑ Primjer: WCF \ PrimjeriPoziva \ FormREST

- Asinkroni postupci razreda *HttpClient* za 4 vrste HTTP zahtjeva
 - *GetAsync*, *PostAsync*, *PutAsync*, *DeleteAsync*
- Mogućnost automatske serijalizacije u *xml* ili *json* kod PUT i POST zahtjeva
 - *PostAsJsonAsync*, *PostAsXmlAsync*
 - *PutAsJsonAsync*, *PutAsXmlAsync*

```
private async void btnUpdate_Click... {  
    var osoba = osobaBindingSource.Current as Ugovor.Osoba;  
    string serviceUrl = tbServiceUrl.Text + "/UpdateOsoba";  
    HttpClient client = new HttpClient();  
    HttpResponseMessage response = await  
        client.PutAsJsonAsync(serviceUrl, osoba);  
    ...  
}
```

WebApi

❑ Okvir za razvoj REST servisa

- Struktura projekta nalik ASP.Net MVC-u (usmjeravanje, upravljači)
- Primjer:  WCF \ WebApi \ App_Start \ WebApiConfig.cs

```
config.Routes.MapHttpRoute(  
    name: "DefaultApi",  
    routeTemplate: "api/{controller}/{id}",  
    defaults: new { id = RouteParameter.Optional }  
);
```

- Nasljeđuje se razred ApiController
- Primjer:  WCF \ WebApi \ Controllers \ OsobaController.cs

```
public class OsobaController : ApiController  
    //GET api/<controller>  
    public IEnumerable<WebApi.Models.Osoba> Get()  
    public IHttpActionResult Get(int id)  
    public void Post(...  
    public void Put(int id, [FromBody]WebApi.Models.Osoba o)  
    public void Delete(int id)
```

Prednosti WCF-a i WebAPI-a

❑ WCF:

- Podržava više protokola, a ne samo HTTP
- Mogućnost definiranja vlastitog protokola
- Podržava više formata (tekst, binarni sadržaj)
- Napredniji sigurnosni mehanizmi
- Automatsko generiranje klijenta na osnovu WSDL-a

❑ WebAPI:

- Prikladan za sve preglednike i mobilne uređaje
- Veća podrška za REST
- Naprednije mogućnosti formiranja URL-a
- Jednostavnija izrada

Reference

- WCF
 - <http://msdn.microsoft.com/en-us/netframework/aa663324>
- JSON
 - <http://www.json.org>
 - <http://en.wikipedia.org/wiki/JSON>
- REST
 - http://en.wikipedia.org/wiki/Representational_state_transfer
- Postman
 - <http://www.getpostman.com/>
- Fiddler
 - <http://www.fiddler2.com>
- Popis specifikacija za web servise
 - http://en.wikipedia.org/wiki/List_of_Web_service_specifications
- MVC i REST: ASP.NET Web API
 - <http://www.asp.net/web-api>
- WCF and WebApi
 - <http://msdn.microsoft.com/en-us/library/jj823172.aspx>