Osnove WCFa

Uvod

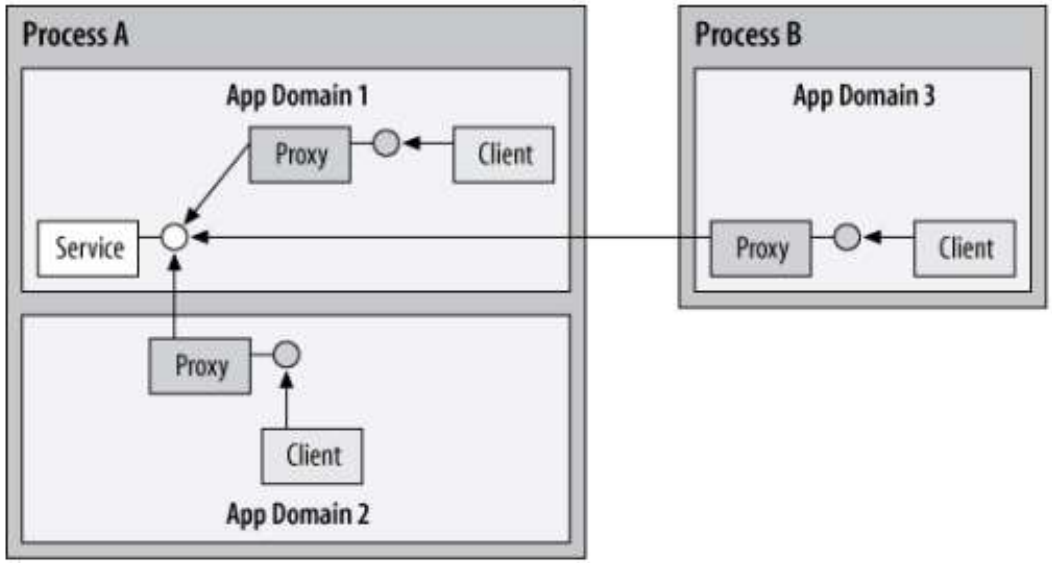
- **Windows Communication Foundation (WCF)**

- Omogućava potpun pristup distribuiranom programiranju
- Efikasan način da proizvedemo i upotrebljavamo servise.

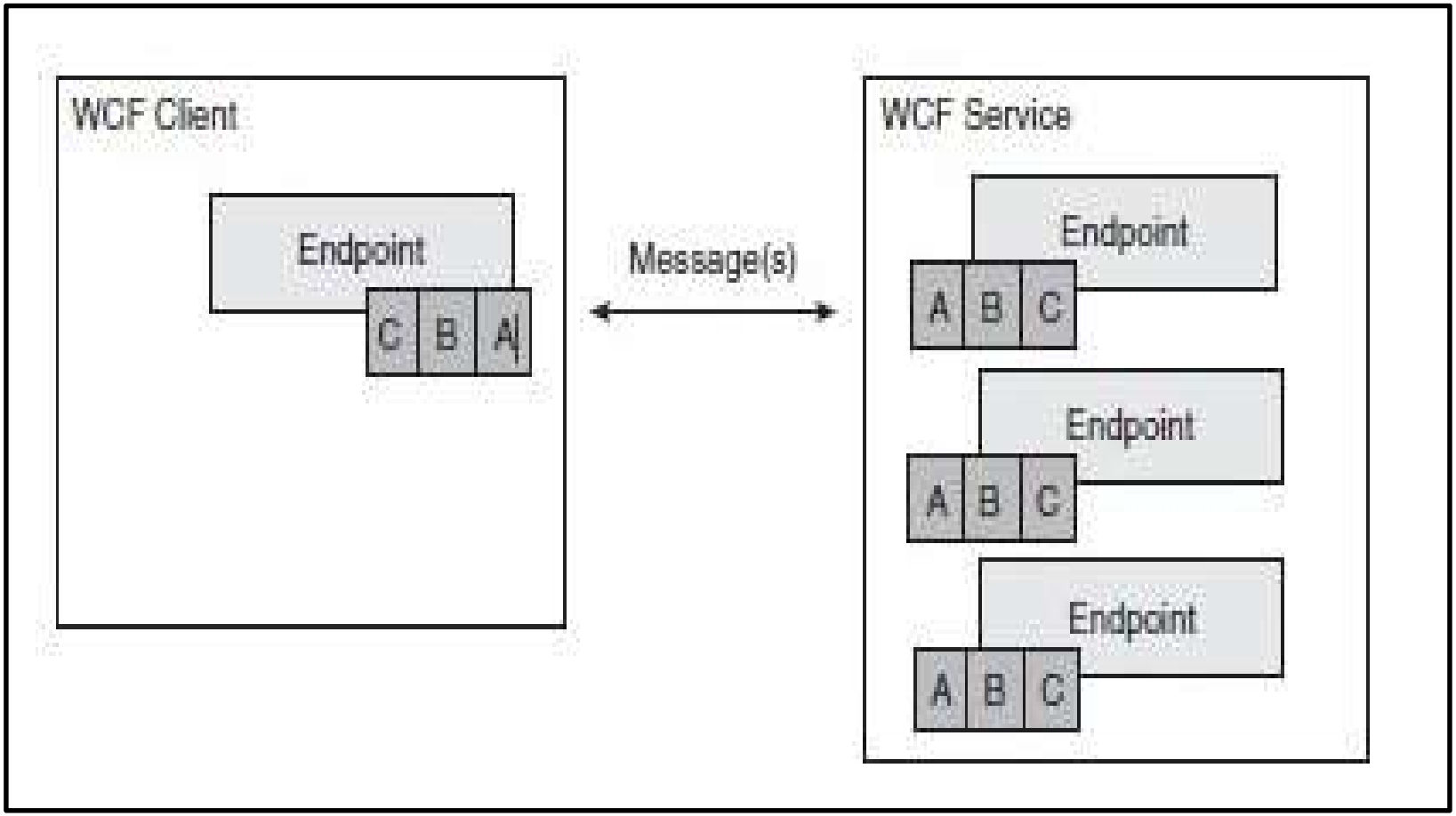
Komunikacija preko posrednika

WCF podržava sledeće transportne šeme:

- HTTP/HTTPS
- TCP
- IPC
- Peer network
- MSMQ
- Service bus



Krajnja tačka (**endpoint**)



- **Adresa** pristupne tačke na kojoj servis sluša dolazne upite. Predstavlja se u standardnom **Url** formatu.
- **Povezivanje (binding).** Definiše osobine komunikacionog kanala kojim se pristupa servisu. Kanal može da se sastoji od niza povezanih elemenata. Najniži element tog komunikacionog kanala je transportni protokol, koji definiše način na koji se poruke razmenjuju između klijenta i servisa. Standardni format koji se koristi za opis zahteva koje servis postavlja za povezivanja je WS-Policy.
- **Ugovor (contract).** Definiše funkcionalnosti koje pristupna tačka pruža i format poruka koje implementirane funkcije očekuju. Za opis ugovora koristi se stanardni format (eng. Web Service Description Language - **WSDL**)

Vrste operacija

- Standardne operacije za razmenu poruka su:
  - Poruke u jednom smeru – engl. one-way,
  - Poruke koje vraćaju podatak – engl. request-reply i
  - Istovremena razmena poruka u oba pravca – engl. duplex.

Kontejner servisa

- Da bi servis mogao da radi tj. da osluškuje zahteve odnosno pozive operacija, mora postojati kontejner za hostovanje. Kod standardnih WS to je web server, na primer IIS server.

- Kontejner za WCF **može biti bilo koji proces koji se izvršava**.

WSDL

- Klijentska aplikacija pristupa servisnoj krajnjoj tački kako bi prikupila ABC-du servisa preko *Web Service Description Language (WSDL)*. Servis dodaje i opis podataka: *Metadata Exchange (MEX)*.
- Klijent pošto prihvati WSDL kreira *proxy* klasu i *app.config*.
- Proxy klasa preslikava potpise operacija krajnje tačke kako bi kod na klijentskoj strani mogao da pozove na odgovarajući način operacije na udaljenoj krajnjoj tački. Tačnije, proxy mora da obezbedi da poruke koje se šalju servisu budu po tačno zadatoj spesifikaciji.

Primer

- Servis za konverziju stepeni:
- WSDL na adresi:
- https://www.w3schools.com/xml/tempconvert.asmx?WSDL
- http://www.webservicex.net/geoipservice.asmx?WSDL
- http://www.webservicex.net/CurrencyConvertor.asmx
- http://www.webservicex.net/globalweather.asmx?WSDL
- http://secure.smartbearsoftware.com/samples/testcomplete14/webservices/Service.asmx
- http://www.dneonline.com/calculator.asmx

- Za registrovane korisnike (pravna lica): https://webservices.nbs.rs/CommunicationOfficeService1_0/CurrentExchangeRateService.asmx?WSDL

Primer

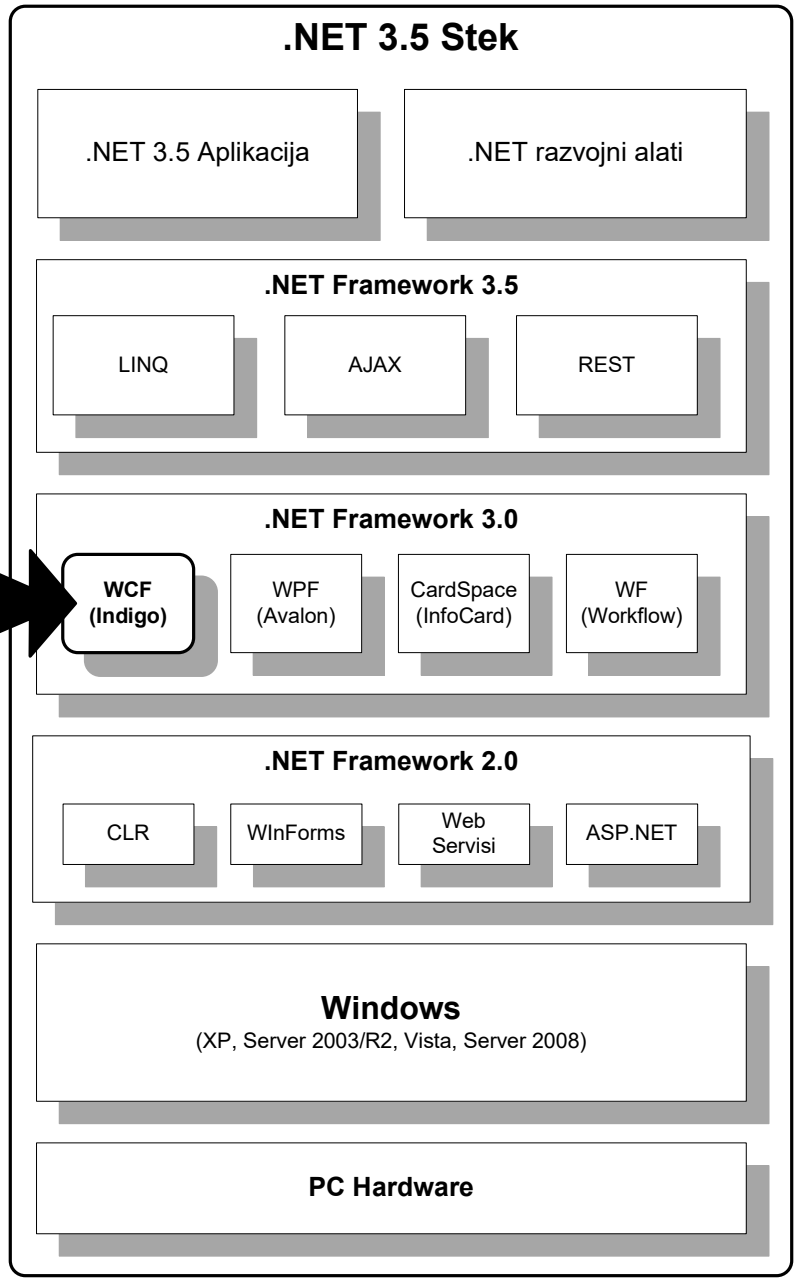
- Kreiranje veze do postojećeg servisa

Arhitektura servisa

- WCF je jedan od četiri API-ja koji su predstavljeni u .NET Framework-u 3.0, a kasnije nadograđena u .NET Framework-u 3.5 da podrži mogućnosti kao što su JSON, REST i intergracija sa WWF.

- WCF nudi jedinstven programski model za razvoj servis orjentisanih, konektovanih aplikacija. Uspeo je da objedini razne različite modele koji su postojali u .NET Framwork-u 2.0; Web Servis API za razvoj SOAP bazirane komunikacija, binarnu komunikaciju za komunikaciju između Windows mašina (.NET Remoting), transakcione komunikacija (Distributed Transactions) i model asinhrone komunikacije (Message Queues).

- Na slici vidimo strukturu .NET 3.5 Framework steka.
- Prepoznaju se slojevi sa novim mogućnostima koje su inkremetalno dodavane .NET Framework-u.

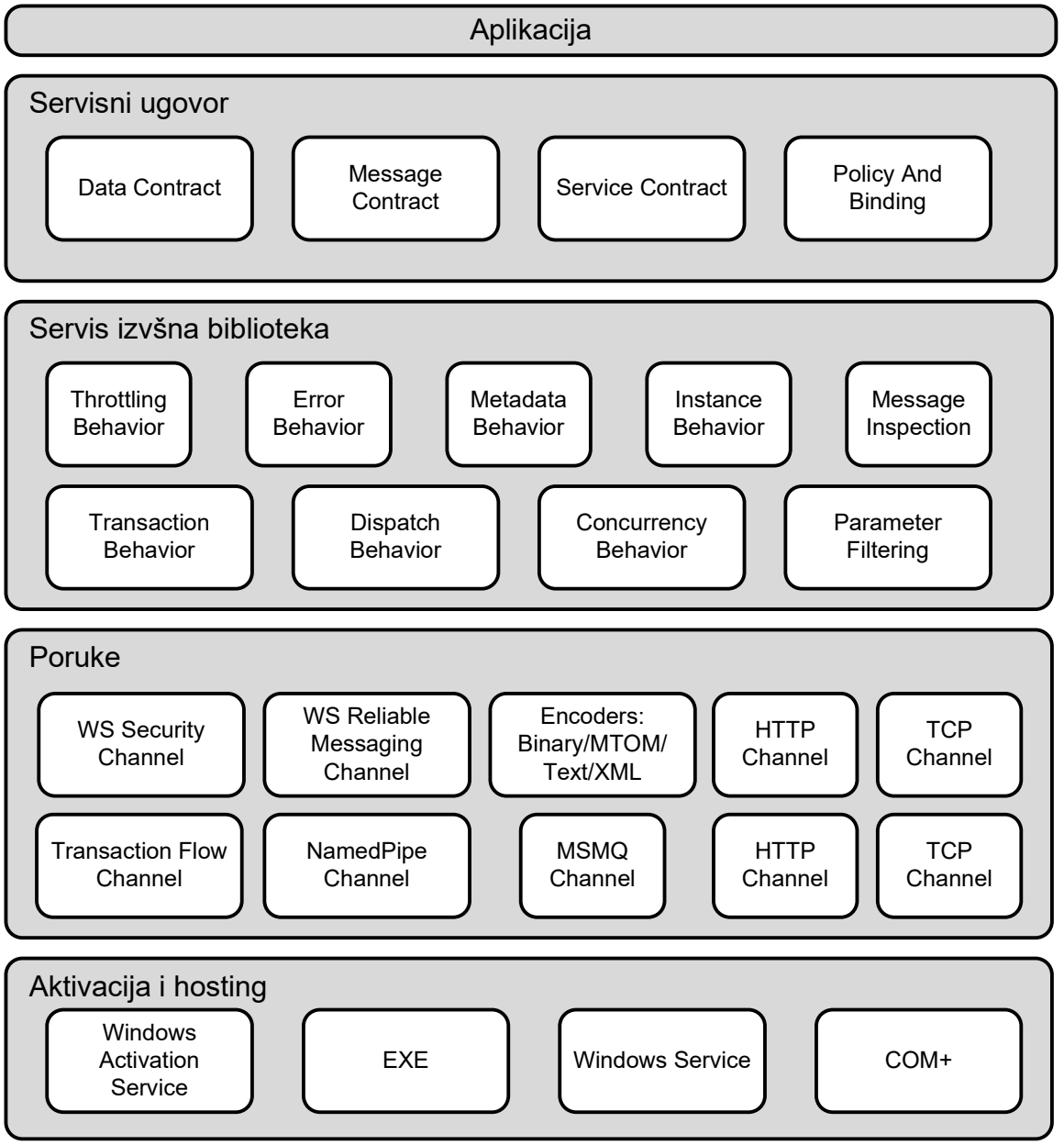


Prednosti

- Prednosti koje WCF donosi razvojnom timu mogu se grupisati na sledeći način:
- *Produktivnost*
  - Jedinstven programski model
  - Razvoj baziran na atributima
  - Integracija sa Visual Studiom
- *Interoperabilnost*
  - Podrška za WS-* specifikaciju
  - Kompatibilnost sa svim postojećim MS tehnologijama za razvoj distribuiranih sistema
- *Servisno orjentisani razvoj*
  - Omogućava razvoj labavo povezanih servisa
  - Komunikacioni kanali bazirani na konfiguracijama

S obzirom na broj tehnologija koje je uspeo da objedini i pomak u produktivnosti kroz konfigurabilnost izvršne biblioteke, ne čudi da je strukturni dijagram arhitekture samog WCF nešto komplikovaniji.

Opisi servisa i servisni ugovori (Contracts and descriptions) – zadužen sve podatke koji su potrebni da se definišu da bi se servis mogao koristiti od strane klijenta. Definicije operacija, parametara, poruka koje servis šalje i prima, komunikacioni kanali, autentifikacija, itd...



Servisna izvšna biblioteka (Service runtime) - zadužena za režime rada servisa u toku samog procesiranja zahteva klijenata. Na osnovu konfigurisanog režima rada ovaj sloj može da promeni način instanciranja servisa, paralelizam, transakcije, dodatnu kontrolu i obradu poruka koje stižu i odlaze od servisa, itd...

Razmena poruka (Messaging) – sastoji se od komunikacionih kanala i zadužen je za razmenu poruka sa klijentom. Izdvajaju se dve osnovne grupe kanala, transportni kanali i protokoli. Transportni kanali su zaduženi za čitanje i pisanje poruka na komunikacioni medijum, korišćenjem nekog od enkodera, koji su takođe deo ovog sloja WCF. Protokoli su zaduženi za procesiranje pristiglih poruka, i upravljanje višeg nivoa (uobičajeno čitajući zaglavlja poruka). Primeri protokola su WS-Security, WS-Realiable Messaging i WS-Transactions.

Aktivacija i hosting – zadužen za kontekst u kome se servis izvršava i sluša dolazne poruke klijenata. To može biti izvšni fajl, Windows servis, IIS web server ili WAS.

Ulogu ovih slojeva i njihovu međusobnu zavisnost i interakciju razmotrićemo u daljem tekstu.

Programski model

- Da bi implementirali WCF servis, potrebno je prvo da eksplicitno definišemo njegov **intefejs** sa svim operacijama koje želimo da budu vidljive klijentima.

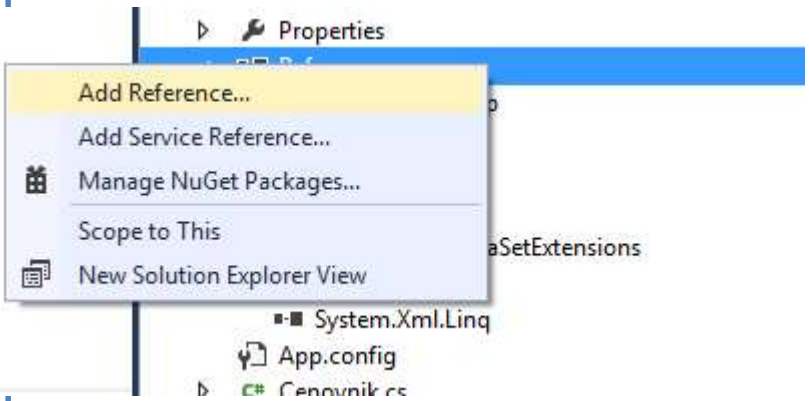
Primer interfejsa

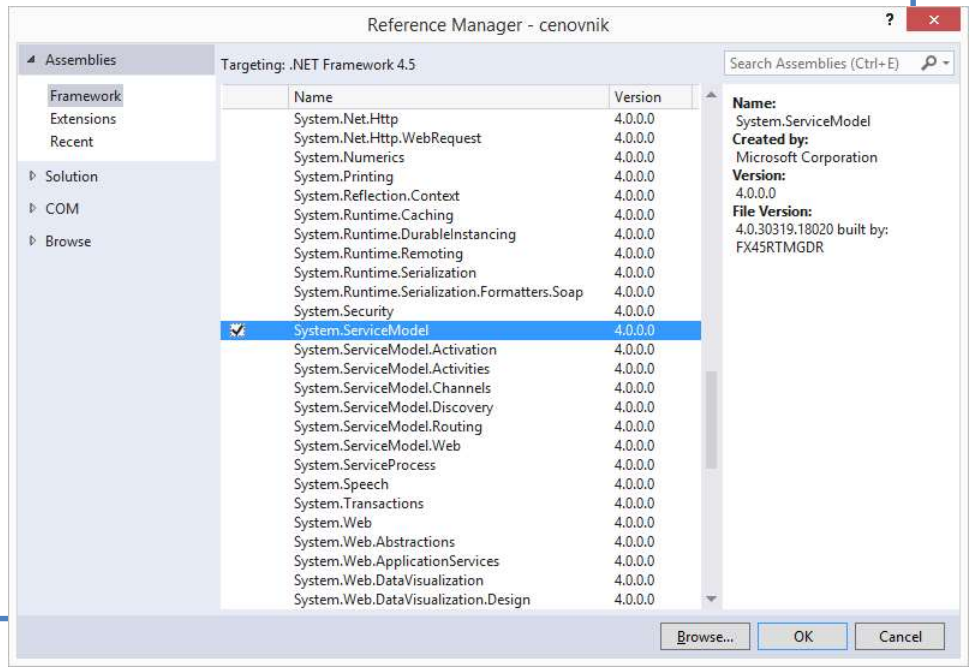
```
using System.ServiceModel;

//a WCF contract defined using an interface
[ServiceContract]
public interface IMath
{
    [OperationContract]
    int Add(int x, int y);
}
```

Dodavanje reference - 1

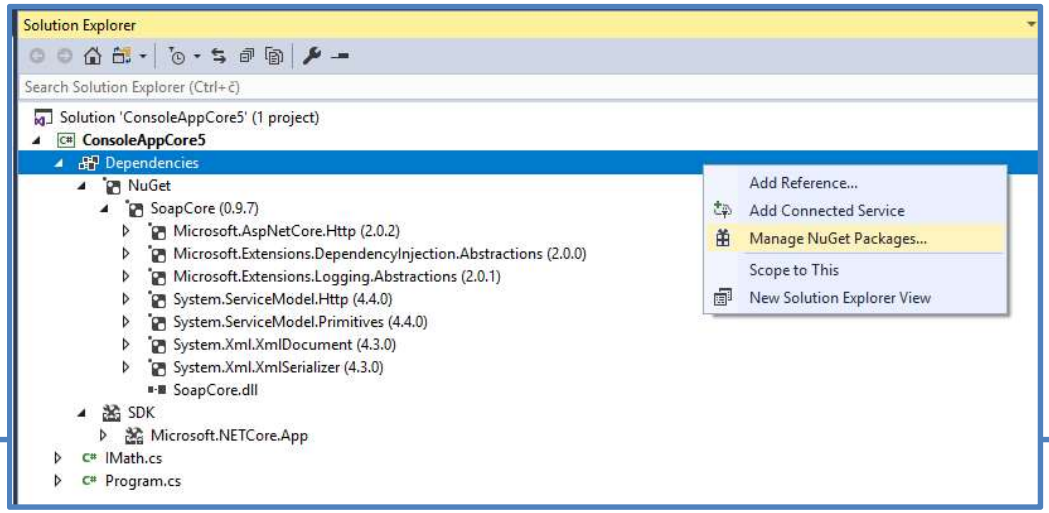
- Ukoliko tip projekta koji je napravljen nije za WCF servise (na primer Console Application) potrebno je uključiti dodatne biblioteke. Na slici je prikazano dodavanje System.ServiceModel biblioteke postojećem projektu
- using System.ServiceModel;





Dodavanje reference - 2

- Ako se referenca dodaje Core (univerzalni projekti za sve platforme) projektima onda se takvim projektima ako ne sadrže potrebnu biblioteku dodaje biblioteka SoapCore koristeći NuGet okruženje za dodavanje paketa postojećem projektu.



- Moguće je ubaciti gotov servisni ugovor ili napraviti novi nezavisno od razvojnoj okruženja i bez upotrebe atributa na interfejsu, ali se takva praksa ne preporučuje.
- Razlog? Svaki servisni ugovor definisan pomoću WSDL-a je validan, pa se manualnim ubacivanjem ovih fajlova može da dođe do nekozistentnosti servisnog ugovora i samog servisa.

- Korišćenje atributa za kreiranje servisnog ugovora osigurava da IDE proverava validnost servisnog ugovora na osnovu naše implementacije.

- Sam servis se implementira kao klasa koja nasleđuje interfejs servisa (i ugovor koji se iz interfejsa generiše). Na slici vidimo *MathService* koji nasleđuje *Imath* intefejs i implementira operaciju sabiranja koja je definisana u interfejsu.

Implementacija interfejsa

```
//the service class implements the interface
public class MathService : IMath
{
    public int Add(int x, int y)
    { return x + y; }
}
```

Hostovanje

- Ovako implementiran servis, da bi bio dostupan klijentu mora biti hostovan u nekom izvršnom fajlu.
- Servis može biti
  1. **samo-hostovan**, u svom izvšnom fajlu, ili u nekom drugom procesu koji je kontrolisan od stane nekih eksternih agenata, kao na primer
  2. **IIS** ili
  3. Windows Authentication Service (**WAS**).
  4. Izvršni fajl servisa može biti pokrenut i automatki kao **Windows servis**.
- Za ovako hostovan servis preostaje samo da se difiniše jedna ili više servih pristupnih tačaka sa svim informacijama potrebnim klijentu da bi komunicirao sa servisom.

Primer 2

```
// int. koji definise ponasanje s.
[ServiceContract]
interface ICenovnik
{
    [OperationContract]
    string dajCenu(string vrsta);
}
```

```
// klasa koja definise servis
class Cenovnik : ICenovnik
{
   public string dajCenu(string vrsta)
    {
          return vrsta + ": 55din";
     }
}
```

Pokretanje WCF servisa

```
Uri adresa = new Uri("http://localhost:8000/Cenovnik");
 ServiceHost serviceHost = new ServiceHost( typeof(Cenovnik),adresa);
 serviceHost.AddServiceEndpoint(typeof(ICenovnik), new BasicHttpBinding(),"");
 serviceHost.Open();
 Console.WriteLine("Press <Enter> to terminate.\n\n");
 Console.ReadLine();
  serviceHost.Close();
```

Pokretanje klijenta

```
ChannelFactory<ICenovnik> cf = new ChannelFactory<ICenovnik>
(new BasicHttpBinding(), new EndpointAddress("http://localhost:8000/Cenovnik"));
        ICenovnik o = cf.CreateChannel();
        string p = o.dajCenu("kafa");
        Console.WriteLine("Dobijeno: {0} \n\n", p);
        cf.Close();
```

```
namespace WCFclient
{
   // interface koji definise ponasanje servisa
   [ServiceContract]
   interface ICenovnik
   {
      [OperationContract]
      string dajCenu(string vrsta);
   }
}
```