Vežba 1 - Osnove WCF-a

Windows Communication Foundation (WCF) je framework koji služi za pravljenje servisno-orijentisanih aplikacija. Uz pomoć WCF-a imamo mogućnost slanja asinhronih poruka sa jednog na drugi servis.

Svaki WCF servis definiše minimalno jedan endpoint koji sadrži skup neophodnih informacija koje su potrebne WCF klijentu za pristup servisu.

Svaki endpoint mora da sadrži sledeće informacije

A. Informacije o adresi na kojoj servis sluša (***address***) koja je predstavljena u standardnom URI formatu na sledeći način:
*Scheme***://***<MachineName>[:Port]/Path*, gde je:

- **Scheme**: transportni protokol (TCP, HTTP, itd.),
- **Machine nam**e: ime ili adresa računara nam kom je pokrenut (u WCF kontekstu se koristi još i izraz „hostovan“) servis,
- **Port**: ovaj parametar je opcioni, i ukoliko se ne navede koristi se podrazumevani broj porta za odgovarajući protokol, npr. 443 za HTTPS.
- **Path**: putanja do servisa.

B. O komunikacionom protokolu (***binding***),

C. Informacije o opisu usluge (***contract***) koje servis pruža klijentima.

Slika 1. Osnovni elementi WCF Endpoint -a

Konfiguracija i pokretanje servisa

```
[ServiceContract]
public interface IExample
{
    [OperationContract]
    string TestFunction();
}
```

Kako bismo pokrenuli neki servis prethodno moramo definisati interfejs koji ćemo koristiti kao Contract. Da bi se jedan interfejs koristio kao contract neophodno ga je dekorisati

atributom *ServiceContract*, a metode koje želimo da izlaže sa *OperationContract* atributom (nalaze se u *System.ServiceModel* biblioteci). Primer interfeja se nalazi iznad.

Rad se složenim podatacima

Pored prenošenja ugrađenih tipova WCF nudi mogućnost slanja korisnički definisanih tipova podataka. U WCF-u se koristi *DataContract* serijalizacija prilikom prenosa podataka. Svi ugrađeni .NET primitivni tipovi, kao i neki ugrađeni izvedeni tipovi mogu biti serijalizovani i deserijalizovani pomoću *DataContract* serijalizatora.

Korisnički-definisane tipove podataka je potrebno označiti (dekorisati) sledećim atributima (nalaze se u *System.Runtime.Serialization* prostoru imena iz):

1. *DataContract*: eksplicitno definiše novi tip podatka za serijalizaciju, koristi se za dekorisanje klasa, struktura i enumeracija;
2. *DataMember*: definiše članove klase koji će biti uključeni u proces serijalizacije;
3. *IgnoreDataMember*: navodi se kako bi se preskočila serializacija određenog člana klase.

```
[DataContract]
public class Knjiga
{
    public Knjiga(string nazivKnjige, string imeAutora, string prezimeAutora)
    {
        this.NazivKnjige = nazivKnjige;
        this.ImeAutora = imeAutora;
        this.PrezimeAutora = prezimeAutora;
    }
    [DataMember]
    public string NazivKnjige{ get => nazivKnjige; set => nazivKnjige= value; }
    [DataMember]
    public string ImeAutora{ get => imeAutora; set => imeAutora= value; }
    [DataMember]
    public string PrezimeAutora{ get => prezimeAutora; set => prezimeAutora= value; }

    private string imeAutora;
    private string imeAutora;
    private string prezimeAutora;
}
```

Rad sa WCF izuzecima

WCF podržava propagaciju grešaka od servisa do klijenata preko izuzetaka, tj. omogućava da se izuzetak kreira na strani servisa, propagira kroz komunikacioni kanal do klijenta, gde ga klijent može (i treba) uhvatiti sa odgovarajućim kodom za obradu grešaka.

Navođenje izuzetka koji WCF metod može da baci je moguće navođenjem atributa *FaultContract* iznad metode koja može da baci izuzetak

```
        [OperationContract]
        [FaultContract(typeof(Exception))]
           string TestFunction();
```

Konfiguracija WCF aplikacija

Platforma .NET podržava konfigurisanje izvršnih aplikacija pomoću konfiguracione datoteke koja sadrži konfiguraciju definisanu na XML jeziku. Prevođenjem projekta se pravi kopija ove datoteke sa promenjenim imenom ***ime_aplikacije.exe.config***. Konfiguraciju .NET aplikacija je moguće menjati u tim datotekama. Ovakve datoteke se dodaju u projekte koji su izvršnog tipa. Za rad sa konfiguracionim datotekama potrebno je uključiti **System.Configuration** namespace.

U donjem listingu je prikazano na koji način se konfiguriše server.

```
        ServiceHost svc = new ServiceHost(typeof(ServiceName));
```

```
        <?xml version="1.0" encoding="utf-8" ?>
        <configuration>
            <startup>
                <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7"/>
            </startup>
          <system.serviceModel>
            <services>
              <service name="NamespaceOfService.ServiceName">
                <host>
                  <baseAddresses>
                    <add baseAddress="net.tcp://localhost:4000" />
                  </baseAddresses>
                </host>
                <!-- Service Endpoints -->
                <endpoint address="Service" binding="netTcpBinding"
                contract="NamespaceOdInterface.InterfaceName" /> </service>
              </services>
          </system.serviceModel>
        </configuration>
```

U donjem listingu je prikazano na koji način se konfiguriše klijent.

```
        ChannelFactory<InterfaceName> factory = new ChannelFactory<InterfaceName>("NazivServisa");
```

```
        <?xml version="1.0" encoding="utf-8" ?>
        <configuration>
            <startup>
                <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5"
            /> </startup>
          <system.serviceModel>
            <client>
              <endpoint name="NazivServisa"
                address="net.tcp://localhost:4000/Service"
                binding="netTcpBinding"
                contract="NamespaceOdInterface.InterfaceName" />
            </client>
          </system.serviceModel>
        </configuration>
```

Zadatak

Neophodno je implementirati klijent-server arhitekturu koristeći konfiguracioni fajl prilikom konfigurisanja servisa i klijenta

Servis treba da izloži sledeće metode :

- Dodavanje novih knjiga u biblioteku
- Brisanje postojećih knjiga iz biblioteke, metoda treba da baci CustomException izuzetak ukoliko pokuša da se obriš knjiga koja ne postoji
- Izlistavanje svih knjiga koje postoje u biblioteci
- Izlistavanje svih knjiga po nazivu autora (ime + prezime)
- Izlistavanje svih knjiga na osnovu zanra
- Izlistavanje svih knjiga koje su izdate odredjene godine

Treba implementirati klase Knjiga i CustomException

Klasa Knjiga treba da sadrži :

- ID knjige (predstavlja redni broj kreiranog objekta klase)
- Ime knjige
- Ime autora
- Prezime
- Zanr (predstavljen enumeracijom)
- Datum izdavanja
- Za svako odgovarajuce polje obezbediti Property
- Za inicijalizaciju objekta NE koristiti default-ni konstruktor

Klasa CustomException treba da sadrzi

- Message i odgovarajuci Property

Metode interfejsa implementirati na strani Servisa, sa strane servisa neophodno je da postoji kolekcija Knjiga