Serijalizacija

Serijalizacija predstavlja proces prevođenja objekata u sekvencu bita, kako bi se stanje objekta moglo sačuvati u memorijskom baferu, datoteci ili preneti preko mreže. Obrnuti proces - ekstrakcije objekta iz sekvence bitova naziva se **deserijalizacija**. Za namenu razmene informacija putem standardizovanih formata poruka može se koristiti *DataContractSerializer* .Net Framework-a.

DataContractSerializer

Klase koje je potrebno serijalizovati se obeležavaju *DataContract* atributom. Na polja klase koja će biti uključena u serijalizaciju se stavlja atribut *DataMember*. Da bi se mogao koristiti DataContractSerializer, potrebno je dodati u projektu referencu na System.Runtime.Serialization.dll i uključiti namespace System.Runtime.Serialization.

```
[DataContract]
public class Person
{
        [DataMember] public string Name;
        [DataMember] public int Age;
}
```

DataMember atribut može se koristiti za polja (eng. *Fields*) i svojstva (eng. *Properties*) sa svim pravima pristupa (*public*, *protected*, *private*). Podržani su svi ugrađeni tipovi, enumeracije, kolekcije i svi izvedni tipovi koji su deklarisani sa *DataContract* i *Serializable* atributima.

```
static void Main(string[] args)
 {
       MemoryStream stream = new MemoryStream();

       DataContractSerializer serializer =
             new DataContractSerializer(typeof(Person));

       Person p1 = new Person { Name = "Anita", Age = 18 };

       //Serialization
        serializer.WriteObject(stream, p1);
        stream.Position = 0;

        //Deserialization
       Person p2 = serializer.ReadObject(stream) as Person;
  }
```

U nastavku je dat rezultat serijalizacija iz prethodnog primera.

```
<Person ...>
   <Age>18</Age>
```

```
   <Name>Anita</Name>
</Person>
```

Serijalizacija izvedenih klasa

Da bi se omogućila serijalizacija izvedenih klasa potrebno je u atributu bazne klase navesti tipove izvedenih klasa za koje se odobrava serijalizacija.

```
[DataContract, KnownType (typeof (Student)), KnownType (typeof (Teacher))]
public class Person
{
	...
}

[DataContract]
public class Student : Person
{
	[DataMember]
	public string Index;
}
```

Reference objekata

Polja klase mogu biti izvedeni tipovi (npr. klase), ukoliko su u deklaraciji obeleženi *DataContract* atributom.

```
[DataContract]
public class Person
{
	...

	[DataMember]
	public Address HomeAddress;
}

[DataContract]
public class Address
{
	[DataMember]
	public string Street, Postcode;
}
```

Redosled serijalizacije

Moguće je uticati na redosled serijalizovanja polja sa atributom *Order.* Prvo se serijalizuju polja bazne klase, pa tek onda polja specifična za izvedenu klasu.

```
[DataContract] public class Person
{
	[DataMember (Order=0)] public string Name;
```

```
        [DataMember (Order=1)] public int Age;
}
```

Esencijalna polja

Polja koja su od esencijalnog zančaja za tip, deklarišu se sa atributom *IsRequired*.

```
[DataMember(IsRequired=true)] public int Name;
```

Serijalizacija kolekcija

Polja klasa mogu biti kolekcije. U nastavku je dat primer serijalizacije klase čije polje je lista izvedenog tipa *Address*.

```
[DataContract] public class Person
{
  ...
        [DataMember] public List<Address> Addresses;
}

[DataContract] public class Address
{
        [DataMember] public string Street, Postcode;
}
```

Odgovarajući rezultat serijalizacije:

```
<Person>
   ...
   <Addresses>

      <Address>
         <Postcode>21000</Postcode>
         <Street>Jovan Jovanovic Zmaj </Street>
      </Address>

     <Address>
         <Postcode>11000</Postcode>
         <Street>Knez Mihajlova</Street>
      </Address>

   </Addresses>
...
</Person>
```

Niz objekata koji pripadaju istoj baznoj klasi moguće je serijalizovati u jedan memorijski stream ukoliko se objekti prethodno dodaju u kolekciju pa se serijalizuje sama kolekcija.

```
Person p1 = new Person { Name = "Anita", Age = 30 };
Student s1 = new Student { Name = "Dusan", Age = 23, Index = "E21717" };

// Serialize
serializer.WriteObject(stream, new List<Person>(2) { p1, s1});

// Deserialize
List<Person> persons = (serializer.ReadObject(stream)) as List<Person>;
```