Serialization, bir nesnenin durumunu (state) bir dizi bite veya başka bir taşınabilir formata dönüştürme işlemidir. Bu, nesnenin daha sonra aynı veya farklı bir ortamda yeniden oluşturulabilmesi için yapılır. Serialization, nesneleri dosyalar, veri tabanları veya ağ üzerinden taşımak gibi senaryolar için kullanılır. Deserialization ise, bu seri hale getirilmiş verilerden orijinal nesneyi yeniden oluşturma işlemidir.

### Neden Serialization Kullanılır?

1. \*\*Kalıcı Depolama\*\*: Nesneleri dosyalara veya veri tabanlarına kaydetmek.

2. \*\*Veri İletişimi\*\*: Nesneleri ağ üzerinden başka bir uygulamaya veya sisteme göndermek.

3. \*\*Kopyalama\*\*: Nesnenin bir kopyasını oluşturmak için.

### .NET'te Serialization

.NET Framework'te serialization için birden fazla yol vardır. En yaygın kullanılan yöntemler Binary, XML ve JSON serialization'dır.

#### Binary Serialization

Binary serialization, nesneleri ikili (binary) formatta serileştirir. Bu, hızlı ve verimli bir yoldur ancak insan tarafından okunabilir değildir. Aşağıda bir örnek verilmiştir:

```csharp

using System;

using System.IO;

using System.Runtime.Serialization.Formatters.Binary;

[Serializable]

public class Person

{

public string Name { get; set; }

public int Age { get; set; }

}

class Program

{

static void Main()

{

Person person = new Person { Name = "John Doe", Age = 30 };

// Serialize

using (FileStream fs = new FileStream("person.dat", FileMode.Create))

{

BinaryFormatter formatter = new BinaryFormatter();

formatter.Serialize(fs, person);

}

// Deserialize

using (FileStream fs = new FileStream("person.dat", FileMode.Open))

{

BinaryFormatter formatter = new BinaryFormatter();

Person deserializedPerson = (Person)formatter.Deserialize(fs);

Console.WriteLine($"Name: {deserializedPerson.Name}, Age: {deserializedPerson.Age}");

}

}

}

```

#### XML Serialization

XML serialization, nesneleri XML formatında serileştirir. Bu, insan tarafından okunabilir ve yaygın olarak veri değişim formatı olarak kullanılır.

```csharp

using System;

using System.IO;

using System.Xml.Serialization;

[Serializable]

public class Person

{

public string Name { get; set; }

public int Age { get; set; }

}

class Program

{

static void Main()

{

Person person = new Person { Name = "John Doe", Age = 30 };

// Serialize

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

using (FileStream fs = new FileStream("person.xml", FileMode.Create))

{

serializer.Serialize(fs, person);

}

// Deserialize

using (FileStream fs = new FileStream("person.xml", FileMode.Open))

{

Person deserializedPerson = (Person)serializer.Deserialize(fs);

Console.WriteLine($"Name: {deserializedPerson.Name}, Age: {deserializedPerson.Age}");

}

}

}

```

#### JSON Serialization

JSON serialization, nesneleri JSON formatında serileştirir. Bu, insan tarafından okunabilir ve modern web uygulamaları için yaygın olarak kullanılır.

```csharp

using System;

using System.IO;

using System.Text.Json;

public class Person

{

public string Name { get; set; }

public int Age { get; set; }

}

class Program

{

static void Main()

{

Person person = new Person { Name = "John Doe", Age = 30 };

// Serialize

string jsonString = JsonSerializer.Serialize(person);

File.WriteAllText("person.json", jsonString);

// Deserialize

string jsonStringFromFile = File.ReadAllText("person.json");

Person deserializedPerson = JsonSerializer.Deserialize<Person>(jsonStringFromFile);

Console.WriteLine($"Name: {deserializedPerson.Name}, Age: {deserializedPerson.Age}");

}

}

```

### Özet

Serialization, nesnelerin durumunu kalıcı olarak saklamak veya başka sistemlere iletmek için kullanılan bir tekniktir. Binary, XML ve JSON gibi çeşitli formatlarda gerçekleştirilir. Deserialization ise, bu formatlardan orijinal nesne durumunu geri yükleme işlemidir. Serialization, veri taşınabilirliğini ve uygulamalar arası entegrasyonu kolaylaştırır.