Serialización

Serialización en C#

Uno de los problemas más frecuentes durante la comunicación en programación web es conseguir transferir objetos hacia y desde una aplicación. Para eso, el objeto se debe transformar en un formato universal. Esta transformación se llama serialización. Este proceso permite recuperar el objeto representado bajo un formato intercambiable; con frecuencia este formato tiene una forma textual.

Para que los datos puedan transitar, hay que usar un flujo de datos (llamado stream en inglés, de ahí el nombre de la clase base en C#: Stream). Estos flujos pueden tomar varias formas, como por ejemplo un flujo de datos en memoria (representado por la clase MemoryStream en C#) o incluso un flujo de datos hacia un archivo en el disco (representado por la clase FileStream en C#).

Hay varias maneras de serializar un objeto; estas son algunas de ellas:

- La serialización binaria, que permite representar un objeto en un formato binario.

- La serialización XML, que transforma el objeto en cadena de caracteres al formato XML.

- La serialización JSON, que transforma el objeto en cadena de caracteres al formato JSON.

En este capítulo vamos a abordar tres modos de serialización para transformar un objeto a un formato dado, pero también para poder recuperar un objeto desde una fuente de datos en el formato retenido.

Serialización binaria

El enfoque binario es el modo de serialización más simple para implantar. También es el que permite almacenar mayor cantidad de información en el tipo del objeto y tiene más fiabilidad de conversión de los valores y tipos almacenados. Sin embargo, su formato es propietario: no es portable ni compatible con otros lenguajes y soluciones, de manera que sólo lo usaremos si el emisor y el destinatario son programas en C#.

Con la llegada de .NET 6 y C# 10, no se recomienda en absoluto usar este modo de serialización porque están marcados como obsoletos en .NET 6 y se eliminarán en .NET 7.

Serialización XML

A diferencia del enfoque binario, la serialización en XML se basa en un formato reconocido como un estándar en el sector. Usado durante mucho tiempo, el XML es un lenguaje de señalización bastante prolífico, pero de hecho también bastante potente y ampliable. En la actualidad todavía se usa, especialmente en el enfoque de archivos de configuración o incluso de mensajes de intercambios para los servicios que usan el protocolo SOAP.

Clase XmlSerializer

La clase XmlSerializer, que se encuentra en el espacio de nombres System.Xml.Serialization, permite realizar la serialización y deserialización de objetos en XML. Para ello hay que proporcionar un objeto y un stream. Durante la creación de la instancia de la clase XmlSerializer hay que usar la palabra clave typeof:

```
var p = new PersonaXml { Nombre = " Christophe", Apellido = "Mommer" };

var serializer = new XmlSerializer(typeof(PersonaXml));

using (var stream = new FileStream("person.xml", FileMode.Create))
{
   serializer.Serialize(stream, p);
}
using (var stream = new FileStream("person.xml", FileMode.Open))
{
   var p2 = (PersonaXml)serializer.Deserialize(stream);
   System.Console.WriteLine("Hola " + p2.Nombre + " " + p2.Apellido);
}
```

La serialización con este enfoque se basa en el uso de atributos que permiten decir lo que se quiere serializar o no, y cómo debe hacerse la serialización.

Como recordatorio, el formato XML se basa en elementos que pueden tener un contenido, pero también atributos. Se puede personalizar la salida con los atributos [XmlElement] y [XmlAttribute]. Estos últimos pueden tomar como parámetro el nombre que se quiere usar para el elemento o el atributo:

```
[Serializable]
```

```
public class PersonaXml
{
   [XmlElement("Name")]
   public string Nombre { get; set; }
   public string Apellido { get; set; }
   [XmlAttribute]
   public int Edad { get; set; }
}
```

La clase modificada aquí arriba da el siguiente XML en la serialización:

```
<?xml version="1.0"?>
<PersonaXml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" Edad="33">
  <Name>Christophe</Name>
  <Apellido>Mommer</Apellido>
</PersonaXml>
```

Se puede observar que la edad se ha colocado como atributo en el elemento principal, y el nombre se ha puesto dentro de un elemento llamado "Name".

El orden de salida de los elementos también es controlable especificando el valor Order en el atributo XmlElement:

```
[Serializable]
public class PersonaXml
{
   [XmlElement("Name")]
   public string Nombre { get; set; }
   [XmlElement(Order = 1)]
   public string Apellido { get; set; }
   [XmlAttribute]
   public int Edad { get; set; }
}
```

Si no se usa el atributo, todas las propiedades se almacenan por defecto en elementos XML y el orden usado es el de la declaración. Los subobjetos también se serializan como subelementos XML.

Serialización JSON

En la actualidad, el formato de datos JSON es el más usado para intercambiar información debido a su simplicidad, pero también a su ligereza sintáctica respecto al XML. Por ello, el framework .NET permite tratar el formato JSON de manera nativa.

Desde el framework .NET Core 3, los ingenieros de Microsoft han añadido directamente en el framework una manera nueva de tratar los flujos JSON, más eficiente que el planteamiento anterior, y todo esto para dejar de depender del paquete comunitario más ampliamente usado hasta entonces: NewtonSoft.Json.

Se realiza de manera procesal, leyendo el flujo de extremo a extremo con ayuda de un cursor gracias a la clases JsonSerializer.

JsonSerializer

La clase JsonSerializer permite realizar la serialización y deserialización de objetos en JSON. Para ello hay que proporcionar un objeto y un stream.

```
var p = new Persona { Nombre = " Christophe", Apellido = "Mommer" };

using (var stream = new FileStream("person.jsonl", FileMode.Create))
{
   var options = new JsonSerializerOptions { WriteIndented = true };
   JsonSerializer.Serialize(stream, p, options);

}
using (var stream = new FileStream("person.json", FileMode.Open))
{
   var p2 = JsonSerializer.Deserialize<Persona>(stream);
   System.Console.WriteLine("Hola " + p2.Nombre + " " + p2.Apellido);
}
```