Serialization trong C#: binary, xml, json serialization

Serialization trong C# là loại kỹ thuật chuyển đổi object về dạng trung gian (text, mảng byte) phục vụ lưu trữ (trong file) hoặc truyền qua mạng (lập trình socket). Serialization là loại kỹ thuật nền tảng cho các công nghệ phát triển ứng dụng mạng trên .NET framework như ASP.NET Web API, Windows Communications Foundation.

Bài học này sẽ cung cấp cho bạn những khái niệm và kỹ thuật cơ bản về serialization trong C# nhằm hỗ trợ cho việc đọc và ghi dữ liệu từ file.

NỘI DUNG CỦA BÀI [Ẩn]

Serialization trong C# là gì?

Khái niệm serialization

Khi tạo ra một object trong chương trình, .NET framework, bạn hầu như không cần quan tâm đến cách thức lưu trữ và quản lý dữ liệu của object đó trong bộ nhớ vì .NET framework thay bạn thực hiện các công việc này.

Tuy nhiên, nếu bạn muốn lưu trữ trạng thái của object đó (ví dụ, lưu vào file) để sau này có thể khôi phục lại nó, .NET framework lại không thể trực tiếp làm thay.

Quá trình chuyển đổi một object về dạng trung gian để lưu trữ hoặc truyền thông như vậy được gọi là data serialization (tạm dịch là *trình tự hóa dữ liệu*); Quá trình khôi phục lại object từ dạng trung gian được gọi là deserialization (tạm dịch là *giải trình tự hóa*).

Serialization có thể xem là giai đoạn chuẩn bị dữ liệu để ghi vào file hoặc truyền qua mạng. Cũng có thể coi serialization là tiền đề cho quá trình lưu trữ trạng thái của một object trong một môi trường trung gian để có thể khôi phục lại khi cần thiết.

Do môi trường trung gian (truyền thông hoặc lưu trữ) chủ yếu làm việc với hai loại dữ liệu là văn bản và nhị phân (mảng byte), quá trình serialization thực tế có thể xem là chuyển đổi object về một mảng byte, gọi là *trình tự hóa nhị phân* (binary serialization), hoặc về một chuỗi văn bản, gọi là *trình tự hóa văn bản* (text serialization).

Tuy nhiên, việc chuyển đổi này không thể tùy tiện mà phải đảm bảo thực hiện được việc giải mã để khôi phục lại object từ dạng trung gian.

Serialization thường làm việc cùng với stream để ghi dữ liệu trực tiếp vào luồng, tránh tình trạng phải lưu trữ những chuỗi hoặc mảng byte quá lớn trong bộ nhớ. Tương tự, deserialization thường cũng đọc dữ liệu từ một stream.

Hỗ trợ serialization trong C# và .NET framework

Việc chuyển một object về chuỗi ký tự hoặc mảng byte là một công việc tương đối phức tạp, tốn công sức và dễ sai sót, đặc biệt đối với các class lớn có nhiều trường dữ liệu, cũng như khi phải làm việc với nhiều class khác nhau.

Để hỗ trợ cho người lập trình, .NET framework cung cấp các class hỗ trợ cho 3 loại serialization: binary, xml và json.

Lớp `BinaryFormatter`: biến đổi một object về mảng byte và ghi trực tiếp vào một stream; đọc các byte dữ liệu từ một stream và biến đổi về object. Lớp BinaryFormatter nằm trong không gian tên `System.Runtime.Serialization.Formatters.Binary`.

Lớp `XmlSerializer`: tương tự như `BinaryFormatter`, `XmlSerializer` biến đổi một object về dạng xml và ghi vào một stream, cũng như đọc một file xml và biến đổi về object. Do làm việc với xml là một dạng dữ liệu cấp cao, `XmlSerializer` cần đến hai lớp adapter `XmlReader` và `XmlWriter` để làm việc với luồng file.

Đối với định dạng json, mặc dù .NET framework có class hỗ trợ nhưng không thực sự tốt nên chúng ta sử dụng bộ thư viện NewtonSoft.Json. Chúng ta đã cài đặt bộ thư viện này ở bài trước.

Binary serialization trong C#

Hãy cùng thực hiện một ví dụ để xem cách sử dụng của BinaryFormatter.

```
1.  using System;
2.  using System.IO;
3.  using System.Runtime.Serialization.Formatters.Binary;
4.
5.  namespace P01_Binary
6.  {
7.      [Serializable]
8.      public class Student
9.      {
10.         public int Id { get; set; } = 1;
11.         public string FirstName { get; set; } = "";
12.         public string LastName { get; set; } = "";
13.         public DateTime DateOfBirth { get; set; } = DateTime.Now;
14.     }
15.
16.     class Program
17.     {
18.         static void Main(string[] args)
19.         {
20.             var student = new Student
21.             {
22.                 Id = 1,
23.                 FirstName = "Nguyen Van",
24.                 LastName = "A",
25.                 DateOfBirth = new DateTime(1990, 12, 30)
```

```
            };
            Console.WriteLine("Original object:");
            Print(student);

            Save(student);
            var nva = Load();
            Console.WriteLine("Deserialized object:");
            Print(nva);

            Console.ReadKey();
        }

        static void Print(Student student)
        {
            Console.WriteLine($"Id: {student.Id}\r\nFirst Name: {student.FirstName}\r\nL

        }

        static void Save(Student student)
        {
            using (var stream = File.OpenWrite("data.bin"))
            {
                var formatter = new BinaryFormatter();
                formatter.Serialize(stream, student);
            }
        }

        static Student Load()
        {
            Student student;
            using (var stream = File.OpenRead("data.bin"))
            {
                var formatter = new BinaryFormatter();
                student = formatter.Deserialize(stream) as Student;
            }
            return student;
        }
    }
}
```

Trong ví dụ này chúng ta tạo ra một class Student cho thông tin về sinh viên. Trong lớp Program tạo ra 2 method: Save() để lưu thông tin sinh viên vào file nhị phân data.bin; Load() để đọc chuỗi byte từ file data.bin và chuyển đổi ngược lại thành object kiểu Student.

Để ý rằng, bên trên khai báo class Student có dòng [Serializable]. [Serializable] được gọi là **attribute** (thuộc tính). Attribute này cho phép BinaryFormatter được truy xuất thông tin của object Student cho mục đích serialization. Thiếu attribute này, ở giai đoạn runtime chương trình sẽ báo lỗi System.Runtime.Serialization.SerializationException 'Type is not marked as serializable'.

Để sử dụng BinaryFormatter, bạn phải khởi tạo object của nó trước:

```
var formatter = new BinaryFormatter();
```

> Lớp `BinaryFormatter` nằm trong namespace `System.Runtime.Serialization.Formatters.Binary`.

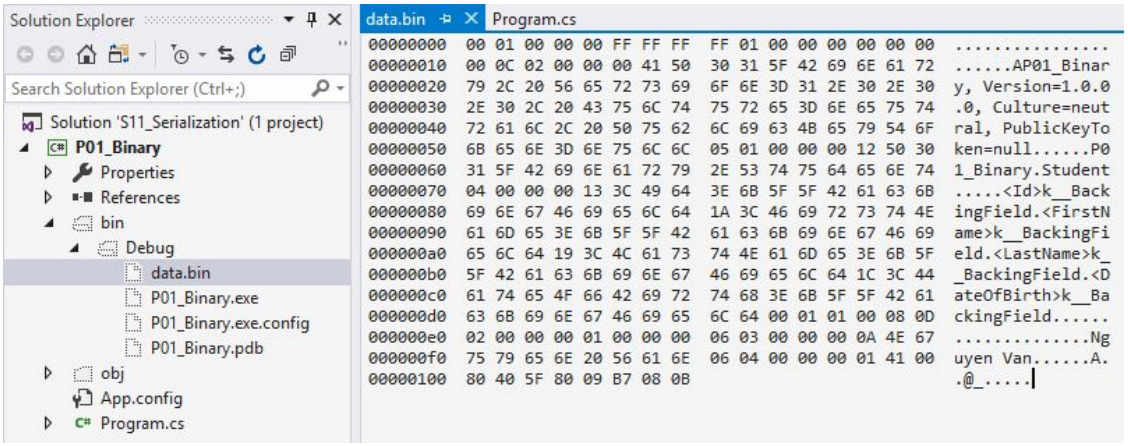
Phương thức thành viên Serialize() nhận một luồng (stream) và một object làm tham số. Phương thức này sẽ chuyển object thành chuỗi byte và ghi vào luồng.

```
formatter.Serialize(stream, student);
```

Phương thức thành viên Deserialize() nhận một luồng làm tham số. Phương thức này đọc các byte từ luồng và chuyển đổi thành một object. Bạn cần cast object này về kiểu tương ứng.

```
student = formatter.Deserialize(stream) as Student;
```

Trong ví dụ trên, bạn đều sử dụng FileStream để đọc/ghi dữ liệu với file data.bin và đặt trong cấu trúc using. File này được tự động tạo ra trong thư mục bin. Sau khi ghi vào file, nó có nội dung như sau:



Xml serialization trong C#

Chúng ta thực hiện một ví dụ tương tự để minh họa cách làm việc với Xml serialization.

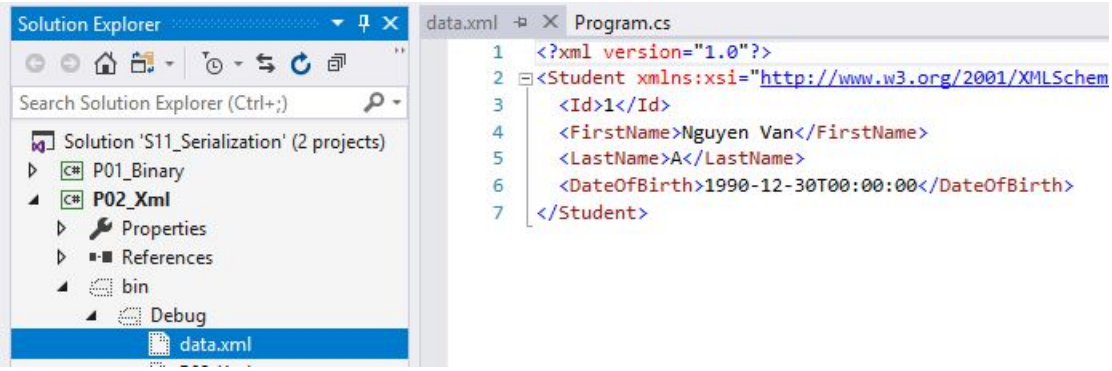
```
using System;
using System.IO;
using System.Xml.Serialization;

namespace P02_Xml
{
    public class Student
    {
        public int Id { get; set; } = 1;
        public string FirstName { get; set; } = "";
        public string LastName { get; set; } = "";
        public DateTime DateOfBirth { get; set; } = DateTime.Now;
    }

    class Program
    {
        static void Main(string[] args)
        {
            var student = new Student
            {
                Id = 1,
                FirstName = "Nguyen Van",
                LastName = "A",
                DateOfBirth = new DateTime(1990, 12, 30)
            };
            Console.WriteLine("Original object:");
            Print(student);

            Save(student);
            var nva = Load();
            Console.WriteLine("Deserialized object:");
            Print(nva);

            Console.ReadKey();
        }

```

```
        static void Print(Student student)
        {
            Console.WriteLine($"Id: {student.Id}\r\nFirst Name: {student.FirstName}\r\nL

        }

        static void Save(Student student)
        {
            using (var stream = File.OpenWrite("data.xml"))
            {
                XmlSerializer serializer = new XmlSerializer(typeof(Student));
                serializer.Serialize(stream, student);
            }
        }

        static Student Load()
        {
            Student student;
            using (var stream = File.OpenRead("data.xml"))
            {
                var serializer = new XmlSerializer(typeof(Student));
                student = serializer.Deserialize(stream) as Student;
            }
            return student;
        }
    }
}
```

Ví dụ này lặp lại hoàn toàn logic và hầu hết code của ví dụ trên.

Sự khác biệt chỉ nằm ở cách sử dụng XmlSerializer:

```
// biến object về dạng xml và ghi vào stream (FileStream)
XmlSerializer serializer = new XmlSerializer(typeof(Student));
serializer.Serialize(stream, student);

// đọc file xml và biến đổi về object
var serializer = new XmlSerializer(typeof(Student));
student = serializer.Deserialize(stream) as Student;
```



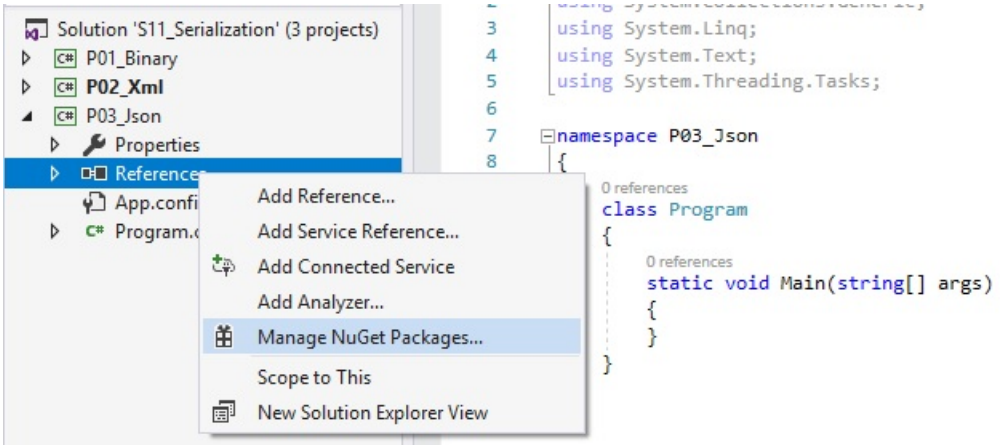
Xml serialization là một nền tảng để truyền dữ liệu trong Asp.net web API và windows communications foundation (WCF).

Json Serialization trong C#

Đối với định dạng JSON, mặc dù .NET framework có class hỗ trợ nhưng không thực sự tốt. Người ta thường dùng thư viện Json của NewtonSoft cho nhiệm vụ này. Có thể download thư viện này từ NuGet theo các bước sau:

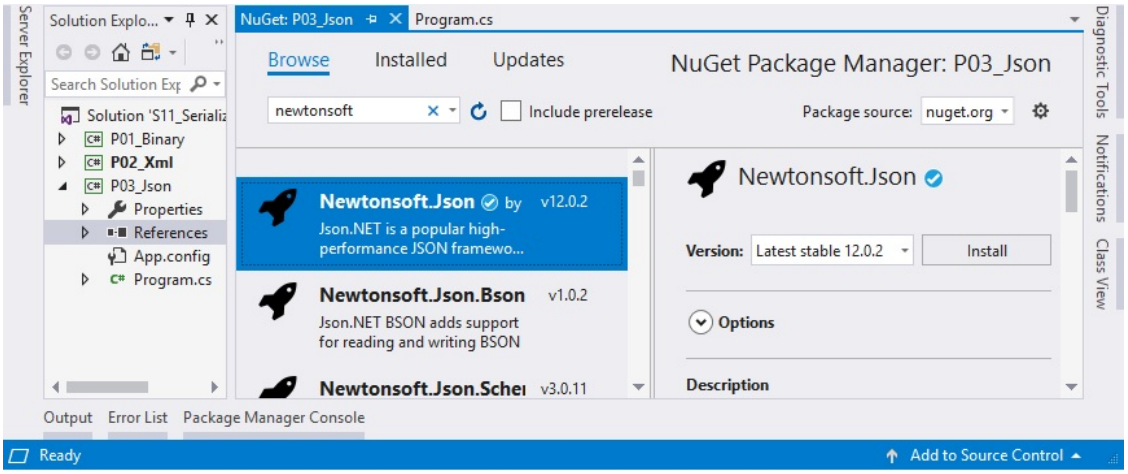
Bước 1. Mở giao diện quản lý các gói thư viện NuGet

Click phải vào References, chọn Manage NuGet Packages (xem hình dưới đây).



Bước 2. Chọn cài gói thư viện

Trong ô tìm kiếm ở tab Browse gõ *newtonsoft*, chọn gói NewtonSoft.Json và ấn Install.



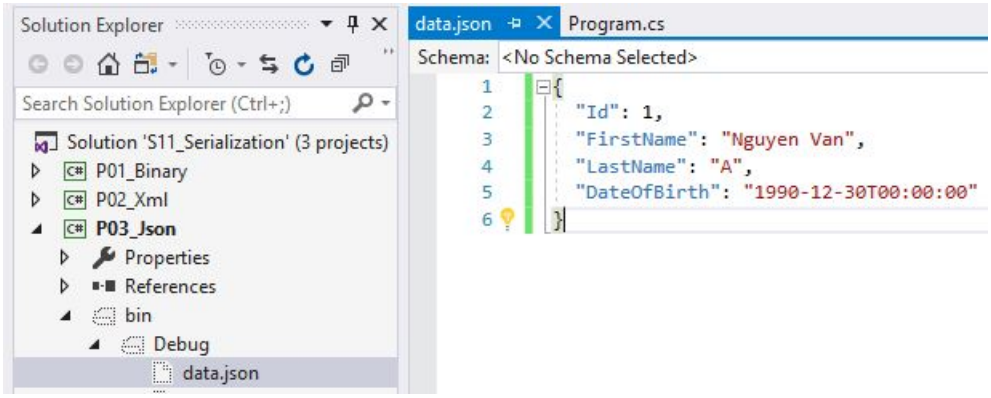
Sau lệnh này, Visual Studio sẽ tải gói thư viện này về và cài đặt lên project tương ứng.

Bạn có thể lặp lại hoàn toàn code ví dụ với xml hoặc binary serialization. Trong đó thay code của các phương thức Save và Load như sau:

```
1.  static void Save(Student student)
2.  {
3.      using (var stream = File.OpenWrite("data.xml"))
4.      {
5.          var writer = new StreamWriter(stream) { AutoFlush = true };
6.          var serializer = new JsonSerializer();
7.          serializer.Serialize(writer, student);
8.      }
9.  }
10.
11. static Student Load()
12. {
13.     Student student;
14.     using (var stream = File.OpenRead("data.xml"))
15.     {
16.         var reader = new StreamReader(stream);
17.         var serializer = new JsonSerializer();
18.         student = serializer.Deserialize(reader, typeof(Student)) as Student;
19.     }
```

```
20.         return student;
21.     }
```

Trong đó, lưu ý đặt lệnh `using Newtonsoft.Json;` ở đầu file code.



Json serialization cũng là một nền tảng để truyền dữ liệu trong Asp.net web API.

Kết luận

+ Nếu bạn thấy site hữu ích, trước khi rời đi hãy **giúp đỡ** site bằng một hành động nhỏ để site có thể phát triển và phục vụ bạn tốt hơn.
+ Nếu bạn thấy bài viết hữu ích, hãy giúp **chia sẻ** tới mọi người.
+ Nếu có thắc mắc hoặc cần trao đổi thêm, mời bạn viết trong phần **thảo luận** cuối trang.
Cảm ơn bạn!