Implementing Deserialization Techniques in .NET

Implementing Deserialization in .NET

In this lab you will set up a .NET project to handle deserialization. You'll implement deserialization for binary, XML, and JSON data formats and verify and validate the integrity of deserialized data.

Step 1: Prepare for the Application

You'll create a new .NET console application using Visual Studio Code. This program will help users deserialize data from different formats (binary, XML, and JSON) into objects.

1. Set up the Console Application:
    - Open Visual Studio Code.
    - Create a new .NET console project if one isn't already set up.
    - Delete any existing code in the Program.cs file.
2. Add the necessary libraries for working with serialization, including System.Text.Json for JSON and System.Xml.Serialization for XML.

Step 2: Implement Binary Deserialization (Using BinaryWriter and BinaryReader)

You'll implement binary deserialization by reading binary data and converting it back into a Person object.

1. Create a class Person with properties UserName (string) and UserAge (integer).
2. Use BinaryWriter to serialize a Person object as binary data into a file named person.dat.
3. Use BinaryReader to read the binary data from person.dat and convert it back into a Person object.
4. Output the deserialized data to confirm that the deserialization works as expected.

Step 3: Implement XML Deserialization

In this step, you'll implement XML deserialization to convert XML-formatted data back into a Person object.

1. Use XmlSerializer to handle XML data conversion for the Person class.
2. Read XML data from a stream or string and deserialize it into a Person object.

3. Test XML Deserialization:
    a. Create an XML-formatted string representing a Person.
    b. Deserialize the XML data and output the Person details to verify the deserialization process.

Step 4: Implement JSON Deserialization

Now, you'll implement JSON deserialization to handle data conversion from JSON format to a Person object.

1. Use System.Text.Json.JsonSerializer to handle JSON deserialization.
2. Read JSON data and deserialize it directly into a Person object.
3. Test JSON Deserialization:
    a. Define a JSON string representing a Person.
    b. Deserialize the JSON and output the details to confirm the data integrity of the deserialized object.

Step 5: Verify Integrity of Deserialized Data

This step will verify that each deserialization process produces valid, consistent results.

1. Verify that all required properties are deserialized correctly and not left as null.
2. Add error handling to each deserialization process to manage issues like missing data or type mismatches.
3. For each format (binary, XML, JSON), print a message indicating whether the deserialized data is complete and valid.

Person.cs:

```
namespace DeserializationLab.Models;

public class Person
{
    public required string UserName { get; set; }
    public int UserAge { get; set; }
    public required string Email { get; set; }
    public bool IsActive { get; set; }
}
```

Program.cs:

```
using System.Text;
using System.Text.Json;
using System.Xml.Serialization;
using DeserializationLab.Models;

class Program
{
    static void Main()
    {
        Directory.CreateDirectory("Data");

        var person = new Person
        {
            UserName = "Bob",
            UserAge = 25,
            Email = "bob@example.com",
            IsActive = true
        };

        var binaryPath = Path.Combine("Data", "person.dat");
        var xmlPath = Path.Combine("Data", "person.xml");
        var jsonPath = Path.Combine("Data", "person.json");

        Console.WriteLine("=== Binary ===");
        try
        {
            using var fsWrite = new FileStream(binaryPath,
FileMode.Create);
            using var writer = new BinaryWriter(fsWrite, Encoding.UTF8);
            writer.Write(person.UserName);
            writer.Write(person.UserAge);
            writer.Write(person.Email);
            writer.Write(person.IsActive);

            using var fsRead = new FileStream(binaryPath,
FileMode.Open);
            using var reader = new BinaryReader(fsRead, Encoding.UTF8);
            var name = reader.ReadString();
            var age = reader.ReadInt32();
            var email = reader.ReadString();
            var active = reader.ReadBoolean();
            Console.WriteLine($"Binary restored: {name}, {age}, {email},
Active={active}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Binary error: {ex.Message}");
        }

        Console.WriteLine("\n=== XML ===");
        try
        {
            var xmlSerializer = new XmlSerializer(typeof(Person));
            using var fsXml = new FileStream(xmlPath, FileMode.Create);
            xmlSerializer.Serialize(fsXml, person);
            using var fsXmlRead = new FileStream(xmlPath,
FileMode.Open);
            var xmlPersonObj = xmlSerializer.Deserialize(fsXmlRead);
            var xmlPerson = xmlPersonObj as Person;
            if (xmlPerson != null)
                Console.WriteLine($"XML restored: {xmlPerson.UserName},
{xmlPerson.UserAge}, {xmlPerson.Email}, Active={xmlPerson.IsActive}");
            else
                Console.WriteLine("XML restored: Deserialization
returned null.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"XML error: {ex.Message}");
        }

        Console.WriteLine("\n=== JSON ===");
        try
        {
            var json = JsonSerializer.Serialize(person, new
JsonSerializerOptions { WriteIndented = true });
            File.WriteAllText(jsonPath, json);

            var jsonPerson =
JsonSerializer.Deserialize<Person>(File.ReadAllText(jsonPath));
            if (jsonPerson != null)
                Console.WriteLine($"JSON restored:
{jsonPerson.UserName}, {jsonPerson.UserAge}, {jsonPerson.Email},
Active={jsonPerson.IsActive}");
            else
                Console.WriteLine("JSON restored: Deserialization
returned null.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"JSON error: {ex.Message}");
        }

        Console.WriteLine("\n=== Integrity Check ===");
        try
        {
            var jsonPerson =
JsonSerializer.Deserialize<Person>(File.ReadAllText(jsonPath));
            if (!string.IsNullOrEmpty(jsonPerson?.UserName) &&
jsonPerson.UserAge > 0 && !string.IsNullOrEmpty(jsonPerson.Email))
                Console.WriteLine("Integrity check passed: all formats
preserved data correctly");
            else
                Console.WriteLine("Integrity check failed: some data
missing or corrupted");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Integrity check error: {ex.Message}");
        }
    }
}
```