Name :- Utkarsh Patil

Class :- MSc ICT sem-2

Subject :- ICT 201 : C#.NET

Roll No :- 35

Assignment :- 3

---

(1) What is serialization? Explain types of serialization and its usage with requirement.

⟹ Serialization is the process of converting an object into a stream of bytes to store the object or transmit it to memory, a database or a file.

- Its main purpose is to save the state of an object in order to be able to recreate it when needed. The reverse process is called deserilization.

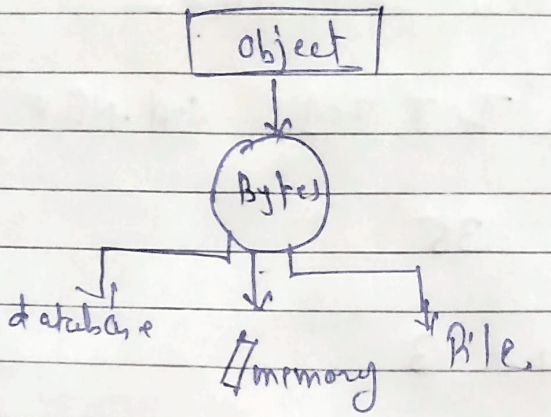
- How serialization works :-

- this illustration show the process of serialization.

- The object is serialized to a stream that carries the data.
- The stream may also have information about

ROOPAM
PAGE NO:
DATE:

the stream, the object can be stored in a database, a file, or memory.



- Uses of serialization:

- serialization allows the developer to save the state of an object and re-create it as needed, providing storage of objects as well as data exchange. Through serialization, a developer can perform actions such as:

- sending the object a remote application by using a web service.
- passing an object from one domain to another.
- passing an object through a firewall as a JSON or XML string.
- maintaining security or user-specific information across applications.

- Type of serialization:-

- ① Json serialization :-

The System.Text.Json

namespace contain classes for Javascript object Notation (JSON) serialization and deserialization. JSON is an open standard that is commonly used for sharing data across the web.

- 2) Binary and XML serialization:-

- System.Runtime.serialization namespace contains classes for binary and XML serialization and deserialization.

- Binary serialization uses binary encoding to produce compact serialization for uses such as storage or socket-based network streams. In binary serialization, all members, even members that are read-only, are serialized, and performance is enhanced.

- XML serialization serializes the public and properties of an object, or the parameters, and return values of methods, into a XML stream that conforms to a specific XML schema definition language (XSD) document. XML serialization result in strongly typed classes with public properties and fields that are converted to XML.

- 3) Making an object serializable:-

- For any for XML serialization, you need,
  - the object to be serialized.

- A stream to contain the serialized object.
- A system.Runtime.Serialization.Formatter instance.

(v) Basic and custom serialization.

- Binary and xml serialization can be performed in two ways, basic and custom.

- Basic serialization uses .NET to automatically serialize the object. The only requirement is that the class has the SerializableAttribute attribute applied. The NonSerialized Attribute can be used to keep specific field from being serialized.

(2) Explain Binary serialization and Deserialization of Generic collection of user defined class in details.

→ Serialization can be defined as the process of storing the state of an object to a storage medium.

During this process, the public and private field of the object and the name of the class, including the assembly containing the class, are converted to a stream of bytes, which is then written to a data stream. When the object is subsequently deserialized an exact clone of the original object is created.

Binary serialization allows modifying private member inside an object and therefore changing the state of it. Because of this, other serialization frameworks, like System.Text.Json, that operate on the public API surface are recommended.

example

```
using System;
using System.Collections;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;

protected void Page_Load(object sender, EventArgs e)
{
    Employee emp1 = new Employee();
```

```
emps.Add (new Employee("1", "Dheeraj"));
emps.Add (new Employee("2", "Anjali"));
emps.Add (new Employee("3", "Deepak"));
emps.Add (new Employee("4", "Harish"));
emps.Add (new Employee("5", "Suresh"));

String pth = @"D:\Test.bin";

Serialize<Employees> (emps, pth);
Deserialize (pth);
}

public void Serialize<T> (T emps, String Filename)
{
    System.IO.Stream ms = File.OpenWrite (Filename);

    BinaryFormatter formatter = new BinaryFormatter();

    formatter.Serialize (ms, emps);
    ms.Flush();
    ms.Close();
    ms.Dispose();
}

public T Deserialize<T> (String Filename)
{
    BinaryFormatter formatter = new BinaryFormatter();

    FileStream fs = File.Open (Filename, FileMode.Open);

    object obj = formatter.Deserialize (fs);
```

```
Temp1 = (T)Obj;
    fs.Flush();
    fs.Close();
    fs.Dispose();
    return temp1;
}

public void Deserialize(String filename)
{
    Employee emps = Deserialize<Employee>
                    (filename);

    Foreach (Employee employee in emp1)
    {
        Response.Write(employee.Name + "<br>");
    }
}
```