C# Advanced – Class12

Trainer: Miodrag Cekikj – cekicmiodrag@gmail.com

Assistant: Andrej Chichakovski – andrejchichak@gmail.com

SEAVUS EDUCATION *and* DEVELOPMENT CENTER

Agenda

- Input and Output
  - Working with Json files
- Serialization and Deserialization
  - Binary serialization
  - Xml serialization
  - Json serialization
  - Discussion
- Hands on activities ☺

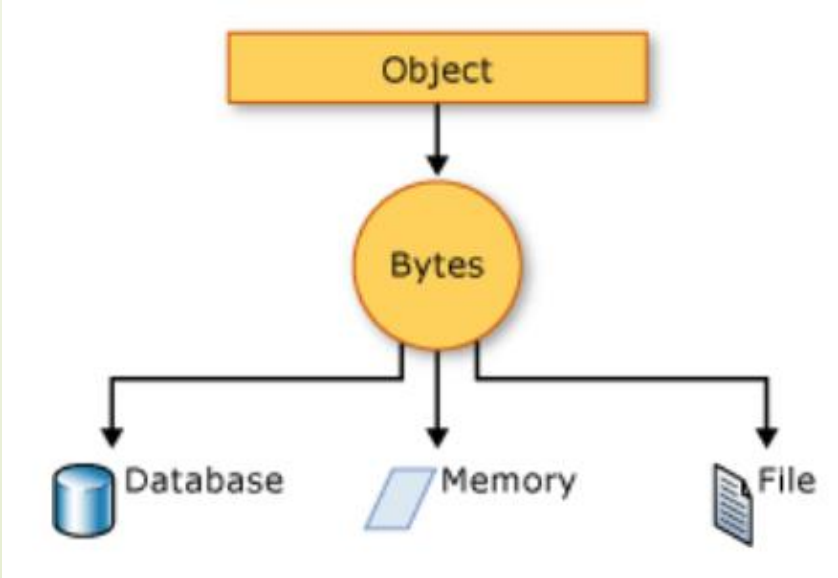
Working with Json files

- Json.NET is a popular high-performance **JSON framework** for .NET
  - https://www.nuget.org/packages/Newtonsoft.Json/
  - using Newtonsoft.Json;
  - using Newtonsoft.Json.Linq;

- Working with JSON:
  - **Creating JSON** and writing to JSON file
  - **Reading JSON** from file
  - **Writing a collection** into JSON file
  - **Writing JSON array** with JArray to JSON file

Serialization

- **Serialization** is the **process of converting** an **object's state** into **information** that can be stored for later retrieval or that can be sent to another system (store the object or transmit it to memory, a database, or a file).
- **Purpose**: to save the state of an object in order to be able to recreate it when needed. The reverse process is called deserialization.



The object is serialized to a stream, which carries not just the data, but information about the object's **type**, such as its **version**, **culture**, and **assembly name**. From that stream, it can be stored in a database, a file, or memory.

Serialization

- Why and Where? ☺
  - **Allows** the developer to save the state of an object and recreate it as needed, providing **storage of objects** as well as **data exchange**.
  - A developer can perform actions like sending the object to a remote application by means of a **Web Service**, passing an object from one **domain to another**, passing an object through a firewall as an **XML string**, or **maintaining security** or **user-specific information across applications**.

- What do I need to know? ☺
  - Object to be serialized.
  - Stream to contain the serialized object.
  - Formatter.

Serialization

- **Binary serialization**
  - Binary serialization allows **single objects or complex models** to be **converted to binary streams**, which may be stored in files or transported to other systems.
- **Xml serialization**
  - With XML serialization, the public state of **objects can be converted into an XML document**. Such XML information is often stored on disk to persist data for later use or is transported over a network or the Internet to send messages between computers.
- **Json serialization**
  - Similar as the XML serialization, but **more flexible and more efficient than the others**.

Deserialization

- **The reversible process of the serialization is called Deserialization**. There is method **Deserialize()** both to the binary and xml serializations.
- Json has the **DeserializeObject** method.
- A commonly experienced problem is that the **constructor** for an object is not executed when data is deserialized.

Json serialization

- The **JsonSerializer** converts .NET objects into their JSON equivalent and back again by **mapping the .NET object property names to the JSON property** names and copies the values for you.

- The quickest method of converting between JSON text and a .NET object is using the JsonSerializer.

- using Newtonsoft.Json;
  - NuGet package

Demos

- Binary serialization
- Xml serialization
- Json serialization

Binary vs. XML vs. Json serialization

- The binary serialization process uses classes from the **System.Runtime.Serialization** and **System.Runtime.Serialization.Formatters.Binary** namespaces. This gives a concise result and ensures that when the data is deserialized, the object structure is correctly reconstructed.
- XML/Json serialization convert the state of objects into XML/Json. **System.Xml.Serialization/Newtonsoft.Json** namespaces. This allows the information to be deserialized into different data types, including into software that has been created using technologies other than the .NET framework. **As XML/Json documents can be verbose, the serialized information can be larger than its binary equivalent. However, it is human-readable and, in appropriate scenarios, can be easily edited.**
- One important disadvantage of XML serialization is that **private properties and fields are not extracted** so cannot be recreated from the data. This is not the case at the Binary and Json serializations.
- The **[Serializable]** attribute is <u>not required for the XML/Json serialization</u>, while it's <u>required for the Binary serialization</u>.

Discussion

- Any questions or discussion? ☺

- FileStream vs. StreamWritter
  - Both, **FileStream** and **StreamWriter** can be used for **writing**.
  - Both, **FileStream** and **StreamReader** can be used for **reading**.

  - A **FileStream** is a **Stream**. Like all Streams it only deals with **byte[] data**.
  - A **StreamWriter : TextWriter**, is a **Stream-decorator**. A TextWriter encodes Text data like string or char to byte[] and then writes it to the linked Stream.

Discussion

- FileStream vs. StreamWritter
  - Use a **bare <u>FileStream</u>** when you have **<u>byte[] data</u>**.
  - Add a **<u>StreamWriter</u>** when **<u>you want to write text</u>**. **Use a Formatter or a Serializer to write more complex data**.

- Can I combine combine these two into one?
  - **<u>You always need a Stream to create a StreamWriter</u>**. The helper method **System.IO.File.CreateText("path")** will create them in combination.

Discussion

- FileStream vs. StreamWritter
  - **FileStream** -> Provides a Stream for a file, supporting both synchronous and asynchronous read and write operations.
  - https://docs.microsoft.com/en-us/dotnet/api/system.io.filestream?view=netframework-4.8
  - **StreamWriter** -> Implements a TextWriter for writing characters to a stream in a particular encoding.
  - https://docs.microsoft.com/en-us/dotnet/api/system.io.streamwriter?view=netframework-4.8
  - **StreamReader** -> Implements a TextReader that reads characters from a byte stream in a particular encoding.
  - https://docs.microsoft.com/en-us/dotnet/api/system.io.streamreader?view=netframework-4.8

Hands on activities

- Workshop time, check the challenge on Git. ☺