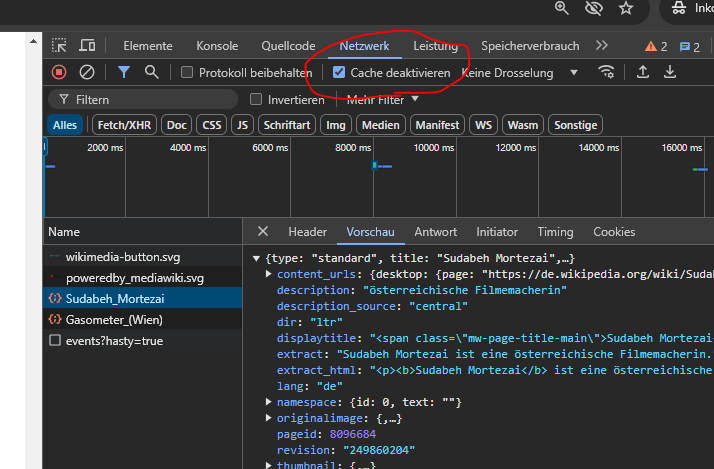
**Serialisieren und Deserialisieren**

* **Definition** der Begriffe:
  + **Serialisierung**: Umwandlung eines Objekts in ein Format, das **gespeichert**, **übertragen** oder **transportiert** werden kann.
  + **Deserialisierung**: Der umgekehrte Prozess, bei dem die serialisierten Daten **zurück in ein Objekt** umgewandelt werden.
* Warum Serialisierung?
  + Um Daten in einem **transportablen** Format(**JSON**, **XML**, **Binary**, etc.) zu bringen, um die…
  + …Kommunikation zwischen verschiedenen Systemen (z. B. **Browser zu Server-Kommunikation im Web**) zu gewähren.
* Vergleich der Formate: JSON, XML, Binärserialisierung

**JSON** (JavaScript Object Notation): Menschlich lesbares Format, „leichtgewichtig“, ideal für **Web-APIs und REST-Dienste**.  
Auch in JavaScript direkt verarbeitbar. JSON wird häufig auf großen Webseiten zum Nachladen von Inhalten verwendet.

**Selbst Ausprobieren**:

* + - Browser
    - <https://de.wikipedia.org/wiki/Simmering>
    - Browsertools – Reiter „Netzwerk“ (
    - Auf der Seite nach unten scrollen und im Entwicklertools Nachladeprozesse mit JSON-Häppchen mitverfolgen.



Hier wird u.a. als eine in JavaScript getriggerte Anforderung (Request) auf (<https://de.wikipedia.org/api/rest_v1/page/summary/Universit%C3%A4t_Wien>) eine Antwort (Response) in JSON-Format gesendet.

Die Darstellung eines JSON-Strings basiert auf dem Konzept der **Schlüssel-Wert** - **Paare** (keys-value-pairs). Z.B. **{ "name": "Max Mustermann" } .** Wobei **„name“** ist der **Schlüssel** und **„Max Mustermann“** dessen **Wert**.

JSON wird oft für **Konfigurationsdateien** verwendet

**Umwandlungsbeispiel.  
 Aus Objekt person:** var person = new Person  
{

GeburtsDatum = new DateOnly(2000, 5, 15),

Gewicht = 75,

Name = "Max Mustermann",

Haarfarbe = Haarfarbe.Braun,

Adresse = new Adresse,

Lebendig = true,

{

Strasse = "Hauptstraße 123",

Stadt = "Wien",

PLZ = "1140",

Land = "Österreich"

},

Schuhgroesse = 42

};

**…wird in JSON:**

{

"GeburtsDatum": "2000-05-15",

"Gewicht": 75,

"Name": "Max Mustermann",

"Haarfarbe": "Braun",

"Adresse": {

"Strasse": "Hauptstraße 123",

"Stadt": "Wien",

"PLZ": "1140",

"Land": "Österreich"

},

"Lebendig": true,

"Schuhgroesse": 42

}

**XML**: ähnlich wie JSON, aber mit **umfangreicherer Syntax** (wegen den vielen spitzigen Klammern, öffnenden und schließenden Tags und der Unterstützung von Attributen).  
XML wird oft in älteren Systemen (wie: SOAP-Dienste (Webservices, asmx)) verwendet.

**Obiges Beispiel als XML serialisiert:**

<Person xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<GeburtsDatum>2000-05-15</GeburtsDatum>

<Gewicht>75</Gewicht>

<Name>Max Mustermann</Name>

<Haarfarbe>Braun</Haarfarbe>

<Adresse>

<Strasse>Hauptstraße 123</Strasse>

<Stadt>Beispielstadt</Stadt>

<PLZ>1140</PLZ>

<Land>Österreich</Land>

</Adresse>

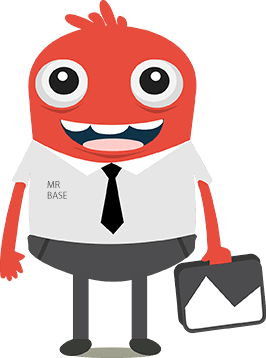
<Lebendig>true</ Lebendig>

<Schuhgroesse>42</Schuhgroesse>

</Person>

**Binäre Serialisierung**: Speichert Daten in einem kompakten, binären Format, das **nicht menschenlesbar**, dafür aber sehr **schnell** und **speichereffizient** ist. Häufig verwendet in **internen Anwendungen** oder bei der Übertragung **großer Datenmengen** per Netz. Für eine HTTP-Übertragung müssen die binär serialisierten Daten in erstem Schritt in das textbasierte **Base64**-Format kodiert werden.

**Beispiel Kodierung zu Base64**

**Aus den binären Daten dieses Bildes**  


**…wird (für eine HHTP-Übertragung) einer zu Base64 kodierter String:**



**… den der Client ein sein Ursprungsformat (z.B. JPG) bringen (dekodieren) kann**. Online-Dekodierung: <https://base64.guru/converter/decode/image>

**Einfaches Beispiel Serialisieren und Deserialisieren mit JSON**

***Schritt 1:*** *Einrichtung des Projekts*

* Neues C#-Projekt erstellen (Konsolenanwendung).

***Schritt 2:*** *Implementierung der Klasse*

using System;

using System.Text.Json;

public class Person

{

public string Name { get; set; }

public int Alter { get; set; }

}

***Schritt 3:*** *Serialisierung einer Instanz*

class Program

{

static void Main(string[] args)

{

// Objekt erstellen

Person person = new Person { Name = "Max Mustermann", Alter = 25 };

**// Objekt in JSON serialisieren**

string jsonString = JsonSerializer.**Serialize**(person);

Console.WriteLine("Serialisiertes JSON: " + jsonString);

}

}

***Schritt 4:*** *Deserialisierung des JSON-Strings*

class Program

{

static void Main(string[] args)

{

// JSON-String

string jsonString = "{\"Name\":\"Max Mustermann\",\"Alter\":25}";

**// JSON in Objekt deserialisieren**

Person person = JsonSerializer.**Deserialize**<Person>(jsonString);

Console.WriteLine("Name: " + person.Name);

Console.WriteLine("Alter: " + person.Alter);

}

}

**Speichern von JSON - Strings auf der Festplatte**

var person = new Person();

var pfadUndDateiName = “C:\Temp\person.json“;

var jsonString = JsonSerializer.Serialize(person);

**File.WriteAllText(pfadUndDateiName, jsonString);**

**Deserialisierung eines auf der Festplatte abgespeichertem JSON in ein Objekt**

var pfadUndDateiName = “C:\Temp\person.json“;

var jsonString = File.ReadAllText(pfadUndDateiName);

var person = JsonSerializer.Deserialize<Person>(jsonString);

**Aufgabe**

Speichern Sie den JSON-String auf Ihrer Festplatte und lesen Sie diesen wieder in ein Person-Objekt ein.

**Weiterte Gestaltungsmöglichkeiten**

Es gibt eine Vielzahl von Gestaltungsmöglichkeiten mit Anweisungen, **WIE** Objekte in JSON und umgekehrt umgewandelt werden sollen.

Man kann es machen mit **JsonSerializerOptions**, die dem **Serializer** und **Deserializer** als Argument mitgegeben werden oder auch durch spezielle **Attribute**, die zu setzen sind an den Properties (Eigenschaften) des zu serialisierenden bzw. desozialisierenden Objektes.

**Beispiel**

var jsonSerializerOptions = new **JsonSerializerOptions**

{

PropertyNamingPolicy = JsonNamingPolicy.CamelCase,

WriteIndented = true,

DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull

};

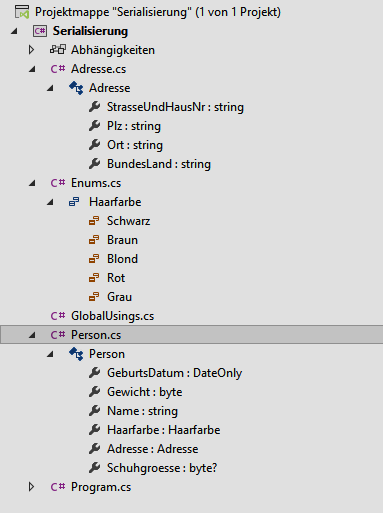
**[JsonPropertyName("Wohnadresse")]**

public required Adresse Adresse { get; set; }

**Klassenstruktur für ein gemeinsames, fortgeschrittenes Beispiel**

Erstellen Sie folgende Projektstruktur.

Das Weitere lernen wir gemeinsam.



**Teilnehmer-Projekt zum Thema Serialisieren und Deserialisieren**

Siehe:

Aufgabe **Bücherverwaltung PREMIUM.docx** und **Aufgabe Bücherverwaltung STANDARD.docx**