



# Serializáció, deserializáció

Répasné Babucs Hajnalka

Répás Csaba



# Serializáció, deserializáció fogalma



- A **szerializáció** (**serialization**) olyan tevékenység, amellyel egy értéket, adatszerkezetet átalakítunk olyan formára, hogy az külső adattárolóra lementhető legyen, és később ebből az eredeti állapot helyreállítható legyen.
- A helyreállítási folyamatot **deszerializációnak** (**deserialization**) nevezzük.
- A szerializáció során a tényleges érték elé bevezető információk kerülnek, melyek egyértelműsítik, hogy milyen típusú adatról van szó, hogyan kell majd a deszerializáció során a tényleges értékeket tartalmazó adatsort értelmezni.





# Példa osztály: Aru



```
public class Aru
{
    2 references
    public string Nev { get; init; }
    3 references
    public int EgysegAr { get; init; }
    3 references
    public int AfaKulcs { get; init; }
    3 references
    public int Db { get; init; }
    2 references
    public List<string> Leiras { get; init; }
    = new List<string>();
    1 reference
    public int FizetendoAr
    => (int)Math.Round(
        EgysegAr * (1 + AfaKulcs / 100.0) * Db,
        MidpointRounding.ToPositiveInfinity);
    0 references
    public override string ToString()...
}
```

- Alapértelmezetten a publikus tulajdonságok szerializálódnak
  - Private tulajdonságok és mezők is megjelölhetők
- Alapértelmezetten minden publikus tulajdonság adata mentődik
  - A számított tulajdonság is!
- Alapértelmezetten a paraméter nélküli konstruktor segítségével deszerializálódik az adat



# Példány az Aru osztályból



```
Aru aru = new()  
{  
    Nev = "alma",  
    EgysegAr = 120,  
    AfaKulcs = 27,  
    Db = 6,  
    Leiras = {"finom", "édes" }  
};
```

- Az osztálynak nem írtunk konstruktort, alapértelmezetten létre jött egy paraméter nélküli konstruktor
- Az adatokat az alapértelmezett konstruktor hívásakor, a tulajdonságok értékének megadásával állítjuk be
- A számított tulajdonság értékét nem állítjuk be



# Serializálás, deserializálás



```
string jsonString = JsonSerializer.Serialize(aru);  
Console.WriteLine(jsonString);  
File.WriteAllText("aru.json", jsonString);
```

Serializálás

Kiírt, fájlba mentett szöveg

```
{"Nev":"alma","EgysegAr":120,"AfaKulcs":27,"Db":6,"Leiras":["finom","\u00E9des"],"FizetendoAr":915}
```

Deserializálás

```
Aru aru2 = JsonSerializer.Deserialize<Aru>(jsonString)!;
```

```
Név: alma egység ár: 120 Ft áfa kulcs: 27 darabszám: 6 fizetendő ár: 915 Ft Leírás: finom, édes
```

Deserializálás után kiírt adatok a ToString() metódus segítségével





# Serializálás opciók

```
var option = new JsonSerializerOptions
{
    WriteIndented = true,
    Encoder = JavaScriptEncoder.UnsafeRelaxedJsonEscaping
};
string jsonString2 = JsonSerializer.Serialize(aru2, option);
Console.WriteLine(jsonString2);
File.WriteAllText("aru2.json", jsonString2);
```

Serializálás formázott JSON-ra

Ékezetes karakteres tárolás

```
{
  "Nev": "alma",
  "EgysegAr": 120,
  "AfaKulcs": 27,
  "Db": 6,
  "Leiras": [
    "finom",
    "édes"
  ],
  "FizetendoAr": 915
}
```

Kiírt, fájlba mentett szöveg

Ékezetes karakteres tárolás



# Serializálás generikus típussal



- Szerializálni és deserializálni az adatokat generikus típusmegadással is lehet:

```
string jsonString3 = JsonSerializer.Serialize<Aru>(aru2!, option);  
Console.WriteLine(jsonString3);
```

```
var aru5 = JsonSerializer.Deserialize<Aru>(jsonString3);  
Console.WriteLine(aru5);
```





# [JsonIgnore], [JsonInclude]



[JsonIgnore]

1 reference

```
public int FizetendoAr  
    => (int)Math.Round(  
        EgysegAr * (1 + AfaKulcs / 100.0) * Db,  
        MidpointRounding.ToPositiveInfinity);
```

- A [JsonIgnore] attribútummal megjelölt tulajdonságok nem kerülnek serializálásra

```
{"Nev": "alma", "EgysegAr": 120, "AfaKulcs": 27, "Db": 6, "Leiras": ["finom", "édes"]}
```



[JsonInclude]







```
public string Marka;
```

- A [JsonInclude] attribútummal meg lehet jelölni mezőket is serializálásra





# [JsonConstructor]

- 
- 
- 
- 
- 
- 
- Ha az osztály több konstruktorral rendelkezik, akkor el kell dönteni, hogy deszerializáláskor melyik konstruktornak kell lefutnia.
  - Ha valamelyik konstruktor meg van jelölve [JsonConstructor] attribútummal, akkor az fut le.
  - A paramétereknek ugyanolyan nevűeknek és típusúaknak kell lennie, mint a tulajdonságoknak, amiből szerializálódott.
  - Ha egyik konstruktor sincs megjelölve, és van paraméter nélküli konstruktor, akkor a paraméter nélküli konstruktor fut le.
  - Ha nincs paraméter nélküli konstruktor, és egyik konstruktor sincs megjelölve, akkor futás idejű hiba keletkezik.