[Flags]

with

enums

What is that?

Stefan Djokic

Purpose of [Flags]

Indicates that an enum should be treated as a set of flags

```
1  [Flags]
2  enum AccessRigths
3  {
4      None    =  0, //0000
5      Read    =  1, //0001
6      Write   =  2, //0010
7      Execute = 4   //0100
8  }
```

Each value is assigned a power of 2, so each value corresponds to a different bit being set in the number's binary representation

Stefan Djokic

Combining Flags

To combine flags, you'd typically use the bitwise OR | Operation

```
1  AccessRights readWrite = AccessRights.Read | AccessRights.Write
```

Checking Flags

To check if a particular flag is set, you'd use bitwise AND & operation

```
1  bool hasRead = (readWrite & AccessRights.Read) == AccessRights.Read;
2  //Result true = 0011 AND 0001 is 0001
```

stefandjokic.tech

Stefan Djokic

Removing Flags

Combination of bitwise AND op and bitwise NOT op

```
readWrite = readWrite & ~AccessRights.Read;
// Result: 0010 in binary (2 = Write Access)
```

[Flags] and .ToString()

```
AccessRights myRights = AccessRights.Read | AccessRights.Write;
Console.WriteLine(myRights); //Output: Read, Write
```

stefandjokic.tech

Stefan Djokic

Best Practices

1. Always include a 'None' value set to 0. This represents the absence of any flag.

2. Assign powers of 2 as values to ensure each value corresponds to a unique bit.

3. Only use the [Flags] attribute when you intend the enum to represents multiple values simultaneously.

Stefan Djokic

DO YOU LIKE THE POST?

REPOST IT!