Attributes

ב #C ניתן להוסיף אינפורמציה על הקוד באמצעות Attributes. Attributes מתארים את הקוד ומסייעים להבין מה ביקשנו לבצע בו, בלי להיכנס לפרטים הטכניים כיצד זה מבוצע.

כאשר מקמפלים את הקוד (מבצעים build), והוא מתורגם ל MSIL, מצורף לקוד גם metadata. ה - metadata זהו בעצם מידע על מידע כלשהו, ובמקרה של #C - metadata הוא מידע על הקוד. Attributes הם חלק מה metadata של הקוד.

במהלך הרצת הקוד, באמצעות Reflection ניתן לקרוא את ה metadata, שכולל גם attributes, ועל פיו לבצע או להימנע מביצוע פעולות שונות.

כמו כן, לעיתים כבר בשלב הקומפילציה (ולא רק הרצה) נעשה שימוש במידע של ה attributes.

ב net.נעשה שימוש ב attributes למטרות שונות ולפתרונות במצבים שונים. עם attributes ניתן להגדיר איך לבצע serialization, הגדרות שונות לאבטחת מידע, חשיפה או הסתרת מידע, ועוד.

כיצד משתמשים ב Attributes?

1. ראשית יש ליצור את ה attribute, או להשתמש באחד שמוגדר כבר ב net framework. (קיימים רבים כאלה)
2. בוחרים את הקוד אותו רוצים לסמן, וכותבים את שם ה attribute מוקף בסוגריים מרובעים , לדוגמה Obsolete attribute שמציין קטע קוד ככזה שלא מומלץ להשתמש בו אלא במחליפו.

```
[Obsolete]
public void FuncA()
```

כתיבת ה attribute שקולה ליצירת אובייקט מסוג המחלקה שלו. כלומר:

```
Obsolete obsolete = new Obsolete();
```

3. ניתן לציין פרמטרים ל attribute – וזה לפי הפרמטרים שהוגדרו ב ctor של מחלקת ה attribute, או public members עם אפשרות לקריאה וכתיבה שקיימים במחלקה.
לדוגמא: ל Obsolete attribute יש ctor שמקבל מחרוזת, (זו המחרוזת שתוצג למשתמש בחלון האזהרות).

```
[Obsolete("It is recommended to use BetterFuncA instead of FuncA")]
```

ודוגמה לשילוב בין פרמטרים ב ctor לבין Named parameters - כאן ה Named parameters לא מוגדרים ב ctor אלא כיון שהם קיימים במחלקת ה Attribute כחברי מחלקה שניתן לקרוא ולכתוב מהם, מתייחסים אליהם כאילו הם named parameters.

```
[AttributeUsage(AttributeTargets.Field, AllowMultiple = true, Inherited = true)]
AttributeUsageAttribute(AttributeTargets validOn, Properties: [AllowMultiple = bool], [Inherited = bool])
Initializes a new instance of the AttributeUsageAttribute class with the specified list of AttributeTargets, the Att
validOn: The set of values combined using a bitwise OR operation to indicate which program elements are valid.
```

רכיבי קוד שניתן לסמן ב attributes:

- Classes
- Methods
- Enums
- Events
- Fields
- Properties
- Structs
- Parameters
- Constructors
- Delegates
- Interfaces
- Return values
- Assemblies
- Other Attributes too!

יצירת Custom Attribute

לעיתים עולה הצורך ליצור custom attribute – attribute שיתאים לצרכים מסוימים שיש לנו בקוד שלנו ספציפית.

ניתן ליצור attribute בקלות יחסית, כדאי לדעת מספר דברים:

1. המחלקה צריכה לרשת ממחלקת Attribute שקיימת כבר בדוט נט, כך שתיכלל ב metadata ללא עבודה נוספת מצידנו.
2. על פי הקונבנציה, שם המחלקה יהיה שם ה attribute שלנו בתוספת המילה Attribute בסוף, לדוגמא: SecretAttribute. בשימוש ב attribute ניתן להשמיט את הסיומת Attribute, #C מזהה אותו גם כך. (קונבנציה – לא חייבים, הקוד יעבוד גם אם לא)
3. כל attribute חייב להשתמש ב UsageAttribute, באמצעותו:
   a. חובה לציין לאילו רכיבי קוד יתאים ה attribute, זהו פרמטר חובה ב ctor.
   ערך זה הוא enum מסוג `AttributeTargets`, זהו enum שמסומן ב...כן, attribute בשם Flags, באופן זה ניתן לבחור יותר מערך אחד, וההיחסות היא כמו אל מערך של bit דלוקים\כבויים. כך בוחרים כמה מהאופציות:

```
AttributeTargets.Class | AttributeTargets.Method | AttributeTargets.Property
```

   b. ניתן לציין האם ניתן לכתוב את ה attribute יותר מפעם אחת על רכיב קוד, באמצעות `AllowMultiple`
   c. ניתן לציין האם ה attribute ישפיע על מחלקות יורשות או פונקציות דורסות, באמצעות `Inherited`
4. הגדרת ctors במידת הצורך
5. הגדרת properties – אם יהיו read\write (עם public get & set))המשתמש יוכל להכניס ערך גם אם לא יצויינו במפורש כפרמטרים ב ctor.

דוגמה:

```
[AttributeUsage(AttributeTargets.Property, AllowMultiple = true, Inherited = true)]
public class ColorAttribute : Attribute
```