DYNAMIC TYPE IN C#

TUTORIAL

หลักการทำงานของ Dynamic Type

Dynamic Type ใน C# คือฟีเจอร์ที่ทำให้เราประกาศตัวแปรโดยไม่ต้องกำหนดชนิดข้อมูลตั้งแต่แรก

ช่วยให้เราสามารถเลื่อนการตรวจสอบประเภทตัวแปรจากช่วงเวลาคอมไพล์ (Compile-time) ไปเป็นช่วงเวลารันไทม์ (Run-time) ได้

```
dynamic x = 10;          // x ตอนนี้เป็น int
x = "Hello";             // x เปลี่ยนเป็น string
x = new List<int>(); // x กลายเป็น List<int>
```

ความสำคัญของ Dynamic Type ใน C#

Dynamic Type ใน C# มีความสำคัญเพราะช่วยให้เราเขียนโค้ดได้อย่างยืดหยุ่นและสะดวกมากยิ่งขึ้นในบางกรณี

ข้อดีหลักๆ ของ dynamic type คือไม่ต้องมากังวลเรื่องการกำหนดชนิดของตัวแปรล่วงหน้า

การใช้ dynamic type เลยสำคัญในงานที่ต้องการความยืดหยุ่นสูง แต่ก็ต้องใช้ด้วยความระมัดระวังเพื่อไม่ให้เกิด bug ในภายหลัง

ความแตกต่างระหว่าง
Dynamic Typing และ Static Typing

Dynamic Typing : ยืดหยุ่น ไม่ต้องระบุชนิดล่วงหน้า แต่มีโอกาสเจอ bug ตอน runtime

Python

```
variable = "Hello"
variable = 10
```

C#

```
dynamic variable = "Hello";
variable = 10;
```

Static Typing : เสถียร ตรวจจับข้อผิดพลาดได้เร็ว แต่ต้องระบุชนิดให้ชัดเจนตั้งแต่แรก

Java, C#

```
int number = 5;
number = "Hello";
```

เปรียบเทียบ Dynamic Typing กับภาษาอื่นๆ

1. C (Static Typing)
ในภาษา C ตัวแปรทุกตัวต้องระบุชนิดข้อมูลตั้งแต่ช่วงคอมไพล์ เช่น int, char, หรือ float และเปลี่ยนชนิดข้อมูลไม่ได้หลังจากประกาศแล้ว ถ้าเรากำหนดชนิดผิดหรือพยายามเปลี่ยนชนิด ระบบจะแจ้งเตือนข้อผิดพลาดในตอนคอมไพล์

ข้อดี : ตรวจจับข้อผิดพลาดได้ตั้งแต่ตอน compile-time ทำให้โปรแกรมมีความเสถียรมาก
ข้อเสีย : เขียนโค้ดได้ช้ากว่า เพราะต้องระบุชนิดข้อมูลทุกครั้ง

```c
#include <stdio.h>

int main() {
    void *variable;       // ใช้ void pointer เพื่อเก็บที่อยู่ของข้อมูล

    variable = "Hello";  // เก็บ string
    printf("%s\n", (char *)variable); // พิมพ์ค่าเป็น string

    variable = 10;        // พยายามเก็บ int
    printf("%d\n", (int)variable);  // พิมพ์ค่าเป็น int (จะเกิด warning)

    return 0;
}
```

```
warning: cast to pointer from integer of different size
```

เปรียบเทียบ Dynamic Typing กับภาษาอื่นๆ

2. Java (Static Typing)
ในภาษา Java จะคล้ายกับภาษา C ตรงที่ Java เป็นภาษาแบบ static typing ซึ่งทุกตัวแปรจะต้องระบุชนิดข้อมูล เช่น int, String และชนิดข้อมูลไม่สามารถเปลี่ยนได้ แต่ Java มีสิ่งที่เรียกว่า autoboxing ช่วยให้จัดการชนิดข้อมูลบางอย่างได้ง่ายขึ้น

ข้อดี : ตรวจสอบชนิดข้อมูลได้ตั้งแต่ compile-time ทำให้ระบบปลอดภัยและป้องกันการเกิดข้อผิดพลาด
ข้อเสีย : ขาดความยืดหยุ่นเมื่อทำงานกับข้อมูลที่ชนิดไม่แน่นอน

```java
public class DynamicTypingExample {
    public static void main(String[] args) {
        Object variable;           // ประกาศตัวแปรเป็น Object

        variable = "Hello";       // เก็บ String
        System.out.println(variable); // พิมพ์ค่าเป็น String

        variable = 10;            // พยายามเก็บ int
        System.out.println(variable); // พิมพ์ค่าเป็น int (จะเกิด warning)

        // เมื่อพยายามแคสต์กลับเป็น String จะเกิดข้อผิดพลาด
        String str = (String) variable; // จะเกิด ClassCastException ใน runtime
        System.out.println(str);
    }
}
```

```
Exception in thread "main" java.lang.ClassCastException: java.lang.Integer cannot
    at DynamicTypingExample.main(DynamicTypingExample.java:10)
```

เปรียบเทียบ Dynamic Typing กับภาษาอื่นๆ

3. Python (Dynamic Typing)
ใน Python ทุกตัวแปรเป็น dynamic typing โดยไม่ต้องระบุชนิดข้อมูลล่วงหน้า เราสามารถเปลี่ยนชนิดของตัวแปรได้ตลอดเวลา เช่น ตัวแปรสามารถเก็บทั้งตัวเลขและสตริงในเวลาต่างๆ

ข้อดี : เขียนโค้ดได้ยืดหยุ่น ไม่ต้องระบุชนิดข้อมูลตั้งแต่แรก ทำให้เขียนโค้ดได้เร็ว
ข้อเสีย : อาจทำให้เจอ bug ในช่วง runtime เพราะระบบจะตรวจชนิดข้อมูลเมื่อรันโปรแกรม

```python
# กำหนดตัวแปรเป็น string
variable = "Hello"
print(variable)  # Output: Hello

# เปลี่ยนตัวแปรเป็น int
variable = 10
print(variable)  # Output: 10

# เปลี่ยนตัวแปรเป็น list
variable = [1, 2, 3]
print(variable)  # Output: [1, 2, 3]

# เกิดข้อผิดพลาดเมื่อลองใช้ string operation กับตัวแปรที่เป็น list
print(variable + " World")  # พยายามรวม list กับ string จะเกิด error
```

```
Hello
10
[1, 2, 3]
Traceback (most recent call last):
  File "example.py", line 11, in <module>
    print(variable + " World")  # พยายามรวม list กับ string จะเกิด error
TypeError: can only concatenate list (not "str") to list
```

ข้อดี - ข้อเสียของ Dynamic Type

ข้อดี ของ Dynamic Type

1. ยืดหยุ่นสูง
2. ลดความซับซ้อน
3. เขียนโค้ดสั้นลง
4. เหมาะกับงานที่เน้น prototyping

ข้อเสีย ของ Dynamic Type

1. เสี่ยงต่อข้อผิดพลาด
2. ประสิทธิภาพอาจต่ำกว่า
3. บั๊กตรวจเจอยากขึ้น
4. อ่านโค้ดยากเมื่อโครงการใหญ่

ข้อควรระวังในการใช้งาน

1. ระวังเรื่องชนิดข้อมูลผิดพลาด
2. ปัญหาค้นหา bug
3. ประสิทธิภาพลดลง
4. อ่านโค้ดยากขึ้นเมื่อทีมใหญ่
5. ตรวจสอบชนิดข้อมูลให้ดี

THANK YOU

517321 PL