C# Syntactic Sugars

SPARCS #teaparty 2016-05-04 samjo

Today's Topics

- Warning: C# >= 6.0
- ref / out
- Implicit Type
- Property
- Extension Methods
- LINQ
- Async / Await
- Null Propagator / Conditional Operators

ref / out

- C#: 포인터 그런거 없다 ㅅㄱ (except unsafe)

- swap(int a, int b) => swap(ref int a, ref int b)
- ref vs out
  - ref는 반드시 초기화해야 함, out는 반드시 method 안에서 대입해야 함

```
static void Method(out int i)
{
    i = 44;
}
static void Main()
{
    int value;
    Method(out value);
    // value is now 44
}
```

```
bool result = Int32.TryParse(value, out number);
if (result)
{
   Console.WriteLine("Converted '{0}' to {1}.", value, number);
}
else
{
```

Implicit Type

- SomeVeryLongNameClass<Int, String, MyClass> foo = new SomeVeryLongNameClass<Int, String, MyClass>();

- ??? bar = obj.IForgotReturnTypeOfThisMethod();

Implicit Type

- var foo = new SomeVeryLongNameClass<Int, String, MyClass>();
- var bar = obj.IForgotReturnTypeOfThisMethod();

- Dynamically typed language 아님
- 모든 var로 선언된 변수의 type는 compile 시점에 계산

Property

- Class에 많은 수의 field가 필요함

```
public class Student {
    private int age;
    public int getAge() {
        return age;
    }
    public int setAge(int age) {
        if (age < 5)
            throw new Exception("넌 너무 젊은이야");
        this.age = age;
    }
}
me = new Student();
me.setAge(19);
int age = me.getAge();
```

Property

- “flexible mechanism to read, write, or compute the value of a private fields”

- get과 set로 구성
  - set이 없으면? => 외부에서는 readonly!

- set에서는 value라는 임시 변수를 생성

Property

```
public class Student {
    private int age;
    public int Age {
        get{
            return age;
        }
        set{
            if (value < 5)
                throw new Exception("넌 너무 젊은이야");
            age = value;
        }
    }
}
me = new Student();
me.Age = 19;
int age = me.Age;
```

```
public int Age { get; set; };
```

```
public string Name => First + " " + Last;
```

Extension Method

- 이미 있는 class에 추가하고 싶은 method가 있음
- 이름 바꿔서 상속한 다음에 거기다가 method를 추가

```
public class MyArrayList<T> extends ArrayList<T> {
    public T getMiddle() {
        ...
    }
}
MyArrayList<int> list = new MyArrayList<int>();
...
list.getMiddle();
```

Extension Methods

- “Add methods to existing types without creating a new derived type, recompiling, or otherwise …”

- 1. static class를 하나 만든다
- 2. static method를 하나 만든다
- 3. 첫번째 인자로 확장하고 싶은 type를 적는다
  - 이때 this를 앞에 붙인다

Extension Methods

```
public static class MyExtensions {
    public static T GetMiddle(this List<T> array) {
        ...
    }
}
List<int> list = new List<int>();
...
list.GetMiddle();
```

```
public static int WordCount(this String str)
{
    return str.Split(new char[] { ' ', '.', '?' },
                     StringSplitOptions.RemoveEmptyEntries).Length;
}
```

```
string s = "Hello Extension Methods";
int i = s.WordCount();
```

LINQ

- List<Customer> myCustomers = method();
- 런던에 살거나 이름이 SamJo인 사람만 뽑아줘봐

```
List<Customer> result = new List<Customer>();
for (Customer c in myCustomers) {
    if (c.City == "London" || c.Name == "SamJo") {
        result.Add(c);
    }
}
```

- 고객들을 사는 도시에 따라 묶고 2명 이상 사는 도시들만 추출해서 그걸 도시 이름 순서대로 정렬해봐
- ???

LINQ

- SQL on the LIST (IEnumerable)

```
var custQuery =
    from cust in customers
    group cust by cust.City into custGroup
    where custGroup.Count() > 2
    orderby custGroup.Key
    select custGroup;

var innerJoinQuery =
    from cust in customers
    join dist in distributors on cust.City equals dist.City
    select new { CustomerName = cust.Name, DistributorName = dist.Name };
```

LINQ + Extension Methods

- students.Where(s => s.Score.Calculus > 2.3).Sort(s => s.Name);

- Defined in Enumerable<T> class
  - Aggregate, All, Any, Average, Cast, Contact, Contains, Count, Distinct, FirstOrDefault, GroupBy, GroupJoin, Join, LastOrDefault, Max, Min, OrderBy, Reverse, Select, Sum, Where, …

- https://msdn.microsoft.com/en-us/library/system.linq.enumerable(v=vs.110).aspx

Async / Await

- Similar as Python
  - Return type SHOULD be Task, Task<TResult>, void

- Can use async keyword in event handler!

```
public async void StartButton_Click(object sender, RoutedEventArgs e) {
    Textbox.Text += "Start!!!";
    try {
        int length = await ExampleMethodAsync();
        Textbox.Text += length.ToString();
    }
    catch (Exception) {}
}

public async Task<int> ExampleMethodAsync() {
    var httpClient = new HttpClient();
    int r = (await httpClient.GetStringAsync("abc.com")).Length;
    return r;
}
```

Null Propagation Operator

- Object가 null일 수도 있고 아닐 수도 있음
- Null이 아닐 때는 object안의 특정 method의 return 값을 사용 / Null일 경우에는 기본값을 사용

```
if (obj == null)
    value = 0;
else
    value = obj.abc();

value = (obj == null) ? 0 : obj.abc();
```

Null Propagation Operator

- a?
  - a가 null이면 null
  - 아니면 계속 실행

- obj?.record?.method1()?.array?[0].age

- type: int?

Null Conditional Operator

- a ?? b
  - a가 null이면 b
  - 아니면 a

- int? a = b?.doit()
- int a = b?.doit()
- int a = b?.doit() ?? 0

Q&A

THANK YOU!