C# 4.0
Co- and Contravariance

Oliver Sturm
http://www.oliversturm.com

Outline

- **What is Variance?**
  - Some C# Variance features pre-4.0
- **Interface and Delegate Variance support**
  - The keywords “in” and “out”

(Co-) Variance

- A bird is an animal – Check
- A flock of birds is a group of animals – not generally true in C# 3.0!
- We expect to be able to view a bird as an animal under all circumstances, and C# 4.0 brings us closer to that ideal – variance is the feature responsible for this

- Variance makes things work the way we intuitively think they should

Variance, the theory

- **The types T and U can have these relationships:**
  - T is smaller/narrower than U
  - T is larger/wider than U
  - T is equal to U
  - T and U are independent
- **An operation that works with T and U and keeps the relationship the same is called covariant**
- **An operation that inverts the relationship is called contravariant**

Co- and Contravariance with delegates (C# 2.0)

```
Bird CreateBird( ) { … }
Func<Animal> function = CreateBird;
```

```
void TakeTiger(Tiger tiger) { … }
void TakeAnimal(Animal animal) { … }

Action<Mammal> action1 = TakeAnimal;
Action<Mammal> action2 = TakeTiger;
```

Variance: New in C# 4.0

```
var strings = new List<string> { "one", "two" };
IEnumerable<object> objects = strings;
```

- This wasn't possible before 4.0, to prevent calls like `objects.Add(42);`
- In C# 4.0 this is made possible by means of extensions to generic type parameters
  - The keyword "out" for covariance
  - "in" for contravariance
- These extensions work only with interfaces and delegates
- Standard .NET Framework interfaces and delegates have been extended with these keywords

Summary

- Variance is an important language feature that promotes intuitive use of types and type conversion
- C# has had certain variance features in the past
- In C# 4.0, extensions for generic type parameters add another level of variance support

References

- C# 4.0 Language Specification: http://osturm.me/cs40spec