T, EARL GREY, HOT

GENERICS IN .NET

Presented by Jeremy Clark
www.jeremybytes.com

DEFINITION

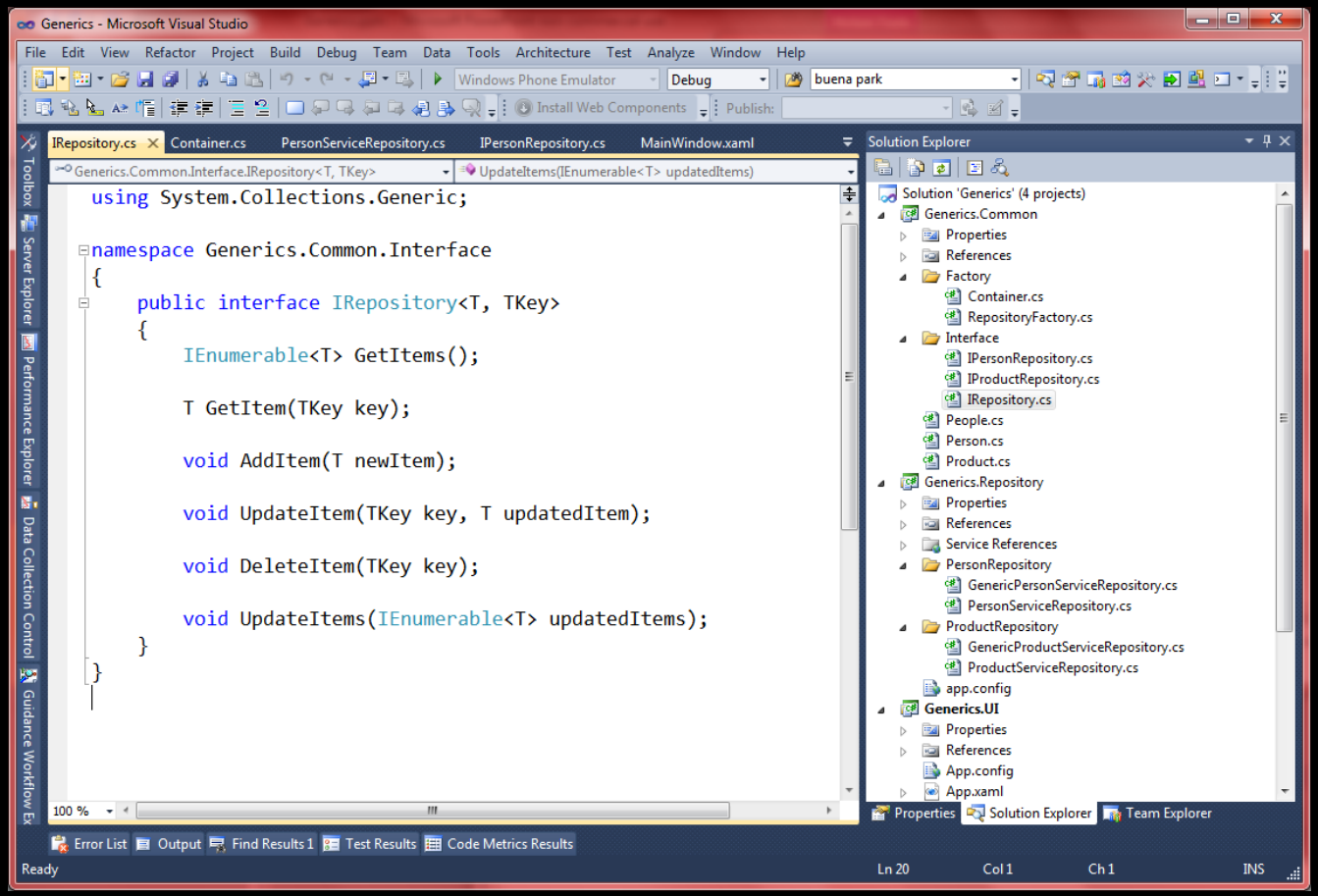
- Generics are classes, structures, interfaces, and methods that have placeholders (type parameters) for one or more of the types that they store or use.

- Example:
  - Definition: List<T>
    - Defines a collection (List) with a type parameter (T).
    - T can represent any type (class, struct, interface)
  - Usage: List<string>
    - Declares that the collection (List) will operate on type string.
    - List insert methods will prevent non-strings from being added.
    - List retrieval methods will return string objects.

BENEFITS

- Type Safety
  - Generics enforce compile-time type checking and eliminate the need for casting “object” to a more specific type.
- Performance
  - Generics prevent boxing / unboxing when using value types.
- Flexibility / Reuse
  - Generics allow us to use a single class, interface, or method with a variety of parameter and/or return types.

LOOK AT THE CODE



CONSTRAINTS

- Usage Example:

```
public static T Resolve<T>() where T : class
```

- Valid Constraints
  - `class` – reference type
  - `struct` – value type
  - `new()` – type with parameterless constructor
  - Name of base class
  - Name of interface

THANK YOU!

- MSDN
  - Generics (C# Programming Guide): http://msdn.microsoft.com/en-us/library/512aeb7t.aspx
- JeremyBytes
  - http://www.jeremybytes.com/Demos.aspx
  - Contains downloadable code and full walkthrough
  - Additional Sessions:
    - IEnumerable, ISaveable, IDontGetIt: Understanding .NET Interfaces
    - Dependency Injection: A Practical Introduction
  - Email: jeremy@jeremybytes.com
- Please rate this talk
  - http://www.speakerrate.com/speakers/10313
  - Link also available from the demo page on JeremyBytes.com