Explicit Interface Implementation

Jeremy Clark
DEVELOPER BETTERER

@jeremybytes www.jeremybytes.com

What & Why

Allow for more control

Resolve conflicting methods

IEnumerable<T> + IEnumerable

Standard Interface Implementation

```
public interface ISaveable {
  void Save();
}
```

```
public class Catalog : ISaveable
{
  public void Save()
  {
    Console.Write("Saved");
  }
}
```

```
Catalog catalog = new Catalog();

catalog.Save();
// "Saved"
```

```
ISaveable saveable = catalog;

saveable.Save();
// "Saved"
```

Explicit Interface Implementation

```
public interface ISaveable {
  void Save();
}
```

```
public class Catalog : ISaveable
{

  void ISaveable.Save()
  {
    Console.Write("Saved");
  }

}
```

```
Catalog catalog = new Catalog();

catalog.Save();
*** COMPILER ERROR ***



ISaveable saveable = catalog;

saveable.Save();
// "Saved"

(ISaveable(catalog)).Save();
// "Saved"
```

```
ISaveable saveable = new Catalog();
saveable.Save();
// "Saved"
```

◀ Interface type

```
Catalog catalog = new Catalog();
catalog.Save();
*** COMPILER ERROR ***
```

◀ Interface not used

```
var varCatalog = new Catalog();
varCatalog.Save();
*** COMPILER ERROR ***
```

◀ Interface not used
(same as using "Catalog" type)

```
((ISaveable)catalog).Save();
// "Saved"
```

◀ Interface type

Mixed Methods

```
public interface ISaveable {
  void Save();
}
```

```
public class Catalog : ISaveable
{
  public void Save()
  {
    Console.Write("Saved (catalog)");
  }

  void ISaveable.Save()
  {
    Console.Write("Saved (interface)");
  }
}
```

```
Catalog catalog = new Catalog();

catalog.Save();
// "Saved (catalog)"



ISaveable saveable = catalog;

saveable.Save();
// "Saved (interface)"

(ISaveable(catalog)).Save();
// "Saved (interface)"
```

Conflicting Method Signatures

```
public interface ISaveable {
  void Save();
}
```

```
public interface IDbSaver {
  string Save();
}
```

```
public class Catalog : ISaveable, IDbSaver
{
  public void Save()       // Catalog & ISaveable
  {
    Console.Write("Saved from ISaveable interface");
  }
  string IDbSaver.Save()  // IDbSaver (explicit)
  {
    return "Saved from IDbSaver interface";
  }
}
```

Another Explicit Implementation

```
public interface ISaveable {
  void Save();
}
```

```
public interface IDbSaver {
  string Save();
}
```

```
public class Catalog : ISaveable, IDbSaver
{
  void ISaveable.Save()    // ISaveable (explicit)
  {
    Console.Write("Saved from ISaveable interface");
  }
  public string Save()     // Catalog & IDbSaver
  {
    return "Saved from IDbSaver interface";
  }
}
```

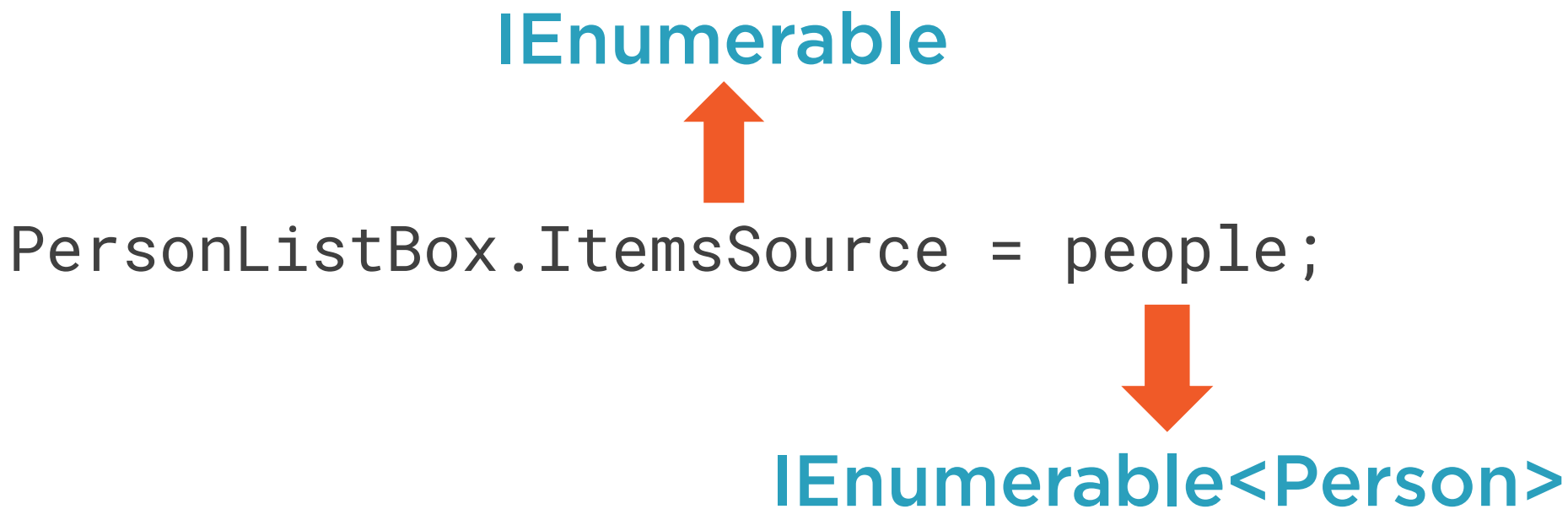
Both Explicitly Implemented

```
public interface ISaveable {
  void Save();
}
```

```
public interface IDbSaver {
  string Save();
}
```

```
public class Catalog : ISaveable, IDbSaver
{
  void ISaveable.Save()   // ISaveable (explicit)
  {
    Console.Write("Saved from ISaveable interface");
  }
  string IDbSaver.Save()  // IDbSaver (explicit)
  {
    return "Saved from IDbSaver interface";
  }
}
```

Type Mismatch?



```
public interface IEnumerable<T> : IEnumerable
```

Interface Inheritance

IEnumerable<T> includes all members from IEnumerable

No Type Mismatch

IEnumerable

```
PersonListBox.ItemsSource = people;
```

IEnumerable<Person>

+

IEnumerable

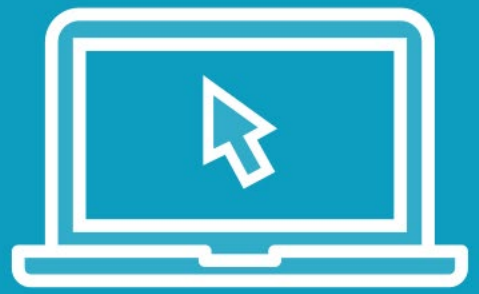
IEnumerable Members

```
public interface IEnumerable
{
    IEnumerator GetEnumerator();
}
```

Conflicting Signatures

```
public interface IEnumerable<T>
{
    IEnumerator<T> GetEnumerator();
}
```

Demo



Explicit interface implementation

- IEnumerable
- IEnumerable<T>

What & Why

Allow for more control

Resolve conflicting methods

IEnumerable<T> + IEnumerable