

Explicit Interface Implementation



Jeremy Clark

Developer Betterer

@jeremybytes www.jeremybytes.com



What & Why



Limit how interface members are used

Resolve conflicting methods

`IEnumerable<T> + IEnumerable`

Preparation for default implementation



Standard Interface Implementation

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

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

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



Explicit Implementation



Implementation belongs to the interface (not the class)

Can only be accessed using the interface type

No access modifiers



Explicitly Implemented Interface Member

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

**No access
modifier**

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

Uses the interface name



Calling an Explicitly Implemented Member

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

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

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

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



Additional Resources



<https://bit.ly/3tYeAee>

[https://github.com/jeremybytes/
csharp-interfaces-resources](https://github.com/jeremybytes/csharp-interfaces-resources)

Includes how to run the samples with Visual Studio Code



Demo



Explicitly implement an interface

Use the explicit implementation




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

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

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



Why?

You probably will not need it.

Exception: conflicting method signatures.



Conflicting Method Signatures

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

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

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



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

Interface Inheritance

IEnumerable<T> includes all members from IEnumerable

IEnumerable Members

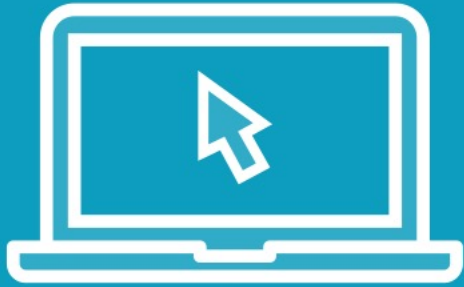
```
public interface IEnumerable
{
    IEnumerator GetEnumerator();
}
```

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

Conflicting Signatures



Demo



Explicit interface implementation

IEnumerable

IEnumerable<T>



What & Why



Limit how interface members are used

Resolve conflicting methods

IEnumerable<T> + IEnumerable

Preparation for default implementation

