



Harsha Vardhan

- › UI Expert
- › .NET Expert
- › Module Lead
- › Corporate Instructor

 Harsha  
Web University

## Namespaces

### Agenda

- 1 Understanding Namespaces  
What is namespace and its features.
- 2 Nested Namespaces  
Inner-namespace in outer-namespace.
- 3 Importing Namespaces ('using' directive)  
Importing namespaces at top of the file.
- 4 Using Alias  
Creating alias names for namespaces.

### Agenda

- 5 Using Static  
Importing static classes.  
New feature in C# 6.0

In real time projects, the main goal of namespaces is to group of all classes and other types as a single unit that are meant for specific purpose.

Introducing Namespaces

2. Creating Namespaces

Next: Nested Namespaces

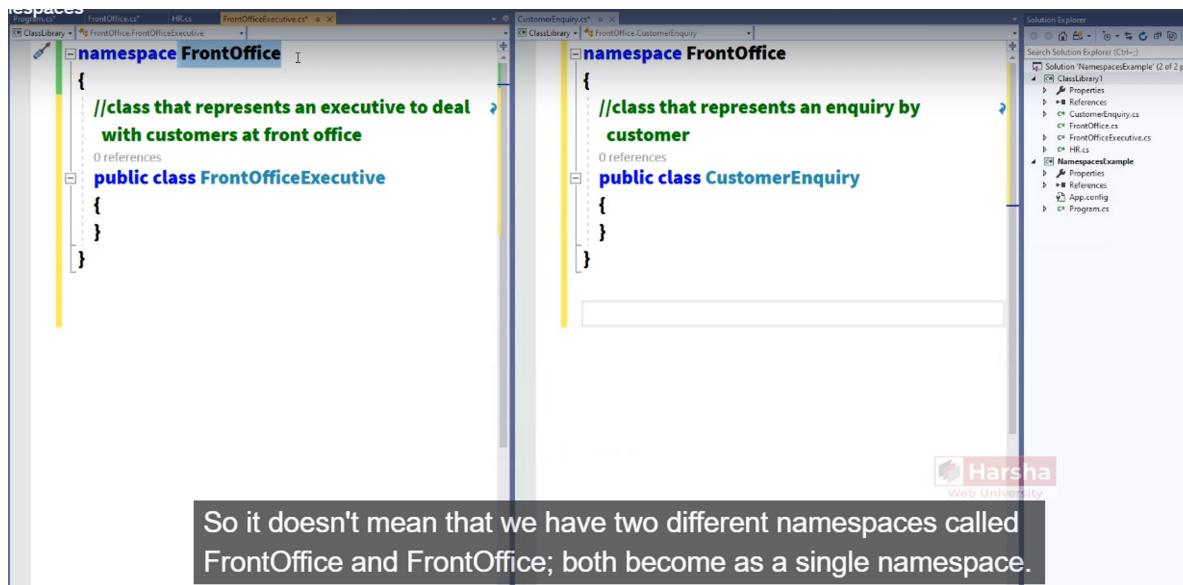
 Namespaces is a collection of classes and "other types such as interfaces, structures, delegate types, enumerations).

**Example**

Namespaces

```
namespace NamespaceName
{
    Classes
    Interfaces
    Structures
    Delegate Types
    Enumerations
}
```

In practical projects, it is recommended to write every interface and every class as a separate file.



```
FrontOfficeExecutive.cs
namespace FrontOffice
{
    //class that represents an executive to deal
    //with customers at front office
    public class FrontOfficeExecutive
    {
    }
}

CustomerEnquiry.cs
namespace FrontOffice
{
    //class that represents an enquiry by
    //customer
    public class CustomerEnquiry
    {
    }
}
```

Solution Explorer:

- ClassLibrary1
  - Properties
  - CustomerEnquiry.cs
  - FrontOffice.cs
  - FrontOfficeExecutive.cs
  - HR.cs
- NamespaceExample
  - Properties
  - App.config
  - References
  - Program.cs

Harsha  
Web University

So it doesn't mean that we have two different namespaces called FrontOffice and FrontOffice; both become as a single namespace.

So inside the same namespace, both classes will exist.

```

ClassLibrary1 -> FrontOffice.FrontOfficeExecutive
namespace FrontOffice
{
    //class that represents an executive to deal
    //with customers at front office
    0 references
    public class FrontOfficeExecutive
    {
    }
}

ClassLibrary2 -> FrontOffice.CustomerEnquiry
namespace FrontOffice
{
    //class that represents an enquiry by
    //customer
    0 references
    public class CustomerEnquiry
    {
    }
}

```

**FrontOffice**  
- **FrontOfficeExecutive**  
- **CustomerEnquiry**



Introducing Namespaces
Next: Nested Namespaces

> Namespaces is a collection of classes and "other types such as interfaces, structures, delegate types, enumerations".

**Example**

```

namespace FrontOffice
{
}

namespace HR
{
}

```

```

namespace Finance
{
}

namespace Inventory
{
}

```

**Namespaces**

```

namespace NamespaceName
{
    Classes
    Interfaces
    Structures
    Delegate Types
    Enumerations
}

```

### Goal

- To group-up classes and other types that are related to a particular project-module, into an unit.

Accessing a type that is present inside the namespace

- Syntax: NamespaceName.TypeName



### Agenda

1

#### Understanding Namespaces

What is namespace and its features.

2

#### Nested Namespaces

Inner-namespace in outer-namespace.

3

#### Importing Namespaces ('using' directive)

Importing namespaces at top of the file.

4

#### Using Alias

Creating alias names for namespaces.



**What**

- The namespace which is declared inside another namespace is called as "Nested namespace" or "Inner Namespace".

**Goal****Accessing types**

## Nested Namespaces

```
namespace OuterNamespace
{
    Classes
    Interfaces
    Structures
    Delegate Types
    Enumerations

    namespace InnerNamespace
    {
        Classes
        Interfaces
        Structures
        Delegate Types
        Enumerations
    }
}
```

If you have a large set of classes inside the same namespace; in order to subgroup them; that means a small unit of classes; we can use the nested namespace.

**What**

- The namespace which is declared inside another namespace is called as "Nested namespace" or "Inner Namespace".

**Goal**

- Use nested namespaces, in order to divide the classes of a larger namespace, into smaller groups.

**Accessing types**

- Syntax:  
OuterNamespace.InnerNamespace.TypeName

## Nested Namespaces

```
namespace OuterNamespace
{
    Classes
    Interfaces
    Structures
    Delegate Types
    Enumerations

    namespace InnerNamespace
    {
        Classes
        Interfaces
        Structures
        Delegate Types
        Enumerations
    }
}
```

## Agenda

1

### Understanding Namespaces

What is namespace and its features.

2

### Nested Namespaces

Inner-namespace in outer-namespace.

3

### Importing Namespaces ('using' directive)

Importing namespaces at top of the file.

4

### Using Alias

Creating alias names for namespaces.

114. Importing Namespaces

## Importing Namespaces ('using' Directive)

Next: 'Using' Alias Name

### What

- The "using" is a directive statement (top-level statement) that should be placed at the top of the file, which specifies the namespace, from which you want to import all the classes and other types.

### Using Directive

`using NamespaceName;`



- When you import a namespace, you can directly access all of its classes and other types (but not inner namespaces).
- The "using directives" are written independently for every file.
- "One using directive" can import "one namespace" only.

## Agenda

1

### Understanding Namespaces

What is namespace and its features.

2

### Nested Namespaces

Inner-namespace in outer-namespace.

3

### Importing Namespaces ('using' directive)

Importing namespaces at top of the file.

4

### Using Alias

Creating alias names for namespaces.

## 'using' Alias Name

## Next: 'Using' Static

### What

- The "using alias" directive allows you to create "alias name" for the namespace.

### "Using Alias" Directive

```
using AliasName = Namespacename;
```



The Alias names that are declared in one file, are not accessible or unknown to other files.

**What**

- The "using alias" directive allows you to create "alias name" for the namespace.

**"Using Alias" Directive**

```
using AliasName = Namespacename;
```



- Use "using alias" directive, if you want to access long namespaces with shortcut name.
- It is much useful to access specific namespace, when there is namespace name ambiguity (two classes with same name in two different namespaces and both namespaces are imported in the same file).

**Agenda**

5

**Using Static**

Importing static classes.

New feature in C# 6.0

**'using' static****What**

- The "using static" directive allows you import a static class directly from a namespace; so that you can directly access any of its methods anywhere in the current file.

**"Using Static" Directive**

```
using static Namespacename.StaticClassName;
```

```
public static void Beep(int frequency, int duration);
```

That means all the members of a static class is only static members, which includes Static Fields, Static Properties, Static Methods and Static Events.

```
public static void MoveBufferArea(int sourceLeft, int sourceTop, int sourceWidth, int sourceHeight, int targetLeft, int targetTop);
```

```
public static void SetConsoleColor(int left, int sourceTop, int sourceWidth, int sourceHeight, int targetLeft, int targetTop, char sourceChar, ConsoleColor sourceForeColor, ConsoleColor targetForeColor);
```

**What**

- › The "using static" directive allows you import a static class directly from a namespace; so that you can directly access any of its methods anywhere in the current file.

**"Using Static" Directive**

```
using static Namespacename.StaticClassName;
```



- › Use the "using static" directive to access methods of static class easily, without repeating the class name each time.