

# Namespace SocketServer

## Classes

### [ExampleClass](#)

Every class and member should have a one sentence summary describing its purpose.

### [Person](#)

This is an example of a positional record.

### [SocketServer](#)

Every class and member should have a one sentence summary describing its purpose.

# Class ExampleClass

Namespace: [SocketServer](#)

Assembly: SocketServer.dll








Every class and member should have a one sentence summary describing its purpose.

```
public class ExampleClass
```

## Inheritance

[object](#)  ← ExampleClass

## Inherited Members

[object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  ,  
[object.GetType\(\)](#)  , [object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  ,  
[object.ToString\(\)](#) 

## Remarks

You can expand on that one sentence summary to provide more information for readers. In this case, the `ExampleClass` provides different C# elements to show how you would add documentation comments for most elements in a typical class.

The remarks can add multiple paragraphs, so you can write detailed information for developers that use your work. You should add everything needed for readers to be successful. This class contains examples for the following:

<b>Summary</b>	This should provide a one sentence summary of the class or member.
<b>Remarks</b>	This is typically a more detailed description of the class or member
<b>para</b>	The para tag separates a section into multiple paragraphs
<b>list</b>	Provides a list of terms or elements
<b>returns, param</b>	Used to describe parameters and return values
<b>value</b>	Used to describe properties
<b>exception</b>	Used to describe exceptions that may be thrown
<b>c, cref, see, seealso</b>	These provide code style and links to other documentation elements

<b>example, code</b>	These are used for code examples
----------------------	----------------------------------

The list above uses the "table" style. You could also use the "bullet" or "number" style. Neither would typically use the "term" element.

Note: paragraphs are double spaced. Use the `*br*` tag for single spaced lines.

## Properties

### Label

```
public string? Label { get; set; }
```

#### Property Value

[string](#)

The `Label` property represents a label for this instance.

#### Remarks

The `Label` is a [string](#) that you use for a label.

Note that there isn't a way to provide a "cref" to each accessor, only to the property itself.

## Methods

### Add(int, int)

Adds two integers and returns the result.

```
public static int Add(int left, int right)
```

#### Parameters

`left` [int](#)

The left operand of the addition.

right [int](#)

The right operand of the addition.

## Returns

[int](#)

The sum of two integers.

## Examples

```
int c = Math.Add(4, 5);  
if (c > 10)  
{  
    Console.WriteLine(c);  
}
```

## Exceptions

[OverflowException](#)

Thrown when one parameter is [MaxValue](#) and the other is greater than 0. Note that here you can also use <https://learn.microsoft.com/dotnet/api/system.int32.maxvalue> to point a web page instead.

### See Also

[Label](#)

# Class Person

Namespace: [SocketServer](#)

Assembly: SocketServer.dll

This is an example of a positional record.

```
public record Person : IEquatable<Person>
```








## Inheritance

[object](#)  ← Person

## Implements

[IEquatable](#)  <[Person](#)>

## Inherited Members

[object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  ,  
[object.GetType\(\)](#)  , [object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  ,  
[object.ToString\(\)](#) 

## Remarks

There isn't a way to add XML comments for properties created for positional records, yet. The language design team is still considering what tags should be supported, and where. Currently, you can use the "param" tag to describe the parameters to the primary constructor.

## Constructors

### Person(string, string)

This is an example of a positional record.

```
public Person(string FirstName, string LastName)
```

## Parameters

**FirstName** [string](#) 

This tag will apply to the primary constructor parameter.

LastName [string](#)

This tag will apply to the primary constructor parameter.

## Remarks

There isn't a way to add XML comments for properties created for positional records, yet. The language design team is still considering what tags should be supported, and where. Currently, you can use the "param" tag to describe the parameters to the primary constructor.

## Properties

### FirstName

This tag will apply to the primary constructor parameter.

```
public string FirstName { get; init; }
```

### Property Value

[string](#)

### LastName

This tag will apply to the primary constructor parameter.

```
public string LastName { get; init; }
```

### Property Value

[string](#)

# Class SocketServer

Namespace: [SocketServer](#)

Assembly: SocketServer.dll

Every class and member should have a one sentence summary describing its purpose.

```
public class SocketServer : ISocketServer
```








## Inheritance

[object](#)  ← SocketServer

## Implements

[ISocketServer](#)

## Inherited Members

[object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  ,  
[object.GetType\(\)](#)  , [object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  ,  
[object.ToString\(\)](#) 

## Remarks

You can expand on that one sentence summary to provide more information for readers. In this case, the `ExampleClass` provides different C# elements to show how you would add documentation comments for most elements in a typical class.

The remarks can add multiple paragraphs, so you can write detailed information for developers that use your work. You should add everything needed for readers to be successful. This class contains examples for the following:

<b>Summary</b>	This should provide a one sentence summary of the class or member.
<b>Remarks</b>	This is typically a more detailed description of the class or member
<b>para</b>	The para tag separates a section into multiple paragraphs
<b>list</b>	Provides a list of terms or elements
<b>returns, param</b>	Used to describe parameters and return values
<b>value</b>	Used to describe properties

<b>exception</b>	Used to describe exceptions that may be thrown
<b>c, cref, see, seealso</b>	These provide code style and links to other documentation elements
<b>example, code</b>	These are used for code examples

The list above uses the "table" style. You could also use the "bullet" or "number" style. Neither would typically use the "term" element.

Note: paragraphs are double spaced. Use the `*br*` tag for single spaced lines.

## Constructors

### SocketServer(int)

```
public SocketServer(int port)
```

#### Parameters

port [int](#)

## Methods

### Start()

```
public void Start()
```

### Stop()

```
public void Stop()
```



# Namespace SocketServer.Interfaces

## Interfaces

[ISocketServer](#)

# Interface ISocketServer

Namespace: [SocketServer.Interfaces](#)

Assembly: SocketServer.dll

```
public interface ISocketServer
```

## Methods

### Start()

```
void Start()
```

### Stop()

```
void Stop()
```