**Array.Clone Method**

**.NET Framework 4**

Creates a shallow copy of the [Array](http://msdn.microsoft.com/en-us/library/system.array.aspx).

**Namespace:** [System](http://msdn.microsoft.com/en-us/library/system.aspx)  
**Assembly:** mscorlib (in mscorlib.dll)

A shallow copy of an [Array](http://msdn.microsoft.com/en-us/library/system.array.aspx) copies only the elements of the [Array](http://msdn.microsoft.com/en-us/library/system.array.aspx), whether they are reference types or value types, but it does not copy the objects that the references refer to. The references in the new [Array](http://msdn.microsoft.com/en-us/library/system.array.aspx) point to the same objects that the references in the original [Array](http://msdn.microsoft.com/en-us/library/system.array.aspx) point to.

In contrast, a deep copy of an [Array](http://msdn.microsoft.com/en-us/library/system.array.aspx) copies the elements and everything directly or indirectly referenced by the elements.

The clone is of the same [Type](http://msdn.microsoft.com/en-us/library/system.type.aspx) as the original [Array](http://msdn.microsoft.com/en-us/library/system.array.aspx).

This method is an O(n) operation, where n is [Length](http://msdn.microsoft.com/en-us/library/system.array.length.aspx).

#### Return Value

Type: [System.Object](http://msdn.microsoft.com/en-us/library/system.object.aspx)  
A shallow copy of the [Array](http://msdn.microsoft.com/en-us/library/system.array.aspx).

#### Implements

[ICloneable.Clone()](http://msdn.microsoft.com/en-us/library/system.icloneable.clone.aspx)

int[] Alpha = { 1, 2, 3, 4, 5 };

int[] Clone = (int[])Alpha.Clone();

Clone[2] = 0;

Console.WriteLine(Alpha[2]);//Display 3

Console.WriteLine(Clone[2]);//Display 0

int[] Beta = { 1, 2, 3, 4, 5 };

int[] Copy = Beta;

Copy[2] = 0;

Console.WriteLine(Beta[2]);//Display 0

Console.WriteLine(Copy[2]);//Display 0

using System;

class Number //struct Number

{

public int Integer;

}

class MainApp

{

static void Main()

{

Number Alpha = new Number();

Number Beta = new Number();

Alpha.Integer = 10;

Beta.Integer = 20;

Console.WriteLine("Alpha.Integer= {0}, Beta.Integer = {1}", Alpha.Integer, Beta.Integer);

Alpha = Beta;

Beta.Integer = 30;

Console.WriteLine("Alpha.Integer = {0}, Beta.Integer = {1}", Alpha.Integer, Beta.Integer);

Console.ReadKey();

}

}

/\*

When you assign one structure to another, a copy of the object is made. This is an

important way in which struct differs from class. When you assign one class reference

to another, you are simply making the reference on the left side of the assignment

refer to the same object as that referred to by the reference on the right.

When you assign one struct variable to another, you are making a copy of the object

on the right.

struct:

Alpha.Integer= 10, Beta.Integer = 20

Alpha.Integer = 20, Beta.Integer = 30

class:

Alpha.Integer= 10, Beta.Integer = 20

Alpha.Integer = 30, Beta.Integer = 30

\*/

}

**HttpSessionState.Timeout Property**

The Timeout property cannot be set to a value greater than 525,600 minutes (1 year). The default value is 20 minutes.

**ASP.NET\_SessionId** is a cookie that ASP.NET uses to store a unique identifier for your session. The session cookie is not persisted on your hard disk.

Session objects are created/retrieved during AcquireRequestState event. If you have HTTP Modules which intercept the request before this event, your session might not have been initialized.

The following code example shows how to set a past expiration date on a cookie.

if (Request.Cookies["UserSettings"] != null)

{

HttpCookie myCookie = new HttpCookie("UserSettings");

myCookie.Expires = DateTime.Now.AddDays(-1d);

Response.Cookies.Add(myCookie);

}

Web Services Framework

ASP.NET supports Web services using Windows Communication Foundation. A Web service is a component that contains business functionality that enables applications to exchange information across firewalls by using standards like HTTP and XML messaging. Web services are not tied to a particular component technology or object-calling convention. As a result, programs written in any language, using any component model, and running on any operating system can access Web services. For more information

**Self Join**

USE [master];

Go

--IF EXISTS(SELECT name FROM sys.databases WHERE name = N'Testdb')

--DROP DATABASE [Testdb];

--GO

-- OR

IF DB\_ID (N'Testdb') IS NOT NULL

DROP DATABASE [Testdb];

GO

CREATE DATABASE [Testdb];

GO

USE [Testdb];

GO

--IF EXISTS(SELECT name FROM sys.tables WHERE name = N'Test')

--DROP TABLE [Employee];

--GO

-- OR

IF OBJECT\_ID (N'Employee', N'U') IS NOT NULL

DROP TABLE [Employee];

GO

-- Create a Table

CREATE TABLE Employee

(

EmployeeID INT PRIMARY KEY,

Name NVARCHAR(50),

ManagerID INT

)

GO

-- Insert Sample Data

INSERT INTO Employee

SELECT 1, 'Mike', 3

UNION ALL

SELECT 2, 'David', 3

UNION ALL

SELECT 3, 'Roger', NULL

UNION ALL

SELECT 4, 'Marry',2

UNION ALL

SELECT 5, 'Joseph',2;

GO

/\* Inner and Outer Join as Self Join \*/

IF OBJECT\_ID('Employee', 'U') IS NOT NULL

DROP TABLE Employee

GO

-- Create a Table

CREATE TABLE Employee

(

EmployeeID INT PRIMARY KEY,

Name NVARCHAR(50),

ManagerID INT

)

GO

-- Insert Sample Data

INSERT INTO Employee

SELECT 1, 'Mike', 3

UNION ALL

SELECT 2, 'David', 3

UNION ALL

SELECT 3, 'Roger', NULL

UNION ALL

SELECT 4, 'Marry',2

UNION ALL

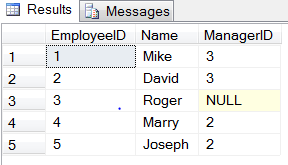
SELECT 5, 'Joseph',2;

GO

-- Check the data

SELECT \* FROM Employee;

GO



-- Inner Join as Self Join

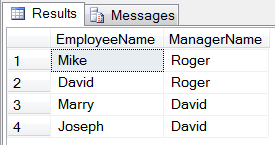
SELECT E1.Name AS EmployeeName, E2.Name AS ManagerName

FROM Employee E1

INNER JOIN Employee E2

ON E1.ManagerID = E2.EmployeeID;

GO



-- Outer Join as Self Join

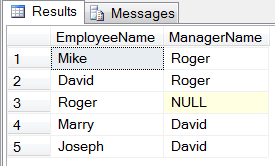
SELECT E1.Name AS EmployeeName, E2.Name AS ManagerName

FROM Employee E1

LEFT JOIN Employee E2

ON E1.ManagerID = E2.EmployeeID;

GO



-- Outer Join as Self Join

SELECT E1.Name AS EmployeeName, ISNULL(E2.Name, 'Top Manager') AS ManagerName

FROM Employee E1

LEFT JOIN Employee E2

ON E1.ManagerID = E2.EmployeeID;

GO

