http://www.dotnetfunda.com/interviews/faq/

**What do you mean by Boxing and Unboxing?**

[Page copy protected against web site content infringement by Copyscape](http://www.copyscape.com/)

Before explain about boxing and unboxing let take a  brief background of  different types of data types.

**There two types.**

Value Type

Reference Type

**Value Types:**

A value type’s variables contain the value itself. I mean it doesn’t contain a pointer to the object. It does not store the data into heap memory. Different value types are

* Integral Types (sbyte, byte, short, ushort, int, uint, long, ulong), bool type, char type,   
  Floating point types(flaot,double) and the decimal types. They are all aliases of the .NET System Types.
* Struct Types
* Enumeration Types

**Reference Types:**

A variable that contains reference or address of the actual data.   
Reference types are allocated on the managed heap. Different reference   
types are

* The Object Type
* The class Type
* Interfaces
* Delegates
* The string type
* Arrays

**Boxing And Unboxing:**

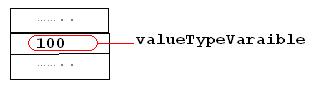
Converting a value type to reference type is called Boxing.

Unboxing is just opposite to boxing, that is Converting a reference type to value type. But here you have to specify explicitly in which value type you want to extract from the object.

Ex:

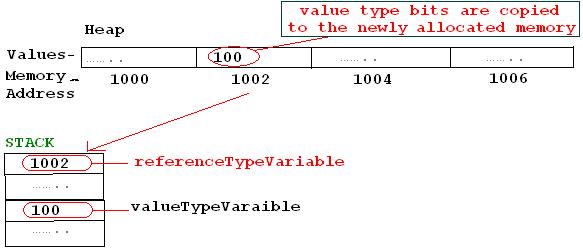
int32 valueTypeVaraible= 100; //value type variable

Memory Allocation into Stack

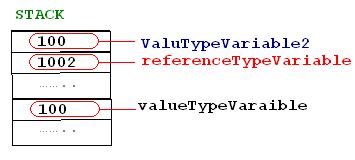
**STACK**  
  
  
  
 

object referenceTypeVariable= valueTypeVaraible;

Now the memory is allocated on the heap of size equal to size of the varaiable,the value type bits are copied to the newly allocated memory and the address of the object is returned and stored in reference type.This is basically called Boxing.

     
    
  
  
  
  
You can see from the above diagram *referenceTypeVariable* is storing the pointer of the heap address where the actual value has stored.

Int32 valueTypeVaraible2=(Int32) referenceTypeVariable ; //Unboxing

  
**Conclusion:**

* Value types are allocated on the thread's stack and they hold the actual value
* Reference types are allocated on the thread's stack and they hold a pointer to an object allocated in the managed heap

In both cases variables are allocated on the thread's stack, the difference being that a value type holds the actual value (for example, an integer, a struct), whereas the reference type holds a pointer to the memory area where the actual value is stored (for example, an instance of a StringBuilder class).

## What is Delegates and Events ?

[Page copy protected against web site content infringement by Copyscape](http://www.copyscape.com/)

The word event and delegates always come together, but they are separate to each other. It is true that delegates are independent objects which each event needs to perform its callback.

#### Delegate:

Delegates can be regarded as function prototypes, which prototype one method so that an object of it can point to that function. In other words, delegates are types which maps a method signature, so that a variable of a delegate can refer a method of same signature.   
For instance,

delegate int MySampleDelegate(int x, int y);  
  public void add(int x, int y)  
  {  
     return x + y;  
  }  
  public void substract(int x, int y)  
  {  
     return x - y;  
  }

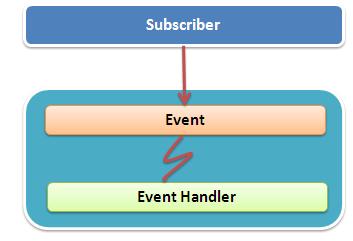
From the main function you can call a delegate using :

MySampleDelegate obj1 = new MySampleDelegate(add);  
MySampleDelegate obj2 = new MySampleDelegate(substract);  
Console.Write(obj1(10, 20)); // Prints 30  
Console.Write(obj2(40, 10)); //Prints 30

So the first object points to add method, while the second object to substract. You can call a delegate similar to how you call a method.

#### Events :

Events are callbacks from a class which invokes the associated delegate directly when the event is executed. In case of event driven approach, there is a subscriber of an event. Event is subscribed from outside using an eventhandler, which is a method declared externally and can be called back whenever the class wants to invoke.   
  
Thus you can say *Events are generated from class itself which calls a delegate*. The delegate if subscribed from outside will call the actual method.



## What are the ways to call a Delegate ?

[Page copy protected against web site content infringement by Copyscape](http://www.copyscape.com/)

There are quite a few ways a delegate may be called. Say for instance, I may call it as we do as normal method, I can make Asynchronous to the method, serialize a delegate, clone a delegate etc.

#### Synchronous call :

MySampleDelegate obj1 = new MySampleDelegate(add);  
obj1.Invoke(19, 30); // obj1(19,30)

#### Asynchronous Call :

obj1.BeginInvoke(19, 30, new AsyncCallback(asynchcallback), null);

Where asynchcallback represents the callback method which will be called automatically with an object of AsyncResult from which we can get the methods.

# How many generations does garbage collector have?

There are 3 generations in garbage collector generation 0, 1 and 2. Generation 0 are fresh created objects, generation 1 are objects which are bit older than gen 0 and gen 2 are the oldest objects

# What is reflection?

Reflection helps us to browse methods, properties and function of a given class or assembly on run time. You can also invoke the methods and functions using reflection. Below is a simple sample code where we are browsing in run time through methods and function of “MyClass”.

MyClass objMyClass = new MyClass();

// Get the class type

Type parameterType = objMyClass.GetType();

string name = parameterType.Name;

// Browse through members

foreach (MemberInfo objMemberInfo in parameterType.GetMembers())

{

Console.WriteLine(objMemberInfo.Name);

}

// Browse through properties.

foreach (PropertyInfo objPropertyInfo in parameterType.GetProperties())

{

Console.WriteLine(objPropertyInfo.Name);

}

If you want to invoke method and function you need to use the invoke member function as shown in the below code.

parameterType.InvokeMember("Display",BindingFlags.Public | BindingFlags.NonPublic | BindingFlags.InvokeMethod | BindingFlags.Instance,null, objMyClass, null);

**What is trigger and different types of Triggers?**

Trigger is a SQL server code, which execute when a kind of action on a table occurs like insert, update and delete. It is a database object which is bound to a table and execute automatically.

Triggers are basically of two type’s namely "**After Triggers**" and "**Instead of Triggers**".  
**1.After Triggers:-** this trigger occurs after when an insert, update and delete operation has been performed on a table.

“After Triggers” are further divided into three types  
AFTER INSERT Trigger.  
AFTER UPDATE Trigger.  
AFTER DELETE Trigger.

Let us consider that we have the following two tables.

Create “Customer” table with the following field as you see in the below table.

|  |  |  |  |
| --- | --- | --- | --- |
| Cust\_ID | Cust\_Code | Cust\_Name | Cust\_Salary |
| 1 | A-31 | Moosa | 4500 |
| 2 | A-09 | Feroz | 5000 |
| 3 | A-16 | Wasim | 4000 |

Create “Customer\_Audit” table with the following field as you see in the below table.

|  |  |  |  |
| --- | --- | --- | --- |
| Cust\_ID | Cust\_Name | Operation\_Performed | Date\_Time |

The main purpose of creating “Customer\_Audit” table is to record the data which occurs on the “Customer” table with their respective operation and date-time.

Let’s begin with “**After Insert Trigger**”:- This trigger fire after an insert operation performed on a table.  
Let us see the example of “After Insert Trigger” for better understanding.

**Query:-**Create Trigger TrigInsert on Customer  
For insert as   
declare @Cust\_ID int;  
declare @Cust\_Name varchar(100);  
declare @Operation\_Performed varchar(100);  
select @Cust\_ID=i.Cust\_ID from inserted i;   
select @Cust\_Name=i.Cust\_Name from inserted i;   
set @Operation\_Performed='Inserted Record -- After Insert Trigger';  
insert into Customer\_Audit  
(Cust\_ID,Cust\_Name,Operation\_Performed,Date\_Time)   
values(@Cust\_ID,@Cust\_Name,@Operation\_Performed,getdate());  
PRINT 'AFTER INSERT trigger fired.'

Now, insert a record into Customer table:

**Query:-** insert into Customer values ('A-10','Danish')

Once the insert statement is successfully done, the record is inserted into the “Customer” table and the “After Trigger” (TrigInsert) is fired and the same record is also stored into “Cutomer\_Audit” table.

To see the record in “Customer\_Audit” table write query as below:-

**Query:-** select \* from Customer\_Audit

|  |  |  |  |
| --- | --- | --- | --- |
| Cust\_ID | Cust\_Name | Operation\_Performed | Date\_Time |
| 4 | Danish | Inserted Record -- After Insert Trigger | 2011-04-06 19:46:56.390 |

You can see that the same record is seen in the “Customer\_Audit” table with Operation\_performed and the date\_time when it was updated.

Now let’s see for “**After Update Trigger**”:-This trigger fire after an update operation performed on a table.  
Let us see the example of “After Update Trigger” for better understanding.

**Query:-** Create trigger TrigUpdate on Customer  
For Update as   
declare @Cust\_ID int;  
declare @Cust\_Name varchar(100);  
declare @Operation\_Performed varchar(100);  
select @Cust\_ID=i.Cust\_ID from inserted i;   
select @Cust\_Name=i.Cust\_Name from inserted i;   
set @Operation\_performed='Inserted Record -- After Insert';

if update(Cust\_Name)  
set @Operation\_Performed='Updated Record -- After Update Trigger.';  
insert into Customer\_Audit  
(Cust\_ID,Cust\_Name,Operation\_Performed,Date\_Time)   
values(@Cust\_ID,@Cust\_Name,@Operation\_Performed,getdate())

PRINT 'AFTER UPDATE Trigger fired.'

Now, update a record into “Customer” table:-

**Query:-** update Customer set Cust\_Name = 'Khan Wasim' where Cust\_Code like 'A-16'

The record is updated into the Customer table and the TrigUpdate is fired and it stores a record into   
“Cutomer\_audit” table.

To see the record Customer\_Audit table write query for that.

**Query:-** select \* from Customer\_Audit

|  |  |  |  |
| --- | --- | --- | --- |
| Cust\_ID | Cust\_Name | Operation\_Performed | Date\_Time |
| 4 | Danish | Inserted Record -- After Insert Trigger | 2011-04-06 19:46:56.390 |
| 3 | Khan Wasim | Updated Record -- After Update Trigger | 2011-04-06 20:03:05.367 |

Now for, “**After Delete Trigger**”:-This trigger fire after a delete operation performed on a table.

In a similar way, you can code “After Delete trigger” on the table.

**2.Instead of Triggers:-** this trigger fire before the DML operations occur, first inserted and deleted get flourished and then trigger fires

“Instead of Triggers” are further divided into three types  
Instead of INSERT Trigger.  
Instead of UPDATE Trigger.  
Instead of DELETE Trigger.

Let us see the example of “Instead of UPDATE Trigger” for better understanding.

**Query:-**  
CREATE TRIGGER trgInsteadOfUpdate ON Customer   
INSTEAD OF update  
AS  
declare @cust\_id int;  
declare @cust\_name varchar(100);  
declare @cust\_salary  int;  
select @cust\_id=d. Cust\_ID from deleted d;  
select @cust\_name=d. Cust\_Name from deleted d;  
select @cust\_salary =d.Cust\_Salary from deleted d;

BEGIN  
if(@cust\_salary >4500)  
begin  
RAISERROR('Cannot delete where salary > 4500',16,1);  
ROLLBACK;  
end  
else  
begin  
delete from Customer where Cust\_ID [=@cust\_id](mailto:=@cust_id);  
COMMIT;  
insert into Customer\_Audit(Cust\_ID,Cust\_Name,Cust\_Salary,Operation\_Performed,Date\_Time)  
values(@cust\_id,@cust\_name,@cust\_salary,'Updated -- Instead Of Updated Trigger.',getdate());  
PRINT 'Record Updated -- Instead Of  Updated Trigger.'  
end  
END  
Now, update a record into “Customer” table:-

**Query:-** update Customer set Cust\_Name = 'Khan Wasim' where Cust\_Code like 'A-09'

When you try to update Customer table it will raise an error as we have use Instead of Update trigger.

**Error:-** Server: Msg 50000, Level 16, State 1, Procedure trgInsteadOfUpdate, Line 15  
Cannot update where salary > 4500

In a similar way, you can code “**Instead Delete trigger**” and “**Instead Insert trigger**” on the table.

**What is the sequence in which ASP.NET page life cycle is executed ?**

The best way to remember this answer is by remembering the word SILVER.

S (Well this word is just to make up the word , you can just skip this.)  
I (Init)  
L (Load)  
V (Validate)  
E (Event)  
R (Render)

# What are Authentication, Authorization and it's different types?

**Authentication:**  Who the user is? or Authentication is process of Identifying the user is valid or not from the database.  
**Authorization:**    To Identify what kind of authority or rights does user has.

***Different Types:***

In ASP.NET there are three way to do Authentication and Authorization.

**1)Windows Authentication:**  
                                    In this methadology ASP.NET web pages will use local windows users and groups to authenticate and authorize resources.

**2)Forms Authentication:**                                   This is a cookie based authentication where user name and password stored on client machine as cookie files or they are sent to URL for every request. Form-based authentication presents the users with an HTML-based web page that prompts the user for credentials.In case browser doesnot support cookies then username and password passed via URL string for every request.

**3)Passport Authentication:**                                     Passport authentication is based on passport website provided by the microsoft.So when user logins with credentials it will be reached to the passport website(i.e. hotmail,devhood,windows live etc) where authentication will happen.If authentication is successful it will return a token to your website.

# What are the benefits of three tier architecture ?

This is a very popular [c# interview question](http://www.questpond.com).In 3 tier architecture / layer we divide the project in to 3 layers UI , Middle  and DAL.  Due the modular approach we have 2 big advantages

**Reusability** :- You can reuse the middle layer with different user interfaces like ASP.NET , windows etc. You can also reuse you DAL with different projects.

**Maintainability** :-  When we change in one layer due to the modular approach it does not have ripple effect on other layers. we have to do less amount of changes in other layer when we change logic of one layer.

# What is the use of private constructor ?

This is one of those [.](http://www.questpond.com)NET interview question which is asked from the perspective to test if you understand the importance of private constructors in a class.

When we declare a class constructor as private , we can not do 2 things:-

* We can not create a object of the class.
* We can not inherit the class.

Now the next question which the interviewer will ask , whats the use of such kind of class which can not be inherited neither instantiated.

Many times we do not want to create instances of certain classes like utility , common routine classes. Rather than calling as shown below

clsCommon common = new clsCommon();  
common.CheckDate("1/1/2011");

You would like to call it as

clsCommon.CheckDate("1/1/2011");

# What is Serialization in .NET ?

Serialization is a process by which we can save the state of the object by converting the object in to stream of bytes.These bytes can then be stored in database, files, memory etc.

Below is a simple code of serializing the object.

MyObject objObject = new MyObject();

objObject.Value = 100;

// Serialization using SoapFormatter

SoapFormatter formatter = new SoapFormatter();

Stream objFileStream = new FileStream("c:\\MyFile.xml", FileMode.Create, FileAccess.Write, FileShare.None);

formatter.Serialize(objFileStream, objObject);

objFileStream.Close();

# What is the advantage of using MVC pattern?

MVC is one of the most used architecture pattern in ASP.NET and this is one of those ASP.NET interview question to test that do you really understand the importance of model view controller.

It provides a clean separation of concerns between UI and model.

UI can be unit test thus automating UI testing.

Better reuse of views and model. You can have multiple views which can point to the same model and also vice versa.

Code is better organized.

# What is public, private, protected, internal and internal protected?

**public** - The members (Functions & Variables) declared as public can be accessed from anywhere.

**private** - Private members cannot be accessed from outside the class. This is the default access specifier for a member, i.e if you do not specify an access specifier for a member (variable or function), it will be  considered as private. Therefore, *string PhoneNumber*; is equivalent to private *string PhoneNumber*;

**protected** - Protected members can be accessed only from the child classes.

**internal** - it can be accessed only within the same assembly.

**internal Protected** - internal protected can be accessed within the same assembly as well as in derived class.

**What is Shadowing?**

Shadowing is a concept of altering the behaviour of the base class member. It basically replaces complete element of the parent class like method becomes a variable.

class Program

{

static void Main(string[] args)

{

class1 obj = new class1();

class2 obj1 = new class2();

obj1.i(); //here i is treated as method

Console.WriteLine(obj.i); //here i is treated as variable

Console.ReadLine();

}

}

class class1

{

public int i=2;

}

class class2 : class1

{

public void i()

{

Console.WriteLine("Hello World");

}

}

# What is an interface?

Interface is a contract that defines the signature of the functionality. So if a class is implementing a interface it says to the outer world, that it provides specific behavior. Example if a class is implementing ‘Idisposable’ interface that means it has a functionality to release unmanaged resources. Now external objects using this class know that it has contract by which it can dispose unused unmanaged objects.

* Single Class can implement multiple interfaces.
* If a class implements a interface then it has to provide implementation to all its methods.

Following code shows that one has the interface definition and other class implements the interface. Below is the source code “IInterface” is the interface and “ClsDosomething” implements the “IInterface”. This sample just displays a simple message box.

Public Interface IInterFace

Sub Do Something ()

End Interface

Public Class ClsDoSomething

Implements IInterFace

Public Sub DoSomething () Implements WindowsInterFace.IInterFace.DoSomething

MsgBox (“Interface implemented”)

End Sub

End Class

**Can you name some aggregate function is SQL Server?**

AVG , SUM, COUNT, MAX, MIN

# What are “GRANT” and “REVOKE’ statements?

GRANT statement grants rights to the objects (table). While revoke does the vice-versa of it, it removes rights from the object.

**Can you explain Scriptmanager control in Ajax?**

Scriptmanager control is the central heart of Ajax. They manage all the Ajax related objects on the page. Some of the core objectives of scriptmanager control are as follows:-

* Helps load core Ajax related script and library.
* Provides access to web services.
* ASP.NET authentication, role and profile services are loaded by scriptmanager control.
* Provided registration of server controls and behaviors.
* Enable full or partial rendering of a web page.
* Provide localization features.