**.Net Framework Part 4**

**22. What is mean by Satellite assembly?**

When we write a multilingual or multi-cultural application in .NET, and want to distribute the core application separately from the localized modules, the localized assemblies that modify the core application are called satellite assemblies.

**23. What is portable executable (PE)?**

The file format defining the structure that all executable files (EXE) and Dynamic Link Libraries (DLL) must use to allow them to be loaded and executed by Windows. PE is derived from the Microsoft Common Object File Format (COFF). The EXE and DLL files created using the .NET Framework obey the PE/COFF formats and also add additional header and data sections to the files that are only used by the CLR.

**24. What is mean by Garbage collection?**

Garbage collection is a CLR features used to automatically manages memory. CLR automatically release the objects which are not longer used or referenced. Developer who forget to release the dispose the objects will be cleared by GC. But it is not known when GC will be called by CLR to clean the memory. So better we can dispose the objects once it is used.

**25. What are the different levels of GC is available?**

Generation 0 , Generation 1, Generation 2

**26. How Garbage collector will get memory from OS?**

When execution engine starts, GC will initialize segment of memory for its operation. GC reserves memory in segment, each segment is 16MB. When we run out of segments we reserve a new segment. If a segment of memory is not in use, it will be deleted.

**27. What is mean by LOH?**

LOH-(Large Object Heap). If size of the object are very high(>64KB) then it will be stored in different segment of memory called as LOH. GC will treat the large objects differently from small objects.

**28. What are situations GC will be called?**

1. If user forcefully calls System.GC.Collect
2. System is in low memory situation
3. Memory allocation exceeds the Generation0 threshold

**29. What is mean by value type and Reference type?**

Value type- Value type stores their value directly to memory address. Value type's values are allocated on the stack memory.

Reference type - Reference type stores the reference to the value's memory address. Reference type values are allocated on head memory.

**30. What is mean by Boxing and Unboxing?**

Boxing - Converting value type variable to reference type is called as boxing

UnBoxing - Converting reference type variable to value type is called as unboxing

int vType = 35;

object rType;

//Boxing process

rType = vType;

//Unboxing process

vType =(int) rType;

**31. How will you decide when to use value type and reference type?**

All depends upon need.

**32. What is difference between System exception and Application exception?**

All exceptions are derived from the Exception base class. Where Exception class is derived from the Object class. Both System and Application exception are derived from exception class but it has difference between them. System exceptions are thrown by the CLR where as Application exceptions are thrown by Application.

|  |  |
| --- | --- |
| **System Exception** | **Application Exception** |
| System exceptions are thrown by CLR | Application exceptions are thrown by Application |
| E.g OutOfMemoryException, NullReferenceException,etc | E.g User defined exception are created to throw application's exception and user defined exceptions are derived from *ApplicationException* |

**33. What is Reflection?**

.Net compilers store metadata information(types defined) about the assemblies inside the assembly itself. Using this metadata we can load an assembly dynamically (at runtime), get information about its containing types, instantiate these types and call methods.

"Reflection" is a mechanism using which we can load an assembly dynamically and call its method. The System.Reflection is the root namespace that contains classes to implement the reflection. The Assembly class is the one which is used to represent the assembly in .Net environment.

Example:

static void Main(string[] args)

{

// Load an assembly from file

Assembly myAssembly = Assembly.LoadFrom("MyService.dll");

// Get the types contained in the assembly and print their names

Type[] types = myAssembly.GetTypes();

foreach (Type type in types)

{

Console.WriteLine(type.FullName);

//Get the members(methods) present inside each type

foreach (MemberInfo member in type.GetMembers())

{

Console.WriteLine(" "+member.Name);

}

}

Console.ReadLine();

}

OutPut:

