|  |  |
| --- | --- |
| **ViewBag** | **ViewState** |
| It serves only for one request | It serves for multiple request |
| The value of viewbag is not posted back | It is posted back |
| It does not need casting | It needs to be decoded and read back |
| It is introduced in MVC | It is a part of core Asp.Net |

|  |  |
| --- | --- |
| **ViewBag** | **ViewState** |
| It serves only for one request | It serves only for one request |
| It does not need casting | It needs casting |
| It does not need null checking before reading | It needs null checking before reading |
|  |  |

**What is component ?**

Component decorator allows you to mark the class as angular decorator and help in adding extra meta data to the class.

Components are the most basic building blocks in Angular Application.

|  |  |
| --- | --- |
| **MSSQL** | **Postgres Sql** |
| It is paid to get advance feature | It is completely open source |
| Compression feature is available in paid version | Compression is available free of cost |
| Needed to specify varchar or nvarchar as per the data | Unicode UTF-8 by default, no confusion of varchar or nvarchar |
| Works only on windows | Works on almost every windows. |

**What is Time Complixity ?**

It is the estimation of time required to execute a program.

**What is CSS Box Model ?**

It is box that wraps around every html content from inner to out it consist of Content, Padding, Border, Margin.

**What are new element in HTML 5 ?**

**New elements**

Header, footer, main, mark, menuItem, summary, time

**New Input Types**

Color, date, time, email, range

**New Graphics**

Canvas, svg

**New Media**

Audio, embed, track, video,track

**What is LESS ?**

Less is a Stylesheet Language ( CSS pre-processor ), meaning that it extends the CSS language, adding features that allow variables, mixins, functions and many other techniques that allow you to make CSS that is more maintainable, themeable and extendable. Less runs inside Node, in the browser and inside Rhino.

**What is SASS ?**

SASS ( Syntactically awesome language ) is a stylesheet programming language. You can use things like variables, nested rules, inline imports, functions etc. Sass works with ruby

**What is event.preventDefault ?**

It will prevent the default action of the component. Clicking on link element will navigate to new window, but preventDefault will stop it from happening. Similarly function of submit button is to post the form, preventDefault will stop the form from submitting to server

**What is event.stopPropogation ?**

It will prevent any parent event from execution.

**What is event.stopImmediatePropagation ?**

If there are multiple event attached to element than it will execute the current event and stop all other events.

**What is Generics ?**

It is the concept of seperating Business logic and datatypes. It allows us to write class or method that can accept any datatype.

public static void ObjectToXMLFile(object T, string FilePath)

{

string xmlData = string.Empty;

var sr = new XmlSerializer(T.GetType());

using (var file = new StreamWriter(FilePath))

{

sr.Serialize(file, T);

}

}

**Features of generics ?**

1. It helps to resuse code, type safety and performance.
2. You can create your own generic interfaces, classes, methods, events.
3. You may create generic classes constrained to enable access to methods on particular data types.
4. You may get information on the types used in a generic data type at run-time by means of reflection

**How to restrict data in generic function ?**

**Constraints in generics.**

There are four types of constraints

**where T: struct** value type constraint

**where T: class** reference type constraint

**where T: new()** default parameterless constraint

**where T: <interface name>** interface constraint

**What are Anonymous Function ?**

Anonymous method is a method without name. It provides the ability to implement delegates without writing the function.

public delegate void Del(string Mesg);

static void Main(string[] args)

{

Del obj = delegate (string Mesg) {

Console.WriteLine(Mesg);

obj("Hello world");

}

**Lambda Expression**

A lambda expression is an anonymous function that you can use to create delegates, It can be implemented without using keyword delegate.

public delegate void Del(string Mesg);

static void Main(string[] args)

{

Del obj = x => { Console.WriteLine(x); };

obj("Hello world");

}

**Difference between Lambda and Anonymous.**

1. Lambdas are type inference. You do not have to specify the input parameter, because compiler can infer the type based on lambda body.
2. Lamdas can be usefull in making expression trees where as it cannot be done in anonymous function

**Delegates**

1. Delegate is a function pointer
2. A method that is going to assign to delegate must have same signature as delegate.
3. Multiple methods can be assigned to the delegate using "+" operator. It is called multicast delegate.

**Unicast Delegate**

class Program

{

public delegate void Print(int value);

static void Main(string[] args)

{

// Print delegate points to PrintNumber

Print printDel = PrintNumber;

printDel(100000);

}

public static void PrintNumber(int num)

{

Console.WriteLine("Number: {0,-12:N0}", num);

}

}

**Multicast delegates**

class Program

{

public delegate void Print(int value);

static void Main(string[] args)

{

Print printDel = PrintNumber;

printDel += PrintHexadecimal;

printDel += PrintMoney;

printDel(100000);

}

public static void PrintNumber(int num)

{

Console.WriteLine("Number: {0,-12:N0}", num);

}

public static void PrintMoney(int money)

{

Console.WriteLine("Money: {0:C}", money);

}

public static void PrintHexadecimal(int dec)

{

Console.WriteLine("Hexadecimal: {0:X}", dec);

}

}

**Action, Func, Predicate**

Func, Action and predicate represent a method that can be passed a parameter without explicitly declaring a custom delegate.

**Func delegate** can be used when the input as well as output present, it can be declared while declaring func

Func<int, string> MyFuncVar = r => (3.14 \* r \* r).ToString();

var result = MyFuncVar(3);

**Action Delegate** can be used when there is only input but no return value.

Action<int> MyActionVar = r =>

{

var area = 3.14 \* r \* r;

Console.WriteLine(area);

};

MyActionVar(4);

**Predicate delegate** can be used when input can be of any type but return type is only bool. Func can replace predicate.

Predicate<string> MyPredicateVar = r => r.Length > 5;

var BoolResult = MyPredicateVar("abc");

**What is Collection ?**

1. Collection classes are specialized classes for data storage and retrieval.
2. These classes provide support for list, stack, queues and hash table.
3. Most classes implement the same interface
4. Collection serve purpose such as allocating memory, accessing item from a list based on index. These classes create collection of object

**There are two types of collection**

1. Non Generic
2. Generic

**Non Generic - works on object type**

Each element represent a value of a different type.

Array size is not fixed

Elements can be added removed during runtime.

**Generic - works on specific types**

Specific Type

Array size is not fixed

Elements can be added removed during runtime.

**Difference between ActionResult and ViewResult ?**

ActionResult have multiple return type whereas ViewResult have only view return type.

**Why do we use store procedure in the project ?**

1. Store procedure increases the performance because the database can optimize the data access plan used by procedure and cache it for subsequent reuse.
2. It result in easier maintainance because its always easy to change the store procedure than to change the hardcoded SQL.
3. It also reduces the network traffic because SQL statement can be executed in batches rather sending multiple request from client to database.
4. It adds an extra level of abstraction as the store procedure are completely seperated from implementation.

**Paging Query**

To find second largest number

SELECT EmployeeID FROM

( SELECT dense\_rank() OVER(ORDER BY EmployeeID desc) as RowNum, \* from Employees ) as EmpTable

where RowNum = 2