Indexer in c#

Introduction:

Indexer allows to access member of class as array. We can define indexer using this keyword, which indicates that we are creating indexer for current class so instance of current class will have access to member of class. Once we have define indexer for class we can access member of class with help of array access operator [].Indexer are similar to property but it allow to access member of class using [] operator. We can also overload indexer.

Why we need to use indexer?

Let us consider simple example

Program 1:

namespace StudentClass

{

public class student

{

Int StudId;

string StudName;

public student(int StudId,string StudName)

{

this.StudId=StudId;

this.StudName=StudName;

}

}

}

In this example, we have created simple student class with two member and parametrized constructor.Once we have created class and if it is public then we can consume that class from any other class or project.

Program 2:

namespace consumestudent

{

public static void Main()

{

student stu = new student(1,”Hitanshi”);

stu[0];//it is not accessible

}

}

In our second example, we have created consumestudent class and we are creating one object of student class. As Default, scope for all member of class is private we cannot access StudId and StudName outside class. Now, if we want to access them outside class then there are three different mechanism are available.

A) Declare member of class as public

If we declare member as public then anyone out of class can get or set value. As developer we loss control on member.

Therefore, we should never declare member as public. For example if you declare prize member as public then anyone outside class can set prize.

B) Use properties

Property allow restricted access. It allow only read, only write or read/write access. Property also allow to set validation on member. For example if we want to store age only if age is greater than 18 then we can set validation in properties before storing them into database.

C) Use Indexer

Indexer are like properties but they provide access to member using index value.

Syntax:

[<Modifiers>] <Type> this [<parameter list>]

Here we use modifier as public, type as object because it may return integer, string or any other valid data type. We use this keyword to specify that we are declaring indexer on current class. Parameter list can be any valid datatype. Mostly we use int and string as parameter list. It means we can access member of class using int and string.

Where indexer are used?

We can use indexer in three way.

1. Indexer are used in class.

Example:

Inside class, we can declare indexer as below. We can also overloaded indexer.

Program 1:

namespace StudentClass

{

public class student

{

Int StudId;

string StudName;

public student (int StudId, string StudName)

{

this.StudId=StudId;

this.StudName=StudName;

}

}

public object this[int index]

{

get

{

If (index==0)

return StudId;

elseif (index==1)

return StudName;

}

set

{

If (index==0)

StudId = (int)value;

Elseif (index==1)

StudName = (string)value;

}

public object this[string name]

{

get

{

If(name.ToUpper()==”STUDID”)

return StudId;

elseif(name.ToUpper()=”STUDNAME”)

return StudName;

}

set

{

If(name.ToUpper()==”STUDID”)

StudId=(int)value;

elseif(name.ToUpper()==”STUDNAME”)

StudName= (string)value;

}

}

}

Here value is implicit variable that provide access to value assigned by user. Data type of value will be same as indexer so here it will be object so we are converting it to integer or string. In integer indexer, it is not compulsory to start index with 0. We can also start our index with 1 but as class is behaving like virtual array we are starting our first index with 0 which will point to StudId. In our example, we have overloaded indexer using int and string. One of the biggest problem with string indexer is that c# is case sensitive. To solve this problem we have used ToUpper(). This code will work even if user enter StudId, studid etc. because we are converting name indexer to uppercase and checking it with uppercase.

namespace consumestudent

{

public static void Main()

{

student stu = new student(1,”Hitanshi”);

stu[1]=”HITANSHI”;

Console.WriteLine(“Student Id ”+stu[“Studid”]);

Console.WriteLine(“Student name”+stu[“Studname”]);

}

}

Output:

1 HITANSHI

1. To store or retrieve data from session state or application state we use indexer.

namespace program

{

protected void page\_Load(object sender,EventArgs a )

{

session[“data1”]=”Show data1”;

session[“data2”]=”Show data2”;

Response.WritLine(“Session1 stores”+ session[0].ToString());

Responser.WritLine(“Session2 stores”+ session[“data2”].ToString());

}

}

So here, we have created first session using “data1” string indexer but when we are printing data of first session we are using integral indexer. We have used 0 as index so it will pull the data that is present in first session key. Here we have used ToString() because Session object actually return an object data type because we can store anything into a session state but we know that it is string so we are going to convert it into string.

1. To retrieve data from specific column when we are looping through “SqlDataReader” object, we can use either the integral indexer or string indexer.

using(SqlConnection con=new SqlConnection(cs))

{

SqlCommand cmd=new SqlCommand(“select \* from employee”,con);

con.open();

SqlDataReader rd=cmd.ExecuteReader();

while(rd.Read())

{

Response.WritLine(“Id=”+ rd[0].ToString());

Response.WritLine(“Name=”+ rd[“name”].ToString());

}

}

As illustrated in example, we can get data from database with help of string or integer indexer. Therefore, in .net framework many classes are already using this indexer.