**Part 66 - C# Tutorial - Overloading indexers in c#**

In this video we will discuss about **Overloading indexers** in c#. Please watch [Part 64](http://csharp-video-tutorials.blogspot.com/2013/04/part-64-c-tutorial-how-and-where-are.html) and [Part 65](http://csharp-video-tutorials.blogspot.com/2013/04/part-65-c-tutorial-indexers-in-c.html) before proceeding. We will be modifying the example, that we discussed in [Part 65](http://csharp-video-tutorials.blogspot.com/2013/04/part-65-c-tutorial-indexers-in-c.html).

In [Part 65](http://csharp-video-tutorials.blogspot.com/2013/04/part-65-c-tutorial-indexers-in-c.html), we discussed about creating an indexer based on integer parameter.   
public string this[int employeeId]  
{  
    get  
    {  
        return listEmployees.  
            FirstOrDefault(x => x.EmployeeId == employeeId).Name;  
    }  
    set  
    {  
        listEmployees.  
            FirstOrDefault(x => x.EmployeeId == employeeId).Name = value;  
    }  
}  
  
**Now let us create another indexer based on a string parameter.**  
public string this[string gender]  
{  
    get  
    {  
        // Returns the total count of employees whose gender matches  
        // with the gender that is passed in.  
        return listEmployees.Count(x => x.Gender == gender).ToString();  
    }  
    set  
    {  
        // Changes the gender of all employees whose gender matches  
        // with the gender that is passed in.  
        foreach (Employee employee in listEmployees)  
        {  
            if (employee.Gender == gender)  
            {  
                employee.Gender = value;  
            }  
        }  
    }  
}  
  
Please note that, indexers can be overloaded based on the number and type of parameters.  
  
**Here is the complete code of Company class.**  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using System.Data;  
using System.Data.SqlClient;  
using System.Configuration;  
  
namespace Demo  
{  
    public class Employee  
    {  
        public int EmployeeId { get; set; }  
        public string Name { get; set; }  
        public string Gender { get; set; }  
    }  
  
    public class Company  
    {  
        private List<Employee> listEmployees;  
  
        public Company()  
        {  
            listEmployees = new List<Employee>();  
  
            listEmployees.Add(new Employee   
            { EmployeeId = 1, Name = "Mike", Gender = "Male" });  
            listEmployees.Add(new Employee   
            { EmployeeId = 2, Name = "Pam", Gender = "Female" });  
            listEmployees.Add(new Employee   
            { EmployeeId = 3, Name = "John", Gender = "Male" });  
            listEmployees.Add(new Employee   
            { EmployeeId = 4, Name = "Maxine", Gender = "Female" });  
            listEmployees.Add(new Employee   
            { EmployeeId = 5, Name = "Emiliy", Gender = "Female" });  
            listEmployees.Add(new Employee   
            { EmployeeId = 6, Name = "Scott", Gender = "Male" });  
            listEmployees.Add(new Employee   
            { EmployeeId = 7, Name = "Todd", Gender = "Male" });  
            listEmployees.Add(new Employee   
            { EmployeeId = 8, Name = "Ben", Gender = "Male" });  
        }  
  
        public string this[int employeeId]  
        {  
            get  
            {  
                return listEmployees.  
                    FirstOrDefault(x => x.EmployeeId == employeeId).Name;  
            }  
            set  
            {  
                listEmployees.  
                    FirstOrDefault(x => x.EmployeeId == employeeId).Name = value;  
            }  
        }  
  
        public string this[string gender]  
        {  
            get  
            {  
                return listEmployees.Count(x => x.Gender == gender).ToString();  
            }  
            set  
            {  
                foreach (Employee employee in listEmployees)  
                {  
                    if (employee.Gender == gender)  
                    {  
                        employee.Gender = value;  
                    }  
                }  
            }  
        }  
    }  
}  
  
Notice that the Company class has **2 indexers**. The first indexer has an **integer** (employeeId) parameter and the second indexer has got a **string** (gender) parameter.  
  
To test the string indexer, that we have just created, copy and paste the following code in **Page\_Load**() event of **WebForm1.aspx.cs**  
  
Company company = new Company();  
  
Response.Write("Before changing the Gender of all male employees to Female");  
Response.Write("<br/>");  
  
// Get accessor of string indexer is invoked to return the total  
// count of male employees  
Response.Write("Total Employees with Gender = Male:" + company["Male"]);  
Response.Write("<br/>");  
Response.Write("Total Employees with Gender = Female:" + company["Female"]);  
Response.Write("<br/>");  
Response.Write("<br/>");  
  
// Set accessor of string indexer is invoked to change the gender  
// all "Male" employees to "Female"  
company["Male"] = "Female";  
  
Response.Write("After changing the Gender of all male employees to Female");  
Response.Write("<br/>");  
Response.Write("Total Employees with Gender = Male:" + company["Male"]);  
Response.Write("<br/>");  
Response.Write("Total Employees with Gender = Female:" + company["Female"]);