**Multi-Threading**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Thread t1 = new Thread(Method1)**

**{**

**Name = "Thread1"**

**};**

**Thread t2 = new Thread(Method2)**

**{**

**Name = "Thread2"**

**};**

**t1.Start();**

**t2.Start();**

**Console.Read();**

**}**

**static void Method1()**

**{**

**Console.WriteLine("Method1 Ended using " + Thread.CurrentThread.Name);**

**}**

**static void Method2()**

**{**

**Console.WriteLine("Method3 Ended using " + Thread.CurrentThread.Name);**

**}**

**}**

**Asynchronous**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**SomeMethod();**

**Console.ReadKey();**

**}**

**public async static void SomeMethod()**

**{**

**Console.WriteLine("Some Method Started......");**

**await Task.Delay(TimeSpan.FromSeconds(10));**

**Console.WriteLine("Some Method End");**

**}**

**}**

**Parallel**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("C# Parallel For Loop");**

**//It will start from 1 until 10**

**Parallel.For(1, 11, number => {**

**Console.WriteLine(number);**

**});**

**Console.ReadLine();**

**}**

**}**

**Open Closed Principle**

**namespace OCPDemo**

**{**

**public interface IDiscountStrategy**

**{**

**double CalculateDiscount(double price);**

**}**

**public class RegularDiscount : IDiscountStrategy**

**{**

**public double CalculateDiscount(double price)**

**{**

**return price \* 0.1;**

**}**

**}**

**public class PremiumDiscount : IDiscountStrategy**

**{**

**public double CalculateDiscount(double price)**

**{**

**return price \* 0.3;**

**}**

**}**

**public class DiscountCalculator**

**{**

**private readonly IDiscountStrategy \_discountStrategy;**

**public DiscountCalculator(IDiscountStrategy discountStrategy)**

**{**

**\_discountStrategy = discountStrategy;**

**}**

**public double CalculateDiscount(double price)**

**{**

**return \_discountStrategy.CalculateDiscount(price);**

**}**

**}**

**public class Program**

**{**

**public static void Main()**

**{**

**var regularDiscount = new RegularDiscount();**

**var calculator = new DiscountCalculator(regularDiscount);**

**double discountedPrice = calculator.CalculateDiscount(100); // 10% discount applied**

**var premiumDiscount = new PremiumDiscount();**

**calculator = new DiscountCalculator(premiumDiscount);**

**discountedPrice = calculator.CalculateDiscount(100); // 30% discount applied**

**Console.ReadKey();**

**}**

**}**

**}**

**public class HomeController : Controller**

**{**

**public ActionResult Index()**

**{**

**EmployeeBusinessLayer employeeBL = new EmployeeBusinessLayer();**

**Employee employee = employeeBL.GetEmployeeDetails(101);**

**ViewBag.Header = "Employee Details";**

**return View(employee);**

**}**

**}**

**@model FirstMVCDemo.Models.Employee**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta name="viewport" content="width=device-width" />**

**<title>Page Title</title>**

**</head>**

**<body>**

**<h2>@ViewBag.Header</h2>**

**<table style="font-family:Arial">**

**<tr>**

**<td>Employee ID:</td>**

**<td>@Model.EmployeeId </td>**

**</tr>**

**<tr>**

**<td>Name:</td>**

**<td>@Model.Name</td>**

**</tr>**

**<tr>**

**<td>Gender:</td>**

**<td>@Model.Gender</td>**

**</tr>**

**</table>**

**</body>**

**</html>**

**public static void RegisterRoutes(RouteCollection routes)**

**{**

**routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");**

**routes.MapRoute(**

**name: "Default",**

**url: "{controller}/{action}/{id}",**

**defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }**

**constraints: new { id = @"\d+" } //Restriction for id**

**);**

**}**

**public static void RegisterRoutes(RouteCollection routes)**

**{**

**//Enabling attribute routing**

**routes.MapMvcAttributeRoutes();**

**}**

**[Route("students/{studentID}")]**

**<a href="@Url.RouteUrl("mymenu")">Main menu</a>**

**<a href='@Url.Action("Action", "Controller")' class="btn btn-primary">Click Me</a>**

**@Html.ActionLink("Click Me", "Action", "Controller", null, new { @class = "btn btn-primary" })**

@model IEnumerable<CRUD\_OperationsInMVC.Models.Employee>

**<ul>**

**@foreach (Employee employee in @Model)**

**{**

**<li>@Html.ActionLink(employee.Name, "Details", new { id = employee.EmployeeId })</li>**

**}**

**</ul>**

@Html.Partial("ProductDetails", Model)

public class LogCustomExceptionFilter : FilterAttribute, IExceptionFilter

{

public void OnException(ExceptionContext filterContext)

{

if (!filterContext.ExceptionHandled)

{

var exceptionMessage = filterContext.Exception.Message;

var stackTrace = filterContext.Exception.StackTrace;

var controllerName = filterContext.RouteData.Values["controller"].ToString();

var actionName = filterContext.RouteData.Values["action"].ToString();

string Message = "Date :" + DateTime.Now.ToString() + ", Controller: " + controllerName + ", Action:" + actionName +

"Error Message : " + exceptionMessage

+ Environment.NewLine + "Stack Trace : " + stackTrace;

File.AppendAllText(HttpContext.Current.Server.MapPath("~/Log/Log.txt"), Message);

filterContext.ExceptionHandled = true;

filterContext.Result = new ViewResult()

{

ViewName = "Error"

};

}

}

##### **How to Register Custom Exception Filter in ASP.NET. MVC**

public class FilterConfig

{

public static void RegisterGlobalFilters(GlobalFilterCollection filters)

{

filters.Add(new LogCustomExceptionFilter());

}

}

protected void Application\_Start()

{

AreaRegistration.RegisterAllAreas();

RouteConfig.RegisterRoutes(RouteTable.Routes);

//calling RegisterGlobalFilters to register filters globally

FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);

}