* In ASP.NET MVC, a user request is routed to the appropriate controller and action method. However, there may be circumstances where you want to execute some logic before or after an action method executes. ASP.NET MVC provides filters for this purpose.
* ASP.NET MVC Filter is a custom class where you can write custom logic to execute before or after an action method executes. Filters can be applied to an action method or controller in a declarative or programmatic way. Declarative means by applying a filter attribute to an action method or controller class and programmatic means by implementing a corresponding interface
* MVC provides different types of filters. The following table list filter types, built-in filters, and interface that must be implemented to create custom filters.

|  |  |  |  |
| --- | --- | --- | --- |
| Authorization filters | Performs authentication and authorizes before executing an action method. | [Authorize], [RequireHttps] | IAuthorizationFilter |
| Action filters | Performs some operation before and after an action method executes. |  | IActionFilter |
| Result filters | Performs some operation before or after the execution of the view. | [OutputCache] | IResultFilter |
| Exception filters | Performs some operation if there is an unhandled exception thrown during the execution of the ASP.NET MVC pipeline. | [HandleError] | IExceptionFilter |

Example :-

[Authorize] (**Authorization filters**)

[HandleError] (**Exception filters**)

//This filter declare on controller level ,it means this attribute Aplicable on All method in this controller)

//(For Authroziation Given Defalut Login Url Path In web.config File

<authentication mode="Forms">

<forms loginUrl="Account/Login">

</forms>

</authentication>

public class HomeController : Controller

{

Student\_DBEntities DB = new Student\_DBEntities();

[AllowAnonymous] (This Attribute Used For Access Allow Anonymous User)

public ActionResult About()

{

return View();

}

[AllowAnonymous]

[OutputCache(Duration = 30)]

public ActionResult Contact()

{

return View();

}

[AllowAnonymous]

public ActionResult Index()

{

return View();

}

public async Task<ActionResult> getList()

{

List<Student\_Details\_Sundram> data = await DBaccess.GetAllStudentsAsync();

return View(data);

}

[ChildActionOnly]

public String TotaltNoOfstudent()

{

String total = "Total No Of Students:-" + "\t" + DB.Student\_Details\_Sundram.Count().ToString();

return total;

}

[HttpGet]

public ActionResult Create()

{

return View();

}

[HttpPost]

public async Task<ActionResult> Create(Student\_Details\_Sundram studentdetails)

{

bool Data = await DBaccess.CreateAsync(studentdetails);

if (Data)

{

return RedirectToAction("getList");

}

else

{

throw new Exception();

}

}

public async Task<ActionResult> Details(int ID)

{

var data = await DBaccess.GetDetailsAsync(ID);

return View(data);

}

public async Task<ActionResult> Delete(int ID)

{

var Data = await DBaccess.DeleteAsync(ID);

if (Data)

{

return RedirectToAction("getList");

}

else

{

throw new Exception();

}

}

[HttpGet]

public async Task<ActionResult> Edit(int ID)

{

var Data = await DBaccess.EditAsync(ID);

return View(Data);

}

[HttpPost]

public async Task<ActionResult> Edit(Student\_Details\_Sundram editStudentobj)

{

bool Data = await DBaccess.EditAsync(editStudentobj);

if (Data)

{

return RedirectToAction("getList");

}

else

{

throw new Exception();

}

}

[HttpGet]

public async Task<ActionResult> UpdatePassWord()

{

return View();

}

[HttpPost]

public async Task<ActionResult> UpdatePassWord(Login\_SignUp\_UserDetails model)

{

String ID\_value = Session["ID"].ToString();

bool Data = await DBaccess.UpdatePassWord(model, ID\_value);

if (Data)

{

return RedirectToAction("Login","Account");

}

else

{

throw new Exception();

}

}

[HttpGet]

public async Task<ActionResult> EditUserProfile()

{

String ID\_value = Session["ID"].ToString();

int ID\_Value\_convert = Int32.Parse(ID\_value);

var Data = await DBaccess.EditUserProfileAsync(ID\_Value\_convert);

return View(Data);

}

[HttpPost]

public async Task<ActionResult> EditUserProfile(Login\_SignUp\_UserDetails model)

{

bool Data = await DBaccess.EditUserProfileAsync(model);

if(Data)

{

FormsAuthentication.SignOut();

return RedirectToAction("Index");

}

else

{

return View("Error");

}

}

}

}