2.6 Using Custom Filters



This section will guide you to:

* Apply and work with custom filters on the html pages.

This guide has eleven subsections, namely:

2.6.1 Creating an ASP.NET MVC project to use custom filters

2.6.2 Creating a Login view to show an account login form

2.6.3 Creating StudentView.cshtml to display student data without authorization

2.6.4 Creating StudentViewSecure.cshtml to display student data with authorization

2.6.5 Creating AccountController to handle the login page

2.6.6 Creating ActionLoggingFilter to log all filter operations

2.6.7 Making changes to Index.cshtml to show custom filter options

2.6.8 Changing HomeController to use the views for custom filters

2.6.9 Building the project

2.6.10 Publishing and running the project

2.6.11 Pushing the code to your GitHub repositories

**Step 2.6.1:** Creating an ASP.NET MVC project to use custom filters

* Open Visual Studio.
* From the top menu, select **File->New->Project**.
* In **Create A New Project** screen, select **ASP.NET Core Web Application** from the list of available project types and click on **Next.**
* Enter **Project Name** as **Phase3Section2.12** and click on **Create.**
* From the list of project sub-types, choose **Web Application (Model-View-Controller)** and uncheck **Configure for HTTPS.** Click on **Create.**
* This will create the files for an ASP.NET MVC project.

**Step 2.6.2:** Creating a Login view to show an account login form

* In **Solution Explorer**, right click **Views** and choose **Add->New Folder.**
* Enter folder name as **Account**.
* Right click **Views->Account** and choose **Add->View.**
* Enter **View Name** as Login and click **Add.**
* Enter the following script:

@{

ViewData["Title"] = "Login";

}

<**h2**>Login</**h2**>

<**div** class="container">

<**form** method="post" action="/Home/HomeLoginAction">

<**div** class="row">

<**div** class="col-sm-5">

<**div** class="row">

<**div** class="col-sm-4">User Id</**div**>

<**div** class="col-sm-8"><**input** class="form-control" name="userid" /></**div**>

</**div**>

<**div** class="row">

<**div** class="col-sm-4">Password</**div**>

<**div** class="col-sm-8"><**input** class="form-control" name="pwd" type="password" /></**div**>

</**div**>

<**div** class="row">

<**div** class="col-sm-12 text-right">

<**button** class="btn btn-default">Login</**button**>

</**div**>

</**div**>

</**div**>

</**div**>

</**form**>

</**div**>

**Step 2.6.3:** Creating StudentView.cshtml to display student data without authorization

* In **Solution Explorer**,expand **Views->Home.** Right click **Home** and choose **Add->View.**
* Enter **View Name** as StudentView and click **Add.**
* Enter the following script:

@{

ViewData["Title"] = "StudentView";

}

<**h2**>StudentView</**h2**>

<**div** class="container">

<**div** class="row">

<**div** class="col-sm-4">Name</**div**>

<**div** class="col-sm-8">Walter Isaacs</**div**>

</**div**>

<**div** class="row">

<**div** class="col-sm-4">Class</**div**>

<**div** class="col-sm-8">7B</**div**>

</**div**>

<**div** class="row">

<**div** class="col-sm-4">Address</**div**>

<**div** class="col-sm-8">8-9, 5th Cross, 36th Street, Ash Town</**div**>

</**div**>

</**div**>

**Step 2.6.4:** Creating StudentViewSecure.cshtml to display student data with authorization

* In **Solution Explorer**,expand **Views->Home.** Right click **Home** and choose **Add->View**.
* Enter **View Name** as StudentViewSecure and click **Add.**
* Enter the following script:

@{

ViewData["Title"] = "StudentView";

}

<**h2**>StudentView - Authorized</**h2**>

<**div** class="container">

<**div** class="row">

<**div** class="col-sm-4">Name</**div**>

<**div** class="col-sm-8">Walter Isaacs</**div**>

</**div**>

<**div** class="row">

<**div** class="col-sm-4">Class</**div**>

<**div** class="col-sm-8">7B</**div**>

</**div**>

<**div** class="row">

<**div** class="col-sm-4">Address</**div**>

<**div** class="col-sm-8">8-9, 5th Cross, 36th Street, Ash Town</**div**>

</**div**>

</**div**>

**Step 2.6.5:** Creating AccountController to handle the login page

* In **Solution Explorer**,expand **Controllers**.Right click **Controllers** and choose **Add->Controller**.
* From the list of types, choose **MVC5 Controller Empty** and click **Add.**
* Enter the following code:

**using** System;

**using** System.Collections.Generic;

**using** System.Diagnostics;

**using** System.Linq;

**using** System.Threading.Tasks;

**using** Microsoft.AspNetCore.Authorization;

**using** Microsoft.AspNetCore.Mvc;

**using** Microsoft.AspNetCore.Authentication;

**using** Microsoft.AspNetCore.Authentication.Cookies;

**using** Phase3Section2.\_12.Models;

**using** System.Security.Claims;

**namespace** Phase3Section2.\_12.Controllers

{

**public** **class** AccountController : Controller

{

**public** IActionResult Login()

{

**return** View();

}

}

}

**Step 2.6.6:** Creating ActionLoggingFilter to log all filter operations

* In **Solution Explorer**, expand **Controllers**.Right click **Controllers** and choose **Add->Controller**.
* From the list of types, choose **MVC5 Controller Empty** and click **Add.**
* Enter the following code:

**using** Microsoft.AspNetCore.Mvc.Filters;

**using** Microsoft.AspNetCore.Routing;

**using** System;

**using** System.Collections.Generic;

**using** System.Diagnostics;

**using** System.Linq;

**using** System.Threading.Tasks;

**namespace** Phase3Section2.\_12.Controllers

{

**public** **class** ActionLoggingFilter : ActionFilterAttribute

{

**public** **override** **void** OnActionExecuted(ActionExecutedContext filterContext)

{

Log("OnActionExecuted", filterContext.RouteData);

}

**public** **override** **void** OnActionExecuting(ActionExecutingContext filterContext)

{

Log("OnActionExecuting", filterContext.RouteData);

}

**public** **override** **void** OnResultExecuted(ResultExecutedContext filterContext)

{

Log("OnResultExecuted", filterContext.RouteData);

}

**public** **override** **void** OnResultExecuting(ResultExecutingContext filterContext)

{

Log("OnResultExecuting ", filterContext.RouteData);

}

**private** **void** Log(**string** methodName, RouteData routeData)

{

**var** controllerName = routeData.Values["controller"];

**var** actionName = routeData.Values["action"];

**var** message = String.Format("{0}- controller:{1} action:{2}", methodName,

controllerName,

actionName);

Debug.WriteLine(message);

}

}

}

**Step 2.6.7:** Making changes to Index.cshtml to show custom filter options

* In **Solution Explorer**,expand **Views->Home** and double click Index.cshtml.
* Enter the following script:

@{

ViewData["Title"] = "Home Page";

}

<**div** class="container">

<**br** />

<**div** class="row">

<**div** class="col-sm-8">

<**a** href="Home/StudentView">Student View without Authorization Filter</**a**><**br** />

<**a** href="Home/StudentViewSecure">Student View requiring Authorization Filter</**a**>

</**div**>

</**div**>

</**div**>

**Step 2.6.8:** Changing HomeController to use the views for custom filters

* In **Solution Explorer**,expand **Controllers** and double click HomeController.
* Enter the following code:

**using** System;

**using** System.Collections.Generic;

**using** System.Diagnostics;

**using** System.Linq;

**using** System.Threading.Tasks;

**using** Microsoft.AspNetCore.Authorization;

**using** Microsoft.AspNetCore.Mvc;

**using** Microsoft.AspNetCore.Authentication;

**using** Microsoft.AspNetCore.Authentication.Cookies;

**using** Phase3Section2.\_12.Models;

**using** System.Security.Claims;

**namespace** Phase3Section2.\_12.Controllers

{

[ActionLoggingFilter]

**public** **class** HomeController : Controller

{

**public** IActionResult Index()

{

**return** View();

}

**public** IActionResult About()

{

ViewData["Message"] = "Your application description page.";

**return** View();

}

**public** IActionResult Contact()

{

ViewData["Message"] = "Your contact page.";

**return** View();

}

**public** IActionResult Privacy()

{

**return** View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = **true**)]

**public** IActionResult Error()

{

**return** View(**new** ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

[AllowAnonymous]

**public** ViewResult StudentView()

{

**return** View();

}

**public** ViewResult Login()

{

**return** View();

}

[Authorize]

**public** ViewResult StudentViewSecure()

{

**return** View();

}

**public** IActionResult HomeLoginAction()

{

**string** userName = "admin";

**var** identity = **new** ClaimsIdentity(**new**[] {

**new** Claim(ClaimTypes.Name, userName)

}, CookieAuthenticationDefaults.AuthenticationScheme);

**var** principal = **new** ClaimsPrincipal(identity);

**var** login = HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, principal);

**return** RedirectToAction("StudentViewSecure", "Home");

}

}

}

**Step 2.6.9:** Building the project

* From the top menu, choose **Build->Build Solution**.
* If any compile errors are shown, fix them as required.

**Step 2.6.10:** Publishing and running the project

* From the top menu, select **Debug->Start Without Debugging.**
* This will execute the program in the default browser.
* To see the student pages, go to the url : http://localhost:xxxx/students.

**Step 2.6.11:** Pushing the code to your GitHub repositories

Open your command prompt and navigate to the folder where you have created your files.

cd <folder path>

Initialize your repository using the following command:

git init

Add all the files to your git repository using the following command:

git add .

Commit the changes using the following command:

git commit -m “Changes have been committed.”

Push the files to the folder you created initially using the following command:

git push -u origin master