Authentication filter

**CustomAuthFilter.cs class in new folder**

1. **using** System;
2. **using** System.Web.Mvc;
3. **using** System.Web.Mvc.Filters;
4. **using** System.Web.Routing;
6. **namespace** CustomAuthenticationFilter.Infrastructure
7. {
8. **public** **class** CustomAuthFilter : ActionFilterAttribute, IAuthenticationFilter
9. {
10. **public** **void** OnAuthentication(AuthenticationContext filterContext)
11. {
12. **if** (**string**.IsNullOrEmpty(Convert.ToString(filterContext.HttpContext.Session["UserName"])))
13. {
14. filterContext.Result = **new** HttpUnauthorizedResult();
15. }
16. }
17. **public** **void** OnAuthenticationChallenge(AuthenticationChallengeContext filterContext)
18. {
19. **if** (filterContext.Result == **null** || filterContext.Result **is** HttpUnauthorizedResult)
20. {
21. //Redirecting the user to the Login View of Account Controller
22. filterContext.Result = **new** RedirectToRouteResult(
23. **new** RouteValueDictionary
24. {
25. { "controller", "Account" },
26. { "action", "Login" }
27. });
28. }
29. }
30. }
31. }

**Authorization filter**

web.config file and write the following code under the system.web section.

1. <authentication mode="Forms">
2. <forms loginUrl="/Home/Login"></forms>
3. </authentication>

**In Controller**

1. [HttpGet]
2. **public** ActionResult Login()
3. {
4. **return** View();
5. }
7. [HttpPost]
8. **public** ActionResult Login(User model)
9. {
10. **if** (ModelState.IsValid)
11. {
12. **using** (var context = **new** SqlDbContext())
13. {
14. User user = context.Users
15. .Where(u => u.UserId == model.UserId && u.Password == model.Password)
16. .FirstOrDefault();
18. **if** (user != **null**)
19. {
20. Session["UserName"] = user.UserName;
21. Session["UserId"] = user.UserId;
22. **return** RedirectToAction("Index", "Home");
23. }
24. **else**
25. {
26. ModelState.AddModelError("", "Invalid User Name or Password");
27. **return** View(model);
28. }
29. }
30. }
31. **else**
32. {
33. **return** View(model);
34. }
35. }
37. [HttpPost]
38. [ValidateAntiForgeryToken]
39. **public** ActionResult LogOff()
40. {
41. Session["UserName"] = **string**.Empty;
42. Session["UserId"] = **string**.Empty;
43. **return** RedirectToAction("Index", "Home");
44. }
45. }
46. }

**Authentication filter**

1. **public** **class** CustomAuthenticationFilter: ActionFilterAttribute, IAuthenticationFilter
2. {
3. **public** **void** OnAuthentication(AuthenticationContext filterContext)
4. {
5. **if** (**string**.IsNullOrEmpty(Convert.ToString(filterContext.HttpContext.Session["UserName"])))
6. {
7. filterContext.Result = **new** HttpUnauthorizedResult();
8. }
9. }
10. **public** **void** OnAuthenticationChallenge(AuthenticationChallengeContext filterContext)
11. {
12. **if** (filterContext.Result == **null** || filterContext.Result **is** HttpUnauthorizedResult)
13. {
14. //Redirecting the user to the Login View of Account Controller
15. filterContext.Result = **new** RedirectToRouteResult(
16. **new** RouteValueDictionary
17. {
18. { "controller", "Account" },
19. { "action", "Login" }
20. });
21. }
22. }
23. }

**CustomAuthorizeAttribute.cs**

1. **public** **class** CustomAuthorizeAttribute : AuthorizeAttribute
2. {
3. **private** **readonly** **string**[] allowedroles;
4. **public** CustomAuthorizeAttribute(**params** **string**[] roles)
5. {
6. **this**.allowedroles = roles;
7. }
8. **protected** **override** **bool** AuthorizeCore(HttpContextBase httpContext)
9. {
10. **bool** authorize = **false**;
11. var userId = Convert.ToString(httpContext.Session["UserId"]);
12. **if** (!**string**.IsNullOrEmpty(userId))
13. **using** (var context = **new** SqlDbContext())
14. {
15. var userRole = (from u **in** context.Users
16. join r **in** context.Roles on u.RoleId equals r.Id
17. where u.UserId == userId
18. select **new**
19. {
20. r.Name
21. }).FirstOrDefault();
22. **foreach** (var role **in** allowedroles)
23. {
24. **if** (role == userRole.Name) **return** **true**;
25. }
26. }

29. **return** authorize;
30. }
32. **protected** **override** **void** HandleUnauthorizedRequest(AuthorizationContext filterContext)
33. {
34. filterContext.Result = **new** RedirectToRouteResult(
35. **new** RouteValueDictionary
36. {
37. { "controller", "Home" },
38. { "action", "UnAuthorized" }
39. });
40. }
41. }

In controller.cs file

1. [CustomAuthenticationFilter]
2. **public** **class** HomeController : Controller
3. {
4. [CustomAuthorize("Normal", "SuperAdmin")]
5. **public** ActionResult Index()
6. {
7. **return** View();
8. }
10. [CustomAuthorize("Admin", "SuperAdmin")]
11. **public** ActionResult About()
12. {
13. ViewBag.Message = "Your application description page.";
15. **return** View();
16. }

CRUD OPERATIONS USING AJAX

Create db class and generate the db

Create another class in model folder

1. **public** **class** EmployeeDB
2. {
3. //declare connection string
4. **string** cs = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
6. //Return list of all Employees
7. **public** List<Employee> ListAll()
8. {
9. List<Employee> lst = **new** List<Employee>();
10. **using**(SqlConnection con=**new** SqlConnection(cs))
11. {
12. con.Open();
13. SqlCommand com = **new** SqlCommand("SelectEmployee",con);
14. com.CommandType = CommandType.StoredProcedure;
15. SqlDataReader rdr = com.ExecuteReader();
16. **while**(rdr.Read())
17. {
18. lst.Add(**new** Employee {
19. EmployeeID=Convert.ToInt32(rdr["EmployeeId"]),
20. Name=rdr["Name"].ToString(),
21. Age = Convert.ToInt32(rdr["Age"]),
22. State = rdr["State"].ToString(),
23. Country = rdr["Country"].ToString(),
24. });
25. }
26. **return** lst;
27. }
28. }
30. //Method for Adding an Employee
31. **public** **int** Add(Employee emp)
32. {
33. **int** i;
34. **using**(SqlConnection con=**new** SqlConnection(cs))
35. {
36. con.Open();
37. SqlCommand com = **new** SqlCommand("InsertUpdateEmployee", con);
38. com.CommandType = CommandType.StoredProcedure;
39. com.Parameters.AddWithValue("@Id",emp.EmployeeID);
40. com.Parameters.AddWithValue("@Name", emp.Name);
41. com.Parameters.AddWithValue("@Age", emp.Age);
42. com.Parameters.AddWithValue("@State", emp.State);
43. com.Parameters.AddWithValue("@Country", emp.Country);
44. com.Parameters.AddWithValue("@Action", "Insert");
45. i = com.ExecuteNonQuery();
46. }
47. **return** i;
48. }
50. //Method for Updating Employee record
51. **public** **int** Update(Employee emp)
52. {
53. **int** i;
54. **using** (SqlConnection con = **new** SqlConnection(cs))
55. {
56. con.Open();
57. SqlCommand com = **new** SqlCommand("InsertUpdateEmployee", con);
58. com.CommandType = CommandType.StoredProcedure;
59. com.Parameters.AddWithValue("@Id", emp.EmployeeID);
60. com.Parameters.AddWithValue("@Name", emp.Name);
61. com.Parameters.AddWithValue("@Age", emp.Age);
62. com.Parameters.AddWithValue("@State", emp.State);
63. com.Parameters.AddWithValue("@Country", emp.Country);
64. com.Parameters.AddWithValue("@Action", "Update");
65. i = com.ExecuteNonQuery();
66. }
67. **return** i;
68. }
70. //Method for Deleting an Employee
71. **public** **int** Delete(**int** ID)
72. {
73. **int** i;
74. **using** (SqlConnection con = **new** SqlConnection(cs))
75. {
76. con.Open();
77. SqlCommand com = **new** SqlCommand("DeleteEmployee", con);
78. com.CommandType = CommandType.StoredProcedure;
79. com.Parameters.AddWithValue("@Id", ID);
80. i = com.ExecuteNonQuery();
81. }
82. **return** i;
83. }
84. }

In Controller.cs file

1. EmployeeDB empDB = **new** EmployeeDB();
2. // GET: Home
3. **public** ActionResult Index()
4. {
5. **return** View();
6. }
7. **public** JsonResult List()
8. {
9. **return** Json(empDB.ListAll(),JsonRequestBehavior.AllowGet);
10. }
11. **public** JsonResult Add(Employee emp)
12. {
13. **return** Json(empDB.Add(emp), JsonRequestBehavior.AllowGet);
14. }
15. **public** JsonResult GetbyID(**int** ID)
16. {
17. var Employee = empDB.ListAll().Find(x => x.EmployeeID.Equals(ID));
18. **return** Json(Employee, JsonRequestBehavior.AllowGet);
19. }
20. **public** JsonResult Update(Employee emp)
21. {
22. **return** Json(empDB.Update(emp), JsonRequestBehavior.AllowGet);
23. }
24. **public** JsonResult Delete(**int** ID)
25. {
26. **return** Json(empDB.Delete(ID), JsonRequestBehavior.AllowGet);
27. }

In Index view

1. <script src="~/Scripts/jquery-1.9.1.js"></script>
2. <script src="~/Scripts/bootstrap.js"></script>
3. <link href="~/Content/bootstrap.css" rel="stylesheet" />
4. <script src="~/Scripts/employee.js"></script>
5. <button type="button" class="btn btn-primary" data-toggle="modal" data-target="#myModal" onclick="clearTextBox();">Add New Employee</button><br /><br />
6. <table class="table table-bordered table-hover">
7. <button type="button" class="btn btn-primary" id="btnAdd" onclick="return Add();">Add</button>
8. <button type="button" class="btn btn-primary" id="btnUpdate" style="display:none;" onclick="Update();">Update</button>
9. <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>

Employee.js file

1. //Load Data in Table when documents is ready
2. $(document).ready(**function** () {
3. loadData();
4. });
6. //Load Data function
7. **function** loadData() {
8. $.ajax({
9. url: "/Home/List",
10. type: "GET",
11. contentType: "application/json;charset=utf-8",
12. dataType: "json",
13. success: **function** (result) {
14. **var** html = '';
15. $.each(result, **function** (key, item) {
16. html += '<tr>';
17. html += '<td>' + item.EmployeeID + '</td>';
18. html += '<td>' + item.Name + '</td>';
19. html += '<td>' + item.Age + '</td>';
20. html += '<td>' + item.State + '</td>';
21. html += '<td>' + item.Country + '</td>';
22. html += '<td><a href="#" onclick="return getbyID(' + item.EmployeeID + ')">Edit</a> | <a href="#" onclick="Delele(' + item.EmployeeID + ')">Delete</a></td>';
23. html += '</tr>';
24. });
25. $('.tbody').html(html);
26. },
27. error: **function** (errormessage) {
28. alert(errormessage.responseText);
29. }
30. });
31. }
33. //Add Data Function
34. **function** Add() {
35. **var** res = validate();
36. **if** (res == **false**) {
37. **return** **false**;
38. }
39. **var** empObj = {
40. EmployeeID: $('#EmployeeID').val(),
41. Name: $('#Name').val(),
42. Age: $('#Age').val(),
43. State: $('#State').val(),
44. Country: $('#Country').val()
45. };
46. $.ajax({
47. url: "/Home/Add",
48. data: JSON.stringify(empObj),
49. type: "POST",
50. contentType: "application/json;charset=utf-8",
51. dataType: "json",
52. success: **function** (result) {
53. loadData();
54. $('#myModal').modal('hide');
55. },
56. error: **function** (errormessage) {
57. alert(errormessage.responseText);
58. }
59. });
60. }
62. //Function for getting the Data Based upon Employee ID
63. **function** getbyID(EmpID) {
64. $('#Name').css('border-color', 'lightgrey');
65. $('#Age').css('border-color', 'lightgrey');
66. $('#State').css('border-color', 'lightgrey');
67. $('#Country').css('border-color', 'lightgrey');
68. $.ajax({
69. url: "/Home/getbyID/" + EmpID,
70. typr: "GET",
71. contentType: "application/json;charset=UTF-8",
72. dataType: "json",
73. success: **function** (result) {
74. $('#EmployeeID').val(result.EmployeeID);
75. $('#Name').val(result.Name);
76. $('#Age').val(result.Age);
77. $('#State').val(result.State);
78. $('#Country').val(result.Country);
80. $('#myModal').modal('show');
81. $('#btnUpdate').show();
82. $('#btnAdd').hide();
83. },
84. error: **function** (errormessage) {
85. alert(errormessage.responseText);
86. }
87. });
88. **return** **false**;
89. }
91. //function for updating employee's record
92. **function** Update() {
93. **var** res = validate();
94. **if** (res == **false**) {
95. **return** **false**;
96. }
97. **var** empObj = {
98. EmployeeID: $('#EmployeeID').val(),
99. Name: $('#Name').val(),
100. Age: $('#Age').val(),
101. State: $('#State').val(),
102. Country: $('#Country').val(),
103. };
104. $.ajax({
105. url: "/Home/Update",
106. data: JSON.stringify(empObj),
107. type: "POST",
108. contentType: "application/json;charset=utf-8",
109. dataType: "json",
110. success: **function** (result) {
111. loadData();
112. $('#myModal').modal('hide');
113. $('#EmployeeID').val("");
114. $('#Name').val("");
115. $('#Age').val("");
116. $('#State').val("");
117. $('#Country').val("");
118. },
119. error: **function** (errormessage) {
120. alert(errormessage.responseText);
121. }
122. });
123. }
125. //function for deleting employee's record
126. **function** Delele(ID) {
127. **var** ans = confirm("Are you sure you want to delete this Record?");
128. **if** (ans) {
129. $.ajax({
130. url: "/Home/Delete/" + ID,
131. type: "POST",
132. contentType: "application/json;charset=UTF-8",
133. dataType: "json",
134. success: **function** (result) {
135. loadData();
136. },
137. error: **function** (errormessage) {
138. alert(errormessage.responseText);
139. }
140. });
141. }
142. }
144. //Function for clearing the textboxes
145. **function** clearTextBox() {
146. $('#EmployeeID').val("");
147. $('#Name').val("");
148. $('#Age').val("");
149. $('#State').val("");
150. $('#Country').val("");
151. $('#btnUpdate').hide();
152. $('#btnAdd').show();
153. $('#Name').css('border-color', 'lightgrey');
154. $('#Age').css('border-color', 'lightgrey');
155. $('#State').css('border-color', 'lightgrey');
156. $('#Country').css('border-color', 'lightgrey');
157. }
158. //Valdidation using jquery
159. **function** validate() {
160. **var** isValid = **true**;
161. **if** ($('#Name').val().trim() == "") {
162. $('#Name').css('border-color', 'Red');
163. isValid = **false**;
164. }
165. **else** {
166. $('#Name').css('border-color', 'lightgrey');
167. }
168. **if** ($('#Age').val().trim() == "") {
169. $('#Age').css('border-color', 'Red');
170. isValid = **false**;
171. }
172. **else** {
173. $('#Age').css('border-color', 'lightgrey');
174. }
175. **if** ($('#State').val().trim() == "") {
176. $('#State').css('border-color', 'Red');
177. isValid = **false**;
178. }
179. **else** {
180. $('#State').css('border-color', 'lightgrey');
181. }
182. **if** ($('#Country').val().trim() == "") {
183. $('#Country').css('border-color', 'Red');
184. isValid = **false**;
185. }
186. **else** {
187. $('#Country').css('border-color', 'lightgrey');
188. }
189. **return** isValid;
190. }

Multilanguage

Create resource folder and add new item as resourse in it

Name like language.es.resx, language.fr.resx, language.resx

Add class in model folder

public class Employee

{

[Display(Name = "FirstName", ResourceType =typeof(Language))]

[Required(ErrorMessageResourceType=typeof(Language),ErrorMessageResourceName ="FirstNameRequired")]

public string FirstName { get; set; }

language controller.cs

public class LanguageController : Controller

{

// GET: Language

public ActionResult Index(string language)

{

if(!String.IsNullOrEmpty(language))

{

Thread.CurrentThread.CurrentCulture = CultureInfo.CreateSpecificCulture(language);

Thread.CurrentThread.CurrentUICulture = new CultureInfo(language);

}

HttpCookie cookie = new HttpCookie("Languages");

cookie.Value = language;

Response.Cookies.Add(cookie);

return View();

}

}

In homecontroller.cs

[HttpPost]

public ActionResult Index(string FirstName)

{

return View();

}