Web API uses authorization filters to implement authorization.  The Authorization filters run before the controller action. If the request is not authorized, the filter returns an error response, and the action is not invoked.

Web API provides a built-in authorization filter, Authorize Attribute. This filter checks whether the user is authenticated. If not then it returns the HTTP status code 401 (Unauthorized), without invoking the action.  
  
**Getting Started**

* Create a new Project. Open Visual Studio 2019
* Go to "File" -> "New" -> "Project...".
* Select "Web" in the installed templates.
* Select "ASP.NET MVC 5 Web Application".
* Select Web API, View engine should remain Razor.
* Enter the Name and choose the location.
* Click"OK".

In this sample, I will use Knockout to display data on the client-side.

First, add a model class as in the following:

1. **public** **class** Employee
2. {
3. **public** **int** EmployeeID { **get**; **set**; }
4. **public** **string** LastName { **get**; **set**; }
5. **public** **string** FirstName { **get**; **set**; }
6. **public** **string** City { **get**; **set**; }
7. **public** **string** Region { **get**; **set**; }
8. **public** **string** PostalCode { **get**; **set**; }
9. **public** **string** Country { **get**; **set**; }
10. }

Now add a class as in the following:

1. **public** **class** ApplicationAuthenticationHandler : DelegatingHandler
2. {
3. // Http Response Messages
4. **private** **const** **string** InvalidToken = "Invalid Authorization-Token";
5. **private** **const** **string** MissingToken = "Missing Authorization-Token";
6. **protected** **override** System.Threading.Tasks.Task<HttpResponseMessage> SendAsync(HttpRequestMessage
7. request, System.Threading.CancellationToken cancellationToken)
8. {
9. IEnumerable<**string**> sampleApiKeyHeaderValues = **null**;
10. // Checking the Header values
11. **if** (request.Headers.TryGetValues("X-SampleAppApiKey", **out** sampleApiKeyHeaderValues))
12. {
13. **string**[] apiKeyHeaderValue = sampleApiKeyHeaderValues.First().Split(':');
14. // Validating header value must have both APP ID & APP key
15. **if** (apiKeyHeaderValue.Length == 2)
16. {
17. // Code logic after authenciate the application.
18. var appID = apiKeyHeaderValue[0];
19. var AppKey = apiKeyHeaderValue[1];
20. **if** (appID.Equals("SampleAppX123") && AppKey.Equals("YesAppKeyIsPersist"))
21. {
22. var userNameClaim = **new** Claim(ClaimTypes.Name, appID);
23. var identity = **new** ClaimsIdentity(**new**[] { userNameClaim }, "SampleAppApiKey");
24. var principal = **new** ClaimsPrincipal(identity);
25. Thread.CurrentPrincipal = principal;
26. **if** (System.Web.HttpContext.Current != **null**)
27. {
28. System.Web.HttpContext.Current.User = principal;
29. }
30. }
31. **else**
32. {
33. // Web request cancel reason APP key is NULL
34. **return** requestCancel(request, cancellationToken, InvalidToken);
35. }
36. }
37. **else**
38. {
39. // Web request cancel reason missing APP key or APP ID
40. **return** requestCancel(request, cancellationToken, MissingToken);
41. }
42. }
43. **else**
44. {
45. // Web request cancel reason APP key missing all parameters
46. **return** requestCancel(request, cancellationToken, MissingToken);
47. }
48. **return** **base**.SendAsync(request, cancellationToken);
49. }
50. **private** System.Threading.Tasks.Task<HttpResponseMessage> requestCancel(HttpRequestMessage
51. request, System.Threading.CancellationToken cancellationToken, **string** message)
52. {
53. CancellationTokenSource \_tokenSource = **new** CancellationTokenSource();
54. cancellationToken = \_tokenSource.Token;
55. \_tokenSource.Cancel();
56. HttpResponseMessage response = **new** HttpResponseMessage();
57. response = request.CreateResponse(HttpStatusCode.BadRequest);
58. response.Content = **new** StringContent(message);
59. **return** **base**.SendAsync(request, cancellationToken).ContinueWith(task =>
60. {
61. **return** response;
62. });
63. }
64. }

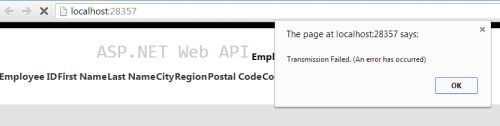
Now add the following controller class:

1. **public** **class** ValuesController : ApiController
2. {
3. **private** List<Employee> EmpList = **new** List<Employee>();
4. // GET api/values
5. **public** IEnumerable<Employee> Get()
6. {
7. EmpList.Add(**new** Employee { EmployeeID = 1, FirstName = "Nancy", LastName = "Davolio",
8. City = "Seattle", Region = "WA", PostalCode = "98122", Country = "USA" });
9. EmpList.Add(**new** Employee { EmployeeID = 2, FirstName = "Andrew", LastName = "Fuller",
10. City = "Tacoma", Region = "WA", PostalCode = "98401", Country = "USA" });
11. EmpList.Add(**new** Employee { EmployeeID = 3, FirstName = "Janet", LastName = "Leverling",
12. City = "Kirkland", Region = "WA", PostalCode = "98033", Country = "USA" });
13. EmpList.Add(**new** Employee { EmployeeID = 4, FirstName = "Margaret", LastName = "Peacock",
14. City = "Redmond", Region = "WA", PostalCode = "98052", Country = "USA" });
15. EmpList.Add(**new** Employee { EmployeeID = 5, FirstName = "Steven", LastName = "Buchanan",
16. City = "London", Region = "WA", PostalCode = "SW1 8JR", Country = "UK" });
17. EmpList.Add(**new** Employee { EmployeeID = 6, FirstName = "Michael", LastName = "Suyama",
18. City = "London", Region = "WA", PostalCode = "EC2 7JR", Country = "UK" });
19. EmpList.Add(**new** Employee { EmployeeID = 7, FirstName = "Robert", LastName = "King",
20. City = "London", Region = "WA", PostalCode = "RG1 9SP", Country = "UK" });
21. EmpList.Add(**new** Employee { EmployeeID = 8, FirstName = "Laura", LastName = "Callahan",
22. City = "Seattle", Region = "WA", PostalCode = "98105", Country = "USA" });
23. EmpList.Add(**new** Employee { EmployeeID = 9, FirstName = "Anne", LastName = "Dodsworth",
24. City = "London", Region = "WA", PostalCode = "WG2 7LT", Country = "UK" });
25. **return** EmpList;
26. }

Add the following view to display data:

1. <script src="~/Scripts/jquery-1.8.2.min.js"></script>
2. <script src="~/Scripts/knockout-2.2.0.js"></script>
3. <script type="text/javascript">
4. $(document).ready(**function** () {
5. FetchEmployees();
6. });
7. **function** FetchEmployees() {
8. viewModel = {
9. employeeCollection: ko.observableArray()
10. };
11. $.ajax({
12. type: "GET",
13. url: "http://localhost:28357/api/values",
14. contentType: "application/json; charset=utf-8",
15. headers: { 'X-SampleAppApiKey': 'SampleAppX123:YesAppKeyIsPersist' },
16. dataType: "json",
17. success: **function** (response) {
18. **if** (response != "") {
19. $(response).each(**function** (index, element) {
20. viewModel.employeeCollection.push(element);
21. });
22. ko.applyBindings(viewModel);
23. }
24. },
25. error: **function** (event) {
26. //If any errors occurred - detail them here
27. alert("Transmission Failed. (An error has occurred)");
28. }
29. });
30. }
31. </script>
32. <h3>Employees List</h3>
33. <table id="empl" data-bind="visible: employeeCollection().length > 0">
34. <thead>
35. <tr>
36. <th>Employee ID
37. </th>
38. <th>First Name
39. </th>
40. <th>Last Name
41. </th>
42. <th>City
43. </th>
44. <th>Region
45. </th>
46. <th>Postal Code
47. </th>
48. <th>Country
49. </th>
50. </tr>
51. </thead>
52. <tbody data-bind="foreach: employeeCollection">
53. <tr>
54. <td data-bind="text: EmployeeID"></td>
55. <td data-bind="text: FirstName"></td>
56. <td data-bind="text: LastName"></td>
57. <td data-bind="text: City"></td>
58. <td data-bind="text: Region"></td>
59. <td data-bind="text: PostalCode"></td>
60. <td data-bind="text: Country"></td>
61. <td>
62. <button data-bind="click: $root.edit">
63. Edit</button>
64. <button data-bind="click: $root.delete">
65. Delete</button>
66. </td>
67. </tr>
68. </tbody>
69. </table>

Now run without authorization. Let's see what we get.



Now add the Authorize attribute to the Get method.

1. [Authorize]
2. **public** IEnumerable<Employee> Get()
3. {
4. }

Now run again.  
  
  
If you want authorization on all the actions of a controller then put Authorize above the controller class as in the following:

1. [Authorize]
2. **public** **class** ValuesController : ApiController
3. {
4. **private** List<Employee> EmpList = **new** List<Employee>();
5. // GET api/values
6. [HttpGet]
7. [Authorize]
8. **public** IEnumerable<Employee> Get()
9. {
10. }
11. // GET api/values/5
12. [AllowAnonymous]
13. **public** Employee Get(**int** id)
14. {
15. **return** EmpList.Find(e => e.EmployeeID == id);
16. }
17. }

You can set permission for a specific user like this.

1. [Authorize(Users = "Raj,Sam")]
2. **public** **class** ValuesController : ApiController
3. {
4. }

You can provide authorization for a specific user role also.

1. [Authorize(Roles = "Administrators")]
2. **public** **class** ValuesController : ApiController
3. {
4. }