486 – Preventing cross-site calls to your API.

# Method #1

This will affect all API calls into your application. It is a global switch.

## Server Side

1. Create a Handlers Folder in your application
2. Inside this newly created folder, add a new C# class called **AntiForgeryHandler** and make this class inherit from **DelegatingHandler**
3. Create an override for the method **Task<HttpResponseMessage> SendAsync(HttpRequestMessage request, CancellationToken cancellationToken)** (*using System.Net.Http*)
4. Set the contents of this method to the following:

string form = string.Empty;

string cookie = string.Empty;

IEnumerable<string> headers;

if (request.Headers.TryGetValues("AntiForgeryToken", out headers))

{

var tokens = headers.First().Split(':');

if (tokens.Length == 2)

{

cookie = tokens[0];

form = tokens[1];

}

}

try

{

AntiForgery.Validate(cookie, form);

}

catch (Exception)

{

var response = new HttpResponseMessage(System.Net.HttpStatusCode.Forbidden)

{

Content = new StringContent("Cross-site request denied")

};

return Task.FromResult(response);

}

return base.SendAsync(request, cancellationToken);

1. Under the App\_Start folder, find and open WebApiConfig.cs. Add the following line of code to the **Register** method immediately following the comment about bearer token authentication and the two lines of code that follow it.

config.MessageHandlers.Add(new AntiForgeryHandler());

1. Complete! All API calls MUST come in with a valid AntiForgerytoken in the Header of the HTTP request.

## Client Side

1. In the razer code section at the beginning of each view which will need to communicate with your API, place the following code to bring the token into the page:

@{

string cookie;

string form;

AntiForgery.GetTokens("", out cookie, out form);

}

1. In the javascript section just before you make your Ajax call (if using JQuery), place the following code:

var AntiForgeryToken = "@cookie" + ":" + "@form";

1. In the JavaScript section where you make your $.ajax() call, place the beforeSend handler as shown:

$.ajax({

…

beforeSend: function (xhr) {

xhr.setRequestHeader("AntiForgeryToken", AntiForgeryToken);

},

success: function (data) {

…

},

error: function (err) {

…

}

1. Complete! All API calls set up in this way will get past the global

# Method #2

This will affect only specific calls or specific controllers that you yourself set

## Server Side

1. Create a new folder in the project call Filters
2. Add a new class to this folder called PreventExternalCalls. Inherit from FilterAttribute and implement IAuthorizationFilter.
3. Implement the required method ExecuteAuthorizationFilterAsync. Set the contents of this method to:

string form = string.Empty;

string cookie = string.Empty;

IEnumerable<string> headers;

if (request.Headers.TryGetValues("AntiForgeryToken", out headers))

{

var tokens = headers.First().Split(':');

if (tokens.Length == 2)

{

cookie = tokens[0];

form = tokens[1];

}

}

try

{

AntiForgery.Validate(cookie, form);

}

catch (Exception)

{

var response = new HttpResponseMessage(System.Net.HttpStatusCode.Forbidden)

{

Content = new StringContent("Cross-site request denied")

};

return Task.FromResult(response);

}

return base.SendAsync(request, cancellationToken);

1. Add the [PreventExternalCalls] attribute on every method where you want the Anti-Forgery Validation Token to be used.

## Client Side

Same as the Global.