**Enable SSL in Visual Studio Development Server**

In this article, I am going to discuss **How to Enable SSL in Visual Studio Development Server** with an example. While developing any application if you want to test the service using https protocol then you need to enable SSL in visual studio.  Let us understand how to enable SSL in Visual Studio with an example.

**Creating an Empty Application:**

First, create an empty Web API application with the name WebAPIEnableHTTP. Once you create the project then add the following model class (Employee.cs) with the Models folder.

**<**span style="font-family: arial, helvetica, sans-serif;"**>namespace** WebAPIEnableHTTPS.Models

**{**

**public** **class** Employee

**{**

**public** **int** EmployeeID **{** **get**; **set**; **}**

**public** string EmployeeName **{** **get**; **set**; **}**

**}**

**}**

**<**/span**>**

Once you add the Model then you need to add a **Web API 2 Controller – Empty** within the Controllers folder and name it as **EmployeesController** and then copy and paste the following code in it.

**<**span style="font-family: arial, helvetica, sans-serif;"**>using** System.Collections.Generic;

**using** *System.Linq;*

**using** *System.Web.Http;*

**using** *WebAPIEnableHTTPS.Models;*

**namespace** *WebAPIEnableHTTPS.Controllers*

**{**

**public** **class** EmployeesController : ApiController

**{**

List**<**Employee**>** employees = new List**<**Employee**>()**

**{**

new Employee**()** **{** EmployeeID = 101, EmployeeName = "Anurag"**}**,

new Employee**()** **{** EmployeeID = 102, EmployeeName = "Priyanka"**}**,

new Employee**()** **{** EmployeeID = 103, EmployeeName = "Sambit"**}**,

new Employee**()** **{** EmployeeID = 104, EmployeeName = "Preety"**}**,

**}**;

**public** IEnumerable**<**Employee**>** Get**()**

**{**

**return** employees;

**}**

**public** Employee Get**(int** id**)**

**{**

**return** employees.FirstOrDefault**(**s =**>** s.EmployeeID == id**)**;

**}**

**}**

**}**

**<**/span**>**

At the moment when we navigate to the following URL, we get the output as expected.

**http://localhost:55486/api/employees** (please change the port number where your application is running),

Let’s change the protocol to https instead of HTTP and see what happened.

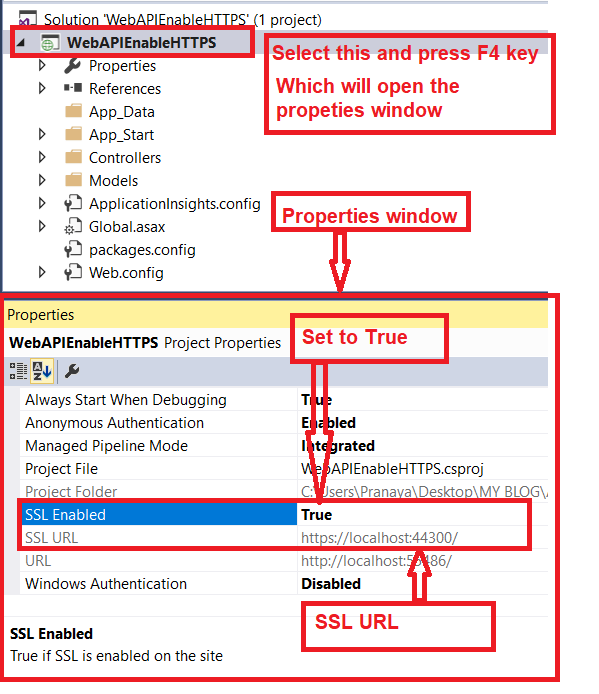
**https://localhost:55486/api/employees**

We get the error page as This site can’t provide a secure connection. This is because we have not enabled SSL for our Web API Service.

**How to Enable SSL in Visual Studio Development Server?**

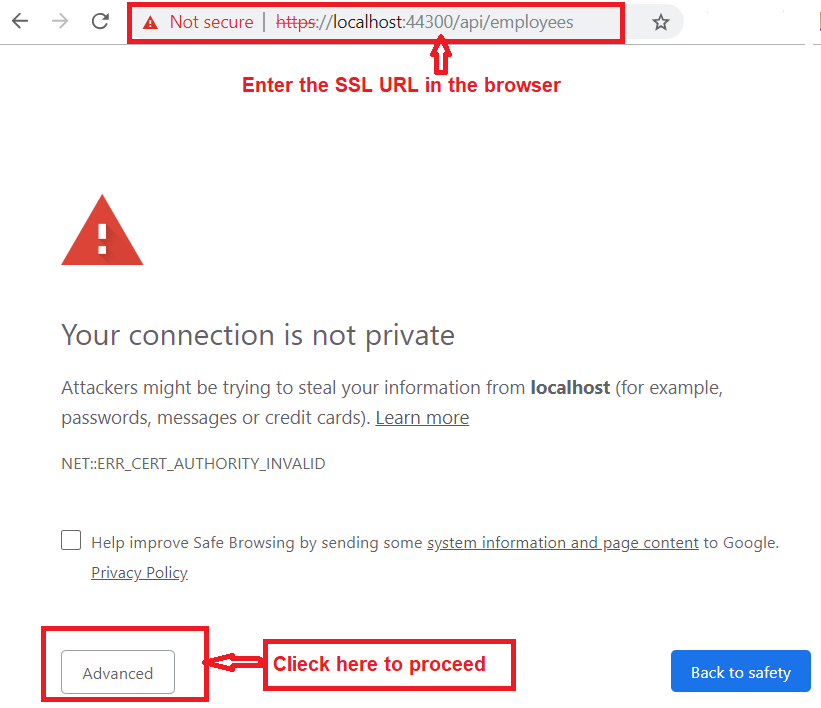
To enable SSL in the Visual Studio development server, you need to follow the below steps

In the Solution Explorer click on the **WebAPIEnableHTTP** Web API project and press **F4** key on the keyboard which will open the Project Properties window. From the Project Properties window, we need to set the **SSL Enabled property** to **true**. As soon as you do this Visual Studio sets the **SSL URL** as shown in the image below.

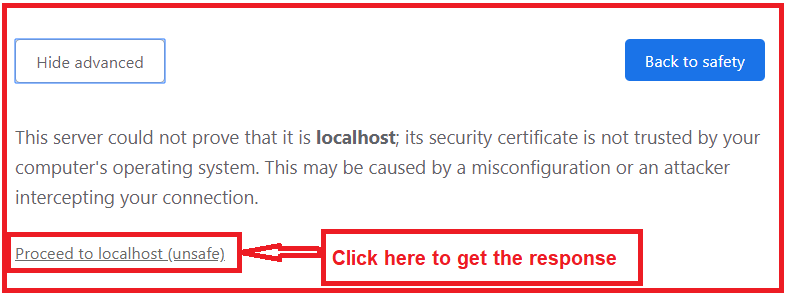


As shown in the above image, once you set the **SSL Enabled** property to **True**, now you have two URLs one is SSL URL and the other one is the only URL. The SSL URL is **https://localhost:44300**/ and the URL is **http://localhost:55486/**

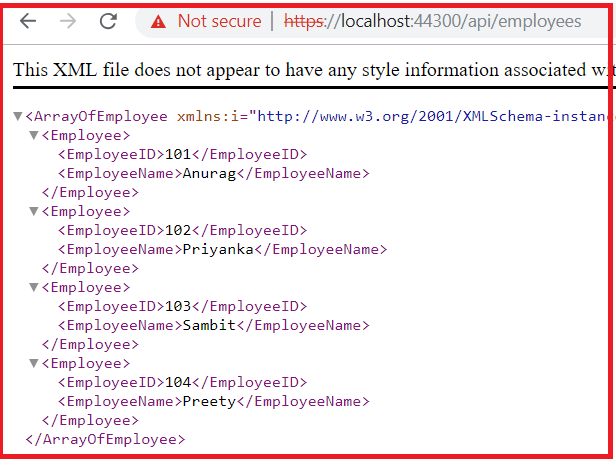
At this point, build the solution and then navigate to **https://localhost:44300/api/employees** URL in the browser and you will see the following browser security page. Make sure you click on the “**Advanced**” link to see the “**Proceed to localhost**” link.



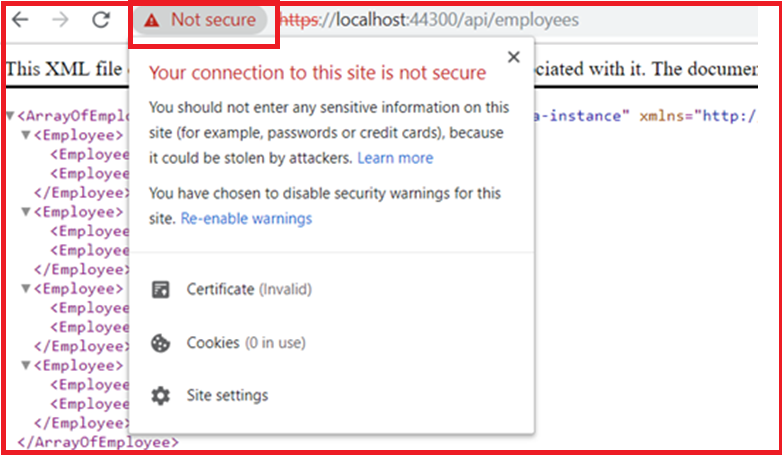
**Once you click on the above Advanced tab it opens the following section within the same page.**



**Once you click on the Proceed to localhost (unsafe) tab, it will give you the response as shown in the image below.**



**As shown in the above image, once you click on the Not Secure link, you will see that the certificate is invalid as shown below.**

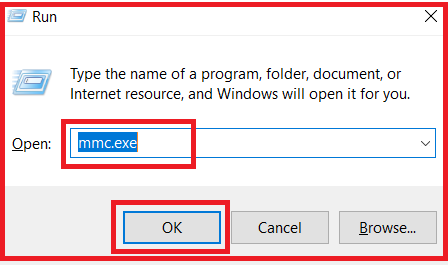


The reason is that the certificate that Visual Studio installed automatically is not trusted. To solve the above problem, what we need to do is, we need to place the certificate that visual studio has issued in the Trusted Root Certificates folder

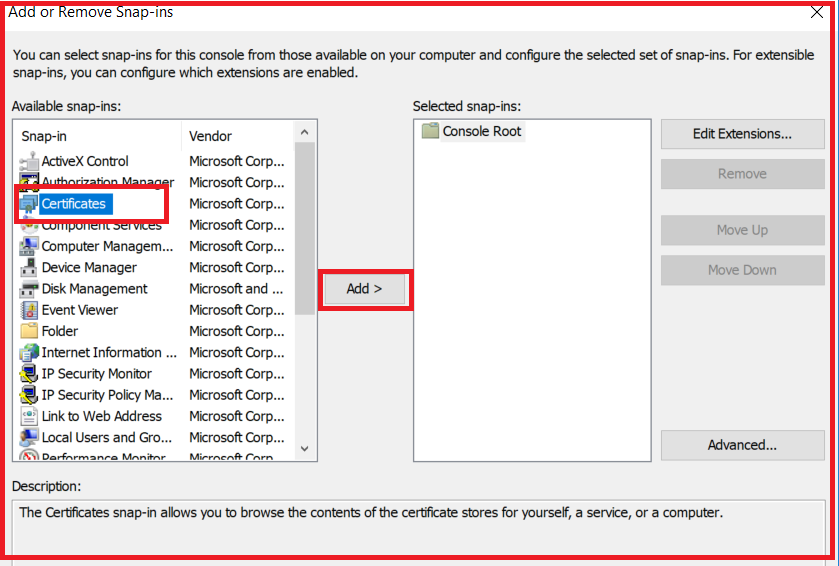
**Generating a Trusted Certificate:**

In order to use a trusted certificate, please follow the below steps

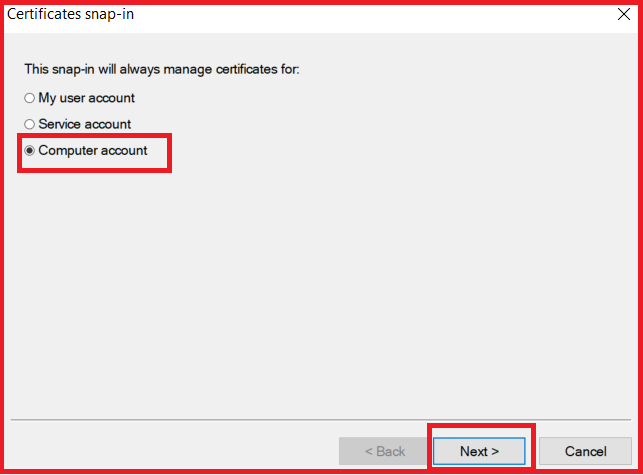
Open the RUN window, then type **mmc.exe** and click on the **OK** button as shown below



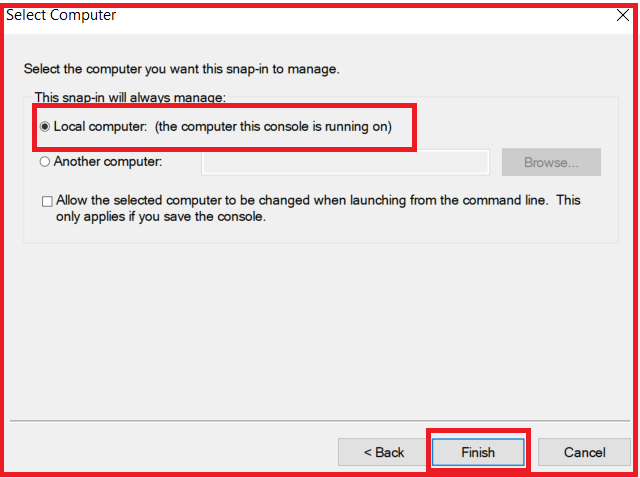
When you click on the OK button, one window will open, click “**File**” => “**Add/Remove Snap-in**” from that window and then from the “**Available snap-ins**” list select the “**Certificates**” and click on the “**Add**” button as shown in the below image



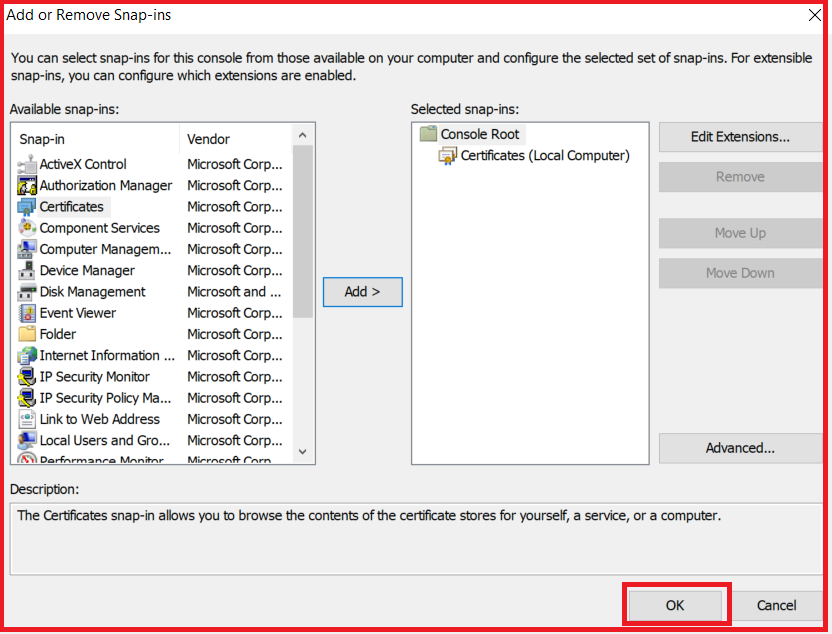
Once you click on the Add button it will open another screen from where select the “**Computer account**” radio button and then click on the Next button as shown below



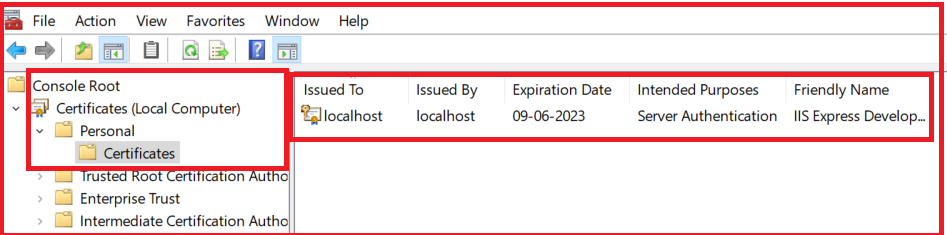
When you click on the Next button, it will open another screen and from that screen select the “**Local computer**” radio button and click on the “**Finish**” button as shown below.



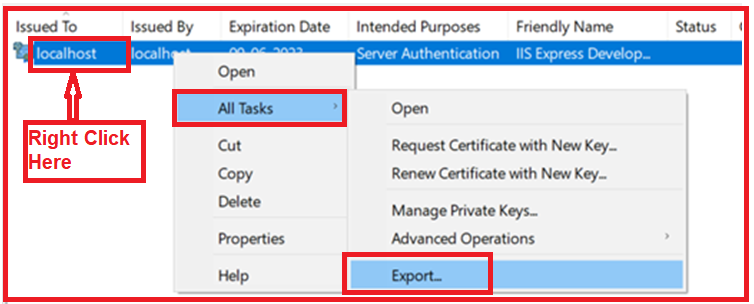
Once you click on the **Finish** button, it will take you back to the **Add or Remove Snap-ins** screen and from there click on the **OK** button as shown in the below image.



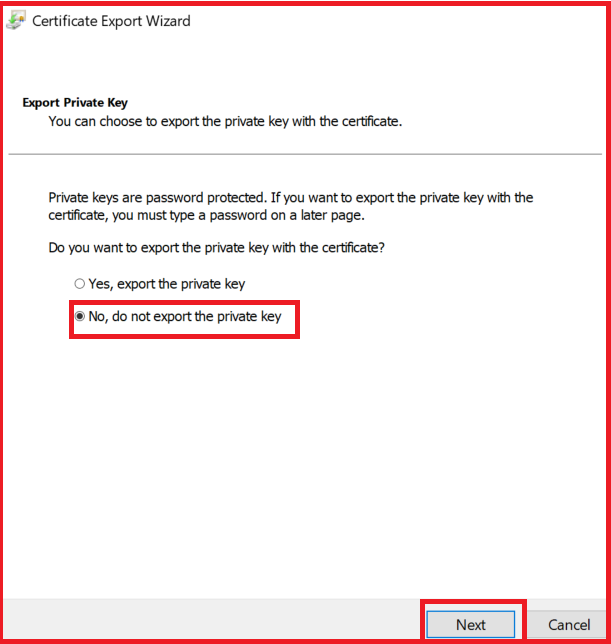
Expand the Console **Root => Certificates (Local Computer) => Personal => Certificates** folder and you will find a certificate that is Issued To localhost and Issued By localhost as shown in the image below.



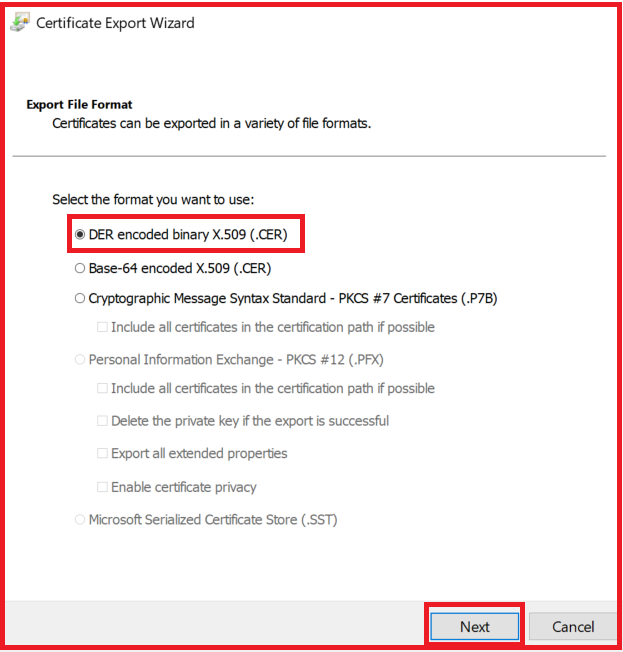
Right click on the localhost certificate, and then select “**All Tasks**” and then click on the “**Export**” option as shown in the image below.



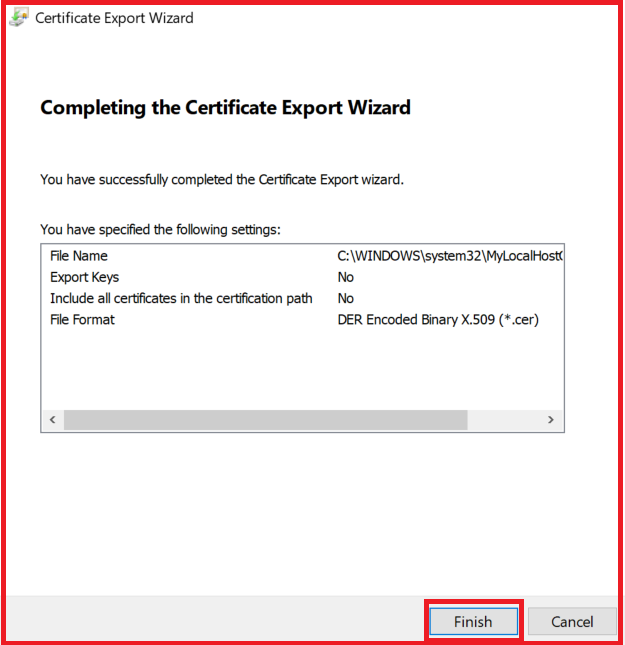
Once you click on the Export option, it will open the welcome to Welcome to Certificate Export Wizard screen and from there just click on the “Next” button. From the next screen select the No, do not export the private key radio button and click on the Next radio button as shown below.



Once you click on the Next button, it will open the select File Format wizard and from that wizard select the “**DER encoded binary X.509 (.CER)**” radio button, and click on the Next button as shown in the below image.



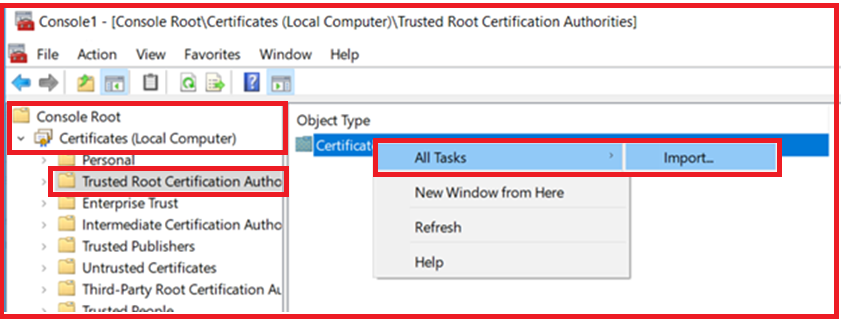
From the next screen, provide a meaningful name (in my case I have given **MyLocalhostCertificate**) for the certificate that you are exporting and then click on the “Next” button. Once you click on the Next button, it will open the following window from there just click on the Finish button. Please remember the path where your certificate is stored. In my case, it is **C:\Windows\System32\ MyLocalhostCertificate**



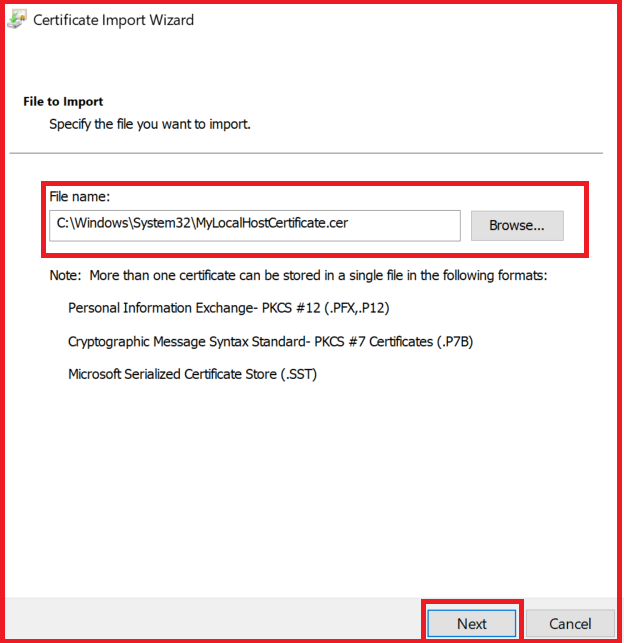
Once you click on the **Finish** button, if everything is ok, then you will get the message Export Successful.

**How to Import the newly Generated Certificate in the Trusted Root Certification Folder?**

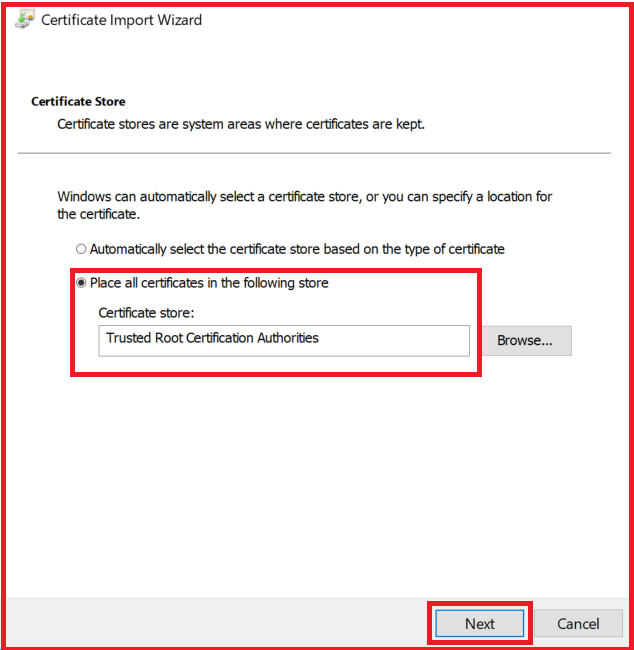
Expand the Console **Root – Certificates (Local Computer) – Trusted Root Certification Authorities – Certificates**. And then right click on the “**Certificates**“, and select “**All Tasks**” and then select the “**Import**” option as shown below.



Click on the “**Next**” button on the subsequent screen. In the next screen, you have to enter the complete path where you have exported the certificate and then click on the click “**Next**” as shown below. In my case, the certificate is at **C:\Windows\System32\MyLocalhostCertificate.cer**



Once you click on the **Next** button, it will open another screen and from that screen select “**Place all certificates in the following store**” radio button and click on the “**Next**” button as shown below

  
Finally, click on the “Finish” button which will give you one message that import was successful. So that’s it. We have created and import the certificate for localhost in the trusted certificate location.

Now first close all the instances of the browser. Open a new browser instance and navigate to **https://localhost:44300/api/employees** and you will not get any certificate error. At the moment we can access the web API service using both HTTP and https.

# Enable HTTPS in Web API

## ****Enable HTTPS in Web API Service****

In this article, I am going to discuss **How to enable HTTPS in Web API Service** with an example. In our **previous article**, we discussed [**how to enable SSL in Visual Studio Development Server**](https://dotnettutorials.net/lesson/enable-ssl-in-visual-studio-development-server/). Please read our previous article before proceeding to this article as we are going to work with the same example that we worked in our previous article.

At the moment, we can use both the HTTP and HTTPS to invoke the Web API resources as shown below and both the URI will give you the same result.

**http://localhost:55486/api/employees**

**https://localhost:44300/api/employees**

In this article, we are going to discuss how to enable HTTPS in Web API Service means once we enabled the HTTPS, if a request is issued using the HTTP then we want that request to be automatically redirected to HTTPS.

**Point to Remember:** If you are coming from the ASP.NET MVC background, then you may be tempted to use the built-in **RequireHttpsAttribute** but the sad thing is that this attribute is not supported in Web API.

##### ****How to enable HTTPS in Web API Service?****

You need to follow the below two steps to enable HTTPS in Web API.

**Step1:**

Right click on the Models Folder and add a class file with the name **CustomRequireHttpsAttribute** and then copy and paste the following code.

**using** *System;*

**using** *System.Net;*

**using** *System.Net.Http;*

**using** *System.Text;*

**using** *System.Web.Http.Controllers;*

**using** *System.Web.Http.Filters;*

**namespace** *WebAPIEnableHTTPS.Models*

**{**

**public** **class** CustomRequireHttpsAttribute : AuthorizationFilterAttribute

**{**

**public** **override** **void** OnAuthorization**(**HttpActionContext actionContext**)**

**{**

//both the request is not https

**if** **(**actionContext.Request.RequestUri.Scheme != Uri.UriSchemeHttps**)**

**{**

actionContext.Response = actionContext.Request.CreateResponse**(**HttpStatusCode.Found**)**;

actionContext.Response.Content = new StringContent

**(**"<p>Use https instead of http</p>", Encoding.UTF8, "text/html"**)**;

//Create the request URI

UriBuilder uriBuilder = new UriBuilder**(**actionContext.Request.RequestUri**)**;

//Set the Request scheme as HTTPS

uriBuilder.Scheme = Uri.UriSchemeHttps;

//Set the HTTPS Port number as 44300

//In the project properties window you can find the port number

//for SSL URL

uriBuilder.Port = 44300;

actionContext.Response.Headers.Location = uriBuilder.Uri;

**}**

**else**

**{**

**base**.OnAuthorization**(**actionContext**)**;

**}**

**}**

**}**

**}**

###### **Step2:**

You need to register the **CustomRequireHttpsAttribute** in the **Register()** method of the **WebApiConfig class** in **WebApiConfig.cs** file which is present in **the App\_Start**folder as shown below.

How to enable HTTPS in Web API

The above line of code will add the **CustomRequireHttpsAttribute** as a global filter to the filters collection as a result for every incoming request the code which is present in this filter is going to be executed. So, if the request is issued using HTTP, then it will be automatically redirected to HTTPS.

**The complete code of the WebApiConfig.cs file is given below.**

**using** *System.Web.Http;*

**using** *WebAPIEnableHTTPS.Models;*

**namespace** *WebAPIEnableHTTPS*

**{**

**public** **static** **class** WebApiConfig

**{**

**public** **static** **void** Register**(**HttpConfiguration config**)**

**{**

config.Filters.Add**(**new CustomRequireHttpsAttribute**())**;

config.MapHttpAttributeRoutes**()**;

config.Routes.MapHttpRoute**(**

name: "DefaultApi",

routeTemplate: "api/{controller}/{id}",

defaults: new **{** id = RouteParameter.Optional **}**

**)**;

**}**

**}**

**}**

Now, build the solution and navigate to the following URL.

**http://localhost:55486/api/employees**

Once you hit the browser you will see that the above URL is transmitted to the below URL

**https://localhost:44300/api/employees**

**Note:** If you don’t want to enable the HTTPS for the entire application, then don’t add the **CustomRequireHttpsAttribute** to the filters collection on the config object in the register method of the WebApiConfig class.

What you need to do is, decorate the controller class or the action method with **CustomRequireHttpsAttribute** for which you want the HTTPS to be enabled. For the rest of the controllers and action methods, HTTPS will not be enabled.