ASP.NET Web APIs | Rest API's with .NET and C#

http verbs-get-post-put-delete-patch-head-jsonp

HTTP (request methods - GET, PUT, POST, DELETE)

HTTP and Interacting with the server

There are different types of APIS that we create like

Get API

Post API

Delete API

Put API

MVC frameworks

Page Request Lifecycle.

Calling ASP.NET Web API service in a cross-domain using jQuery ajax.

SOAP and XML created an excellent solution for creating connected web applications. SOAP is a standard XML based protocol that communicated over HTTP.

The SOAP offered an excellent way of transferring the data between the applications

Web API can automatically serialize the model data to JSON, XML.

Web API Controller is responsible for handling all HTTP requests which can come from a browser, mobile device, desktop web application or any other.

ASP.NET Web API framework is widely used to create RESTful service

REST stands for Representational State Transfer.

REST is an architectural pattern for creating an API that uses HTTP as its underlying communication method. This is a protocol for exchanging data over a distributed environment. The main idea behind REST is that we should treat our distributed services as a resource and we should be able to use simple HTTP protocols to perform various operations on that resource.

One of the choices available in .NET for creating RESTful services is WCF. The problem with WCF is that a lot of configuration is required to turn a WCF service into a RESTful service. The more natural choice for creating RESTful services is ASP.NET Web API,

CRUD HTTP Verb

Create POST

Read GET

Update PUT

Delete DELETE

Post(), Put() and Delete() methods in our ValuesController are returning void. That is the reason we are getting status code 204 No Content. An action that returns void will send status code 204 No Content.

GET: This maps to the R(Retrieve) part of the CRUD operation. This will be used to retrieve the required data (representation of data) from the remote resource.

PUT: This maps to the U(Update) part of the CRUD operation. This protocol will update the current representation of the data on the remote server.

POST: This maps to the C(Create) part of the CRUD operation. This will create a new entry for the current data that is being sent to the server.

DELETE: This maps to the D(Delete) part of the CRUD operation. This will delete the specified data from the remote server.

What is Entity Framework

Entity Framework is an ORM framework. ORM stands for Object Relational Mapping.

What is Object Relational Mapping framework

Object Relational Mapping framework automatically creates classes based on database tables, and the vice versa is also true, that is,

it can also automatically generate necessary SQL to create database tables based on classes.

To perform the Database operations within the service we use Entity framework

For both local and social login, Web API uses OAuth2 to authenticate requests.

Web API with MVC Project

Stand-alone Web API Project

configuration folder

controller folder

global.asax register

Flow-

create model

data access

controller

model

controller-data

api

Browsers allow a web page to make AJAX requests only with in the same domain. Browser security prevents a web page from making AJAX requests to another domain. This is called same origin policy.

Get API Example1

Create a model

Employee.cs

public class Employee

{

public int EmployeeId

{

get;

set;

}

public string EmployeeName

{

get;

set;

}

public string Address

{

get;

set;

}

public string Department

{

get;

set;

}

}

Add controller ->

EmployeeController.cs

public class EmployeeController : ApiController

{

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

{

new Employee()

{

EmployeeId = 1, EmployeeName = "Mukesh Kumar", Address = "New Delhi", Department = "IT"

},

new Employee()

{

EmployeeId = 2, EmployeeName = "Banky Chamber", Address = "London", Department = "HR"

},

new Employee()

{

EmployeeId = 3, EmployeeName = "Rahul Rathor", Address = "Laxmi Nagar", Department = "IT"

},

new Employee()

{

EmployeeId = 4, EmployeeName = "YaduVeer Singh", Address = "Goa", Department = "Sales"

},

new Employee()

{

EmployeeId = 5, EmployeeName = "Manish Sharma", Address = "New Delhi", Department = "HR"

},

};

public IList<Employee> GetAllEmployees()

{

//Return list of all employees

return employees;

}

public Employee GetEmployeeDetails(int id)

{

//Return a single employee detail

var employee = employees.FirstOrDefault(e => e.EmployeeId == id);

if (employee == null)

{

throw new HttpResponseException(Request.CreateResponse(HttpStatusCode.NotFound));

}

return employee;

}

}

}

it creates 2 APIs one Is to get all employees list and another one is to get details of particular employee.

http://localhost:52334/api/employee

http://localhost:52334/api/employee/2

Example2-

Add properties on Model

namespace WebAPIDemo.Models {

public class Employee{

public int ID { get; set; }

public string Name { get; set; }

public DateTime JoiningDate { get; set; }

public int Age { get; set; }

}

}

Controller-

namespace WebAPIDemo.Controllers{

public class EmployeesController : ApiController{

Employee[] employees = new Employee[]{

new Employee { ID = 1, Name = "Mark", JoiningDate =

DateTime.Parse(DateTime.Today.ToString()), Age = 30 },

new Employee { ID = 2, Name = "Allan", JoiningDate =

DateTime.Parse(DateTime.Today.ToString()), Age = 35 },

new Employee { ID = 3, Name = "Johny", JoiningDate =

DateTime.Parse(DateTime.Today.ToString()), Age = 21 }

};

public IEnumerable<Employee> GetAllEmployees(){

return employees;

}

public IHttpActionResult GetEmployee(int id){

var employee = employees.FirstOrDefault((p) => p.ID == id);

if (employee == null){

return NotFound();

}

return Ok(employee);

}

}

http://localhost:52334/api/employee

http://localhost:52334/api/employee/2

Example3-

Data imported from ado.net entity model-entity framework (From SQL Server.

Create employee service entity model.

Employeescontroller.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

namespace EmployeeService.Controllers

{

public class EmployeesController : ApiController

{

public IEnumerable<Employee> Get()

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

return entities.Employees.ToList();

}

}

public Employee Get(int id)

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

return entities.Employees.FirstOrDefault(e => e.ID == id);

}

}

}

}

Create DB and insert data to table-

Create Database EmployeeDB

Create table Employees

(

ID int primary key identity,

FirstName nvarchar(50),

LastName nvarchar(50),

Gender nvarchar(50),

Salary int

)

Insert into Employees values ('Mark', 'Hastings', 'Male', 60000)

Insert into Employees values ('Steve', 'Pound', 'Male', 45000)

Insert into Employees values ('Ben', 'Hoskins', 'Male', 70000)

Insert into Employees values ('Philip', 'Hastings', 'Male', 45000)

Insert into Employees values ('Mary', 'Lambeth', 'Female', 30000)

Insert into Employees values ('Valarie', 'Vikings', 'Female', 35000)

Insert into Employees values ('John', 'Stanmore', 'Male', 80000)

http://localhost:52334/api/employee

http://localhost:52334/api/employee/2

To print html data or json format data-

Add this code on webconfig.cs

config.Formatters.JsonFormatter.SupportedMediaTypes.Add(new MediaTypeHeaderValue("text/html"));

To print data in xml format-

Add these methods in webconfig.cs

public class CustomJsonFormatter : JsonMediaTypeFormatter

{

public CustomJsonFormatter()

{

this.SupportedMediaTypes.Add(new MediaTypeHeaderValue("text/html"));

}

public override void SetDefaultContentHeaders(Type type, HttpContentHeaders headers, MediaTypeHeaderValue mediaType)

{

base.SetDefaultContentHeaders(type, headers, mediaType);

headers.ContentType = new MediaTypeHeaderValue("application/json");

}

Example post API when SQL server is connected-

Employeecontroller.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

namespace EmployeeService.Controllers

{

public class EmployeesController : ApiController

{

public IEnumerable<Employee> Get()

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

return entities.Employees.ToList();

}

}

public Employee Get(int id)

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

return entities.Employees.FirstOrDefault(e => e.ID == id);

}

}

public void Post([FromBody] Employee employee)

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

entities.Employees.Add(employee);

entities.SaveChanges();

}

}

}

}

[Headers]Content-Type: application/json. This tells that we are sending JSON formatted data to the server

[Body-raw]In the Request Body write the data-{"FirstName":"sam","Lastname":"Wicht","Gender":"Male","Salary":54000}

It is executed and add data to SQL database.

It gives a message http 204 no content because it return void so we need to change the void status to 200 content created so add this code that is written below.

public HttpResponseMessage Post([FromBody] Employee employee)

{

try

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

entities.Employees.Add(employee);

entities.SaveChanges();

var message = Request.CreateResponse(HttpStatusCode.Created, employee);

message.Headers.Location = new Uri(Request.RequestUri +

employee.ID.ToString());

return message;

}

}

catch (Exception ex)

{

return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex);

}

}

Delete API Example-

public void Delete(int id)

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

entities.Employees.Remove(entities.Employees.FirstOrDefault(e => e.ID == id));

entities.SaveChanges();

}

}

When the deletion is successful, since the method return type is void we get status code 204 No Content. We should be returning status code 200 OK.

Public HttpResponseMessage Delete(int id)

{

try

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

var entity = entities.Employees.FirstOrDefault(e => e.ID == id);

if (entity == null)

{

return Request.CreateErrorResponse(HttpStatusCode.NotFound,

"Employee with Id = " + id.ToString() + " not found to delete");

}

else

{

entities.Employees.Remove(entity);

entities.SaveChanges();

return Request.CreateResponse(HttpStatusCode.OK);

}

}

}

catch (Exception ex)

{

return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex);

}

}

Example put API or update API-

It basically update the data-

public void Put(int id, [FromBody]Employee employee)

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

var entity = entities.Employees.FirstOrDefault(e => e.ID == id);

entity.FirstName = employee.FirstName;

entity.LastName = employee.LastName;

entity.Gender = employee.Gender;

entity.Salary = employee.Salary;

entities.SaveChanges();

}

}

{"FirstName":"sam1","Lastname":"Wicht11","Gender":"Male","Salary":54000}

http://localhost:54390/api/employees/8

Gives 204 no content

public HttpResponseMessage Put(int id, [FromBody]Employee employee)

{

try

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

var entity = entities.Employees.FirstOrDefault(e => e.ID == id);

if (entity == null)

{

return Request.CreateErrorResponse(HttpStatusCode.NotFound,

"Employee with Id " + id.ToString() + " not found to update");

}

else

{

entity.FirstName = employee.FirstName;

entity.LastName = employee.LastName;

entity.Gender = employee.Gender;

entity.Salary = employee.Salary;

entities.SaveChanges();

return Request.CreateResponse(HttpStatusCode.OK, entity);

}

}

}

catch (Exception ex)

{

return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex);

}

}

It returns error if condition is wrong and give 200 code content created.

ASP.NET Web API query string parameters-

ttp://localhost/api/employees?gender=All All Employees

http://localhost/api/employees?gender=Male Only Male Employees

http://localhost/api/employees?gender=Female Only Female Employees

public HttpResponseMessage Get(string gender = "All")

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

switch (gender.ToLower())

{

case "all":

return Request.CreateResponse(HttpStatusCode.OK,

entities.Employees.ToList());

case "male":

return Request.CreateResponse(HttpStatusCode.OK,

entities.Employees.Where(e => e.Gender.ToLower() == "male").ToList());

case "female":

return Request.CreateResponse(HttpStatusCode.OK,

entities.Employees.Where(e => e.Gender.ToLower() == "female").ToList());

default:

return Request.CreateErrorResponse(HttpStatusCode.BadRequest,

"Value for gender must be Male, Female or All. " + gender + " is invalid.");

}

}

}

By both frombody and fromuri we can update the data-

These are the ASP.NET Web API query string parameters

http://localhost/api/employees??FirstName=am&Lastname=kWicht&Gender=Male&Salary=54000 ->we can change the data from url by using fromuri Attribute.

public HttpResponseMessage Put([FromBody]int id, [FromUri]Employee employee)

{

try

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

var entity = entities.Employees.FirstOrDefault(e => e.ID == id);

if (entity == null)

{

return Request.CreateErrorResponse(HttpStatusCode.NotFound,

"Employee with Id " + id.ToString() + " not found to update");

}

else

{

entity.FirstName = employee.FirstName;

entity.LastName = employee.LastName;

entity.Gender = employee.Gender;

entity.Salary = employee.Salary;

entities.SaveChanges();

return Request.CreateResponse(HttpStatusCode.OK, entity);

}

}

}

catch (Exception ex)

{

return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex);

}

}

Call WEB API from Jquery-

Add Employees.html file in your project-

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta charset="utf-8" />

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

<script type="text/javascript">

$(document).ready(function () {

var ulEmployees = $('#ulEmployees');

$('#btn').click(function () {

$.ajax({

type: 'GET',

url: "api/employees/",

dataType: 'json',

success: function (data) {

ulEmployees.empty();

$.each(data, function (index, val) {

var fullName = val.FirstName + ' ' + val.LastName;

ulEmployees.append('<li>' + fullName + '</li>');

});

}

});

});

$('#btnClear').click(function () {

ulEmployees.empty();

});

});

</script>

</head>

<body>

<div>

<input id="btn" type="button" value="Get All Employees" />

<input id="btnClear" type="button" value="Clear" />

<ul id="ulEmployees" />

</div>

</body>

</html>

Calling ASP.NET Web API service in a cross domain using jQuery ajax-

There are 2 ways

1-Using JSONP (JSON with Padding)

2-Enabling CORS (Cross Origin Resource Sharing)

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta charset="utf-8" />

<script src="Scripts/jquery-1.10.2.js"></script>

<script type="text/javascript">

$(document).ready(function () {

var ulEmployees = $('#ulEmployees');

$('#btn').click(function () {

$.ajax({

type: 'GET',

// Make sure to change the port number to

// where you have the employee service

// running on your local machine

url: 'http://localhost:23258/api/Employees',

dataType: 'jsonp', //changed

success: function (data) {

ulEmployees.empty();

$.each(data, function (index, val) {

var fullName = val.FirstName + ' ' + val.LastName;

ulEmployees.append('<li>' + fullName + '</li>')

});

}

});

});

$('#btnClear').click(function () {

ulEmployees.empty();

});

});

</script>

</head>

<body>

<input id="btn" type="button" value="Get All Employees" />

<input id="btnClear" type="button" value="Clear" />

<ul id="ulEmployees"></ul>

</body>

</html>

s1-

Nuget pacakge

Install-Package WebApiContrib.Formatting.Jsonp

s2-

var jsonpFormatter = new JsonpMediaTypeFormatter(config.Formatters.JsonFormatter);

config.Formatters.Insert(0, jsonpFormatter);

If you want JSONP formatted data back, set Accept header to application/javascript and specify a name for the callback function in the URI. We have set it to ABC.

http://localhost:23258/api/Employees?callback=ABC

Enabling cors-

s1-Nuget package

Install-Package Microsoft.AspNet.WebApi.Cors

s2-Include this code in webconfig.cs

EnableCorsAttribute cors = new EnableCorsAttribute("\*", "\*", "\*");

config.EnableCors();.

press F4 key for properties of main project.

enable SSL property to true.

Web API enable HTTPS-

RequireHttpsAttribute-->Add class to your project

using System;

using System.Net;

using System.Net.Http;

using System.Text;

using System.Web.Http.Controllers;

using System.Web.Http.Filters;

namespace EmployeeService

{

public class RequireHttpsAttribute : AuthorizationFilterAttribute

{

public override void OnAuthorization(HttpActionContext actionContext)

{

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");

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

uriBuilder.Scheme = Uri.UriSchemeHttps;

uriBuilder.Port = 44337;

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

}

else

{

base.OnAuthorization(actionContext);

}

}

}

}

Add this in webconfig-

config.Filters.Add(new RequireHttpsAttribute());

Implementing basic authentication in ASP.NET Web API-

craete a table of users

Create Table Users

(

Id int identity primary key,

Username nvarchar(100),

Password nvarchar(100)

)

Insert into Users values ('male','male')

Insert into Users values ('female','female')

Create a class that checks if the username and password are valid

EmployeeSecurity.cs

using EmployeeDataAccess;

using System;

using System.Linq;

namespace EmployeeService

{

public class EmployeeSecurity

{

public static bool Login(string username, string password)

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

return entities.Users.Any(user =>

user.Username.Equals(username, StringComparison.OrdinalIgnoreCase)

&& user.Password == password);

}

}

}

}

Create basic authentication filter

BasicAuthenticationAttribute.cs

using System;

using System.Net;

using System.Net.Http;

using System.Security.Principal;

using System.Text;

using System.Threading;

using System.Web.Http.Controllers;

using System.Web.Http.Filters;

namespace EmployeeService

{

public class BasicAuthenticationAttribute : AuthorizationFilterAttribute

{

public override void OnAuthorization(HttpActionContext actionContext)

{

if (actionContext.Request.Headers.Authorization == null)

{

actionContext.Response = actionContext.Request

.CreateResponse(HttpStatusCode.Unauthorized);

}

else

{

string authenticationToken = actionContext.Request.Headers

.Authorization.Parameter;

string decodedAuthenticationToken = Encoding.UTF8.GetString(

Convert.FromBase64String(authenticationToken));

string[] usernamePasswordArray = decodedAuthenticationToken.Split(':');

string username = usernamePasswordArray[0];

string password = usernamePasswordArray[1];

if (EmployeeSecurity.Login(username, password))

{

Thread.CurrentPrincipal = new GenericPrincipal(

new GenericIdentity(username), null);

}

else

{

actionContext.Response = actionContext.Request

.CreateResponse(HttpStatusCode.Unauthorized);

}

}

}

}

}

Add code on controller-

EmployeesController

[BasicAuthentication]

public HttpResponseMessage Get(string gender = "All")

{

string username = Thread.CurrentPrincipal.Identity.Name;

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

switch (username.ToLower())

{

case "male":

return Request.CreateResponse(HttpStatusCode.OK,

entities.Employees.Where(e => e.Gender.ToLower() == "male").ToList());

case "female":

return Request.CreateResponse(HttpStatusCode.OK,

entities.Employees.Where(e => e.Gender.ToLower() == "female").ToList());

default:

return Request.CreateResponse(HttpStatusCode.BadRequest);

}

}

}

to check this when you run the get api

http://localhost:54390/api/employees you need to choose the type-basic auth then pass the username and password then you get filtered data according to it.

ASP.NET Web API token authentication-

This only access by authorised user if you use Authorize attribute

it gives an error 401 unauthoriser

http://localhost:54390/api/employees

[Authorize]

public IEnumerable<Employee> Get()

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

return entities.Employees.ToList();

}

}

If you call this api then you do not get any data.

It is post api

create a new WebApi Project-

Add a file Register.html

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta charset="utf-8" />

<link href="Content/bootstrap.min.css" rel="stylesheet" />

</head>

<body style="padding-top:20px">

<div class="col-md-10 col-md-offset-1">

<div class="well">

<!--This table contains the fields that we want to capture to register a new user-->

<table class="table table-bordered">

<thead>

<tr class="success">

<th colspan="2">

New User Registration

</th>

</tr>

</thead>

<tbody>

<tr>

<td>Email</td>

<td><input type="text" id="txtEmail" placeholder="Email" /> </td>

</tr>

<tr>

<td>Password</td>

<td>

<input type="password" id="txtPassword"

placeholder="Password" />

</td>

</tr>

<tr>

<td>Confirm Password</td>

<td>

<input type="password" id="txtConfirmPassword"

placeholder="Confirm Password" />

</td>

</tr>

<tr class="success">

<td colspan="2">

<input id="btnRegister" class="btn btn-success"

type="button" value="Register" />

</td>

</tr>

</tbody>

</table>

<!--Bootstrap modal dialog that shows up when regsitration is successful-->

<div class="modal fade" tabindex="-1" id="successModal"

data-keyboard="false" data-backdrop="static">

<div class="modal-dialog modal-sm">

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal">

&times;

</button>

<h4 class="modal-title">Success</h4>

</div>

<div class="modal-body">

<form>

<h2 class="modal-title">Registration Successful!</h2>

</form>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-success"

data-dismiss="modal">

Close

</button>

</div>

</div>

</div>

</div>

<!--Bootstrap alert to display any validation errors-->

<div id="divError" class="alert alert-danger collapse">

<a id="linkClose" href="#" class="close">&times;</a>

<div id="divErrorText"></div>

</div>

</div>

</div>

<script src="Scripts/jquery-1.10.2.min.js"></script>

<script src="Scripts/bootstrap.min.js"></script>

<script type="text/javascript">

$(document).ready(function () {

//Close the bootstrap alert

$('#linkClose').click(function () {

$('#divError').hide('fade');

});

// Save the new user details

$('#btnRegister').click(function () {

$.ajax({

url: '/api/account/register',

method: 'POST',

data: {

email: $('#txtEmail').val(),

password: $('#txtPassword').val(),

confirmPassword: $('#txtConfirmPassword').val()

},

success: function () {

$('#successModal').modal('show');

},

error: function (jqXHR) {

$('#divErrorText').text(jqXHR.responseText);

$('#divError').show('fade');

}

});

});

});

</script>

</body>

</html>

http://localhost:62594/api/account/register

Change the type to post request and add the parameters in x-wwww-from-encoded

key value

email-sun@gmail.com

password-$UUny123

confirmPassword-$UUny123

It add the the data in app\_data folder in project

It created a default database in the visual studio its string is

<connectionStrings>

<add name="DefaultConnection" connectionString="Data Source=(LocalDb)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\aspnet-qweer-20170718053936.mdf;Initial Catalog=aspnet-qweer-20170718053936;Integrated Security=True" providerName="System.Data.SqlClient" />

</connectionStrings>

To see this data add click on the showfiles button in project.

........................

-----------------------------------------------------------------------------------------------------------------------

change this url to this to make a different database with dbname userdb

<add name="DefaultConnection"

connectionString="Data Source=(LocalDb)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\UsersDB.mdf;Initial Catalog=UsersDB;Integrated Security=True"

providerName="System.Data.SqlClient" />

----------------------------------------------------------------------------------------

change the connection string to this to make db on sqlserver

<add name="DefaultConnection"

connectionString="Data Source=(local);Initial Catalog=UsersDB;Integrated Security=True"

providerName="System.Data.SqlClient" />

...

<add name="DefaultConnection"

connectionString="Data Source=(local);Initial Catalog=EmployeeDB;Integrated Security=True"

providerName="System.Data.SqlClient" />

----------------------------------------------------------------------------------------------------

How To access Authiorize attribute data-

s1-First is you need to create token

s2-Issue a POST request to /token

http://localhost:62594/token

s3-In the request body include username and the password.

username=test@gmail.com&password=Test123!&grant\_type=password

We also need to set grant\_type=password. This indicates that we are presenting password for acquiring access token.

s4-Then it generated the access token

when you access the api

http://localhost:54390/api/employees then tou get error 401 unauthorised

To access this -pass get request

http://localhost:54390/api/employees write the api

and pass the header

Authorization:Bearer Accesstokendata(dscfdscd)

---------------------------------------------------------------------

signin and signout

1. sessionStorage data is lost when the browser window is closed.

2. To store an item in the browser session storage use setItem() method

Example : sessionStorage.setItem("accessToken", response.access\_token)

3. To retrieve an item from the browser session storage use getItem() method

Example : sessionStorage.getItem("accessToken")

4. To remove an item from the browser session storage use removeItem() method

Example : sessionStorage.removeItem('accessToken')

------------------------------------------------------

url encode decode

https://console.developers.google.com

----------------------------------------------

Attribute routing in ASP.NET Web API 2-

StudentsController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

using demoapi.Models;

namespace demoapi.Controllers

{

public class StudentsController : ApiController

{

static List<Student> students = new List<Student>()

{

new Student() { Id = 1, Name = "Tom" },

new Student() { Id = 2, Name = "Sam" },

new Student() { Id = 3, Name = "John" }

};

public IEnumerable<Student> Get()

{

return students;

}

public Student Get(int id)

{

return students.FirstOrDefault(s => s.Id == id);

}

[Route("api/students/{id}/courses")] //Attribute routing

public IEnumerable<string> GetStudentCourses(int id)

{

if (id == 1)

return new List<string>() { "C#", "ASP.NET", "SQL Server" };

else if (id == 2)

return new List<string>() { "ASP.NET Web API", "C#", "SQL Server" };

else

return new List<string>() { "Bootstrap", "jQuery", "AngularJs" };

}

}

}

........................

student.cs

public int Id { get; set; }

public string Name { get; set; }

/api/students/1

/api/students/1/courses

--------------------------------------------------------------------------

enable SSL property to true.

Web API enable HTTPS.

ASP.NET Web API token authentication-

Create Simple Web API In ASP.NET Web API

get api

post api

delete api

put api

ASP.NET Web API query string parameters

By both frombody and fromuri we can update the data

Call webapi from jquery-

Calling ASP.NET Web API service in a cross domain using jQuery ajax-

enable ssl property to true

Web API enable HTTPS

ASP.NET Web API token authentication-

------------------------------------------------------------------------------------------

SOAP and XML created an excellent solution for creating connected web applications. SOAP is a standard XML based protocol that communicated over HTTP.

The SOAP offered an excellent way of transferring the data between the applications

Web API can automatically serialize the model data to JSON, XML.

Web API Controller is responsible for handling all HTTP requests which can come from browser, mobile device, desktop web application or any other.

ASP.NET Web API framework is widely used to create RESTful service

REST stands for Representational State Transfer.

REST is an architectural pattern for creating an API that uses HTTP as its underlying communication method. This is a protocol for exchanging data over a distributed environment. The main idea behind REST is that we should treat our distributed services as a resource and we should be able to use simple HTTP protocols to perform various operations on that resource.

One of the choices available in .NET for creating RESTful services is WCF. The problem with WCF is that, a lot of configuration is required to turn a WCF service into a RESTful service. The more natural choice for creating RESTful services is ASP.NET Web API,

CRUD HTTP Verb

Create POST

Read GET

Update PUT

Delete DELETE

Post(), Put() and Delete() methods in our ValuesController are returning void. That is the reason we are getting status code 204 No Content. An action that returns void will send status code 204 No Content.

GET: This maps to the R(Retrieve) part of the CRUD operation. This will be used to retrieve the required data (representation of data) from the remote resource.

PUT: This maps to the U(Update) part of the CRUD operation. This protocol will update the current representation of the data on the remote server.

POST: This maps to the C(Create) part of the CRUD operation. This will create a new entry for the current data that is being sent to the server.

DELETE: This maps to the D(Delete) part of the CRUD operation. This will delete the specified data from the remote server.

To perform the Database operations within the service we use Entity framework

For both local and social login, Web API uses OAuth2 to authenticate requests.

Web API with MVC Project

Stand-alone Web API Project

configuration folder

controller folder

global.asax register

Flow-

create model

data access

controller

model

controller-data

api

Browsers allow a web page to make AJAX requests only with in the same domain. Browser security prevents a web page from making AJAX requests to another domain. This is called same origin policy.

--------------------------------------------------------------------------------------------------------------------get apis

Example1-

Create a model

-> Model -> Employee.cs

public class Employee

{

public int EmployeeId

{

get;

set;

}

public string EmployeeName

{

get;

set;

}

public string Address

{

get;

set;

}

public string Department

{

get;

set;

}

}

...........

-> Add controller -> EmployeeController.cs

public class EmployeeController : ApiController

{

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

{

new Employee()

{

EmployeeId = 1, EmployeeName = "Mukesh Kumar", Address = "New Delhi", Department = "IT"

},

new Employee()

{

EmployeeId = 2, EmployeeName = "Banky Chamber", Address = "London", Department = "HR"

},

new Employee()

{

EmployeeId = 3, EmployeeName = "Rahul Rathor", Address = "Laxmi Nagar", Department = "IT"

},

new Employee()

{

EmployeeId = 4, EmployeeName = "YaduVeer Singh", Address = "Goa", Department = "Sales"

},

new Employee()

{

EmployeeId = 5, EmployeeName = "Manish Sharma", Address = "New Delhi", Department = "HR"

},

};

public IList<Employee> GetAllEmployees()

{

//Return list of all employees

return employees;

}

public Employee GetEmployeeDetails(int id)

{

//Return a single employee detail

var employee = employees.FirstOrDefault(e => e.EmployeeId == id);

if (employee == null)

{

throw new HttpResponseException(Request.CreateResponse(HttpStatusCode.NotFound));

}

return employee;

}

}

}

it creates 2 apis

http://localhost:52334/api/employee

http://localhost:52334/api/employee/2

-----------------------------------------------------------------------------------------------------------------get api

Example2-

add properties on model

namespace WebAPIDemo.Models {

public class Employee{

public int ID { get; set; }

public string Name { get; set; }

public DateTime JoiningDate { get; set; }

public int Age { get; set; }

}

}

/...........................

namespace WebAPIDemo.Controllers{

public class EmployeesController : ApiController{

Employee[] employees = new Employee[]{

new Employee { ID = 1, Name = "Mark", JoiningDate =

DateTime.Parse(DateTime.Today.ToString()), Age = 30 },

new Employee { ID = 2, Name = "Allan", JoiningDate =

DateTime.Parse(DateTime.Today.ToString()), Age = 35 },

new Employee { ID = 3, Name = "Johny", JoiningDate =

DateTime.Parse(DateTime.Today.ToString()), Age = 21 }

};

public IEnumerable<Employee> GetAllEmployees(){

return employees;

}

public IHttpActionResult GetEmployee(int id){

var employee = employees.FirstOrDefault((p) => p.ID == id);

if (employee == null){

return NotFound();

}

return Ok(employee);

}

}

http://localhost:52334/api/employee

http://localhost:52334/api/employee/2

--------------------------------------------------------------------------------get api

Example3-

data imported from ado.net entity model-entity framework (From SQL Server)

Create employee service entity model

............

Employeescontroller.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

namespace EmployeeService.Controllers

{

public class EmployeesController : ApiController

{

public IEnumerable<Employee> Get()

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

return entities.Employees.ToList();

}

}

public Employee Get(int id)

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

return entities.Employees.FirstOrDefault(e => e.ID == id);

}

}

}

}

..................

Create Database EmployeeDB

Create table Employees

(

ID int primary key identity,

FirstName nvarchar(50),

LastName nvarchar(50),

Gender nvarchar(50),

Salary int

)

Insert into Employees values ('Mark', 'Hastings', 'Male', 60000)

Insert into Employees values ('Steve', 'Pound', 'Male', 45000)

Insert into Employees values ('Ben', 'Hoskins', 'Male', 70000)

Insert into Employees values ('Philip', 'Hastings', 'Male', 45000)

Insert into Employees values ('Mary', 'Lambeth', 'Female', 30000)

Insert into Employees values ('Valarie', 'Vikings', 'Female', 35000)

Insert into Employees values ('John', 'Stanmore', 'Male', 80000)

http://localhost:52334/api/employee

http://localhost:52334/api/employee/2

------------------------------------------------------------------------------------

to print html data or json format data

Add this code on webconfig.cs

config.Formatters.JsonFormatter.SupportedMediaTypes.Add(new MediaTypeHeaderValue("text/html"));

.....................

To print data in xml format add these methods in webconfig.cs

public class CustomJsonFormatter : JsonMediaTypeFormatter

{

public CustomJsonFormatter()

{

this.SupportedMediaTypes.Add(new MediaTypeHeaderValue("text/html"));

}

public override void SetDefaultContentHeaders(Type type, HttpContentHeaders headers, MediaTypeHeaderValue mediaType)

{

base.SetDefaultContentHeaders(type, headers, mediaType);

headers.ContentType = new MediaTypeHeaderValue("application/json");

}

--------------------------------------------------------------------------------------------post api-----sql server connected

Employeecontroller.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

namespace EmployeeService.Controllers

{

public class EmployeesController : ApiController

{

public IEnumerable<Employee> Get()

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

return entities.Employees.ToList();

}

}

public Employee Get(int id)

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

return entities.Employees.FirstOrDefault(e => e.ID == id);

}

}

public void Post([FromBody] Employee employee)

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

entities.Employees.Add(employee);

entities.SaveChanges();

}

}

}

}

[Headers]Content-Type: application/json. This tells that we are sending JSON formatted data to the server

[Body-raw]In the Request Body write the data-{"FirstName":"sam","Lastname":"Wicht","Gender":"Male","Salary":54000}

It is executed and add data to sql database.

it gives a message http 204 no content becsuse it return void so we need to change the void status to 200 content created so add this code that is written below

---------------------------------------------------------------------

public HttpResponseMessage Post([FromBody] Employee employee)

{

try

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

entities.Employees.Add(employee);

entities.SaveChanges();

var message = Request.CreateResponse(HttpStatusCode.Created, employee);

message.Headers.Location = new Uri(Request.RequestUri +

employee.ID.ToString());

return message;

}

}

catch (Exception ex)

{

return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex);

}

}

--------------------------------------------------------------------------------------------------------------------delete api

public void Delete(int id)

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

entities.Employees.Remove(entities.Employees.FirstOrDefault(e => e.ID == id));

entities.SaveChanges();

}

}

When the deletion is successful, since the method return type is void we get status code 204 No Content. We should be returning status code 200 OK.

.......................

Public HttpResponseMessage Delete(int id)

{

try

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

var entity = entities.Employees.FirstOrDefault(e => e.ID == id);

if (entity == null)

{

return Request.CreateErrorResponse(HttpStatusCode.NotFound,

"Employee with Id = " + id.ToString() + " not found to delete");

}

else

{

entities.Employees.Remove(entity);

entities.SaveChanges();

return Request.CreateResponse(HttpStatusCode.OK);

}

}

}

catch (Exception ex)

{

return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex);

}

}

--------------------------------------------------------------------------------------------------------------------put api---update api

It basically update the data-

public void Put(int id, [FromBody]Employee employee)

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

var entity = entities.Employees.FirstOrDefault(e => e.ID == id);

entity.FirstName = employee.FirstName;

entity.LastName = employee.LastName;

entity.Gender = employee.Gender;

entity.Salary = employee.Salary;

entities.SaveChanges();

}

}

{"FirstName":"sam1","Lastname":"Wicht11","Gender":"Male","Salary":54000}

http://localhost:54390/api/employees/8

Gives 204 no content

...............................

public HttpResponseMessage Put(int id, [FromBody]Employee employee)

{

try

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

var entity = entities.Employees.FirstOrDefault(e => e.ID == id);

if (entity == null)

{

return Request.CreateErrorResponse(HttpStatusCode.NotFound,

"Employee with Id " + id.ToString() + " not found to update");

}

else

{

entity.FirstName = employee.FirstName;

entity.LastName = employee.LastName;

entity.Gender = employee.Gender;

entity.Salary = employee.Salary;

entities.SaveChanges();

return Request.CreateResponse(HttpStatusCode.OK, entity);

}

}

}

catch (Exception ex)

{

return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex);

}

}

it returns error of condition is wrong and give 200 code content created

------------------------------------------------------------------------------------

ASP.NET Web API query string parameters

ttp://localhost/api/employees?gender=All All Employees

http://localhost/api/employees?gender=Male Only Male Employees

http://localhost/api/employees?gender=Female Only Female Employees

public HttpResponseMessage Get(string gender = "All")

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

switch (gender.ToLower())

{

case "all":

return Request.CreateResponse(HttpStatusCode.OK,

entities.Employees.ToList());

case "male":

return Request.CreateResponse(HttpStatusCode.OK,

entities.Employees.Where(e => e.Gender.ToLower() == "male").ToList());

case "female":

return Request.CreateResponse(HttpStatusCode.OK,

entities.Employees.Where(e => e.Gender.ToLower() == "female").ToList());

default:

return Request.CreateErrorResponse(HttpStatusCode.BadRequest,

"Value for gender must be Male, Female or All. " + gender + " is invalid.");

}

}

}

-------------------------------------------------------------------------------------------

By both frombody and fromuri we can update the data

http://localhost/api/employees??FirstName=am&Lastname=kWicht&Gender=Male&Salary=54000 ->we can change the data from url by using fromuri Attribute.

public HttpResponseMessage Put([FromBody]int id, [FromUri]Employee employee)

{

try

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

var entity = entities.Employees.FirstOrDefault(e => e.ID == id);

if (entity == null)

{

return Request.CreateErrorResponse(HttpStatusCode.NotFound,

"Employee with Id " + id.ToString() + " not found to update");

}

else

{

entity.FirstName = employee.FirstName;

entity.LastName = employee.LastName;

entity.Gender = employee.Gender;

entity.Salary = employee.Salary;

entities.SaveChanges();

return Request.CreateResponse(HttpStatusCode.OK, entity);

}

}

}

catch (Exception ex)

{

return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex);

}

}

-------------------------------------------------------------------------------------

Call webapi from jquery-

Add Employees.html file in your project-

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta charset="utf-8" />

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

<script type="text/javascript">

$(document).ready(function () {

var ulEmployees = $('#ulEmployees');

$('#btn').click(function () {

$.ajax({

type: 'GET',

url: "api/employees/",

dataType: 'json',

success: function (data) {

ulEmployees.empty();

$.each(data, function (index, val) {

var fullName = val.FirstName + ' ' + val.LastName;

ulEmployees.append('<li>' + fullName + '</li>');

});

}

});

});

$('#btnClear').click(function () {

ulEmployees.empty();

});

});

</script>

</head>

<body>

<div>

<input id="btn" type="button" value="Get All Employees" />

<input id="btnClear" type="button" value="Clear" />

<ul id="ulEmployees" />

</div>

</body>

</html>

---------------------------------------------------------------------------------------------------1

Calling ASP.NET Web API service in a cross domain using jQuery ajax-

There are 2 ways

1-Using JSONP (JSON with Padding)

Enabling CORS (Cross Origin Resource Sharing)

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta charset="utf-8" />

<script src="Scripts/jquery-1.10.2.js"></script>

<script type="text/javascript">

$(document).ready(function () {

var ulEmployees = $('#ulEmployees');

$('#btn').click(function () {

$.ajax({

type: 'GET',

// Make sure to change the port number to

// where you have the employee service

// running on your local machine

url: 'http://localhost:23258/api/Employees',

dataType: 'jsonp', //changed

success: function (data) {

ulEmployees.empty();

$.each(data, function (index, val) {

var fullName = val.FirstName + ' ' + val.LastName;

ulEmployees.append('<li>' + fullName + '</li>')

});

}

});

});

$('#btnClear').click(function () {

ulEmployees.empty();

});

});

</script>

</head>

<body>

<input id="btn" type="button" value="Get All Employees" />

<input id="btnClear" type="button" value="Clear" />

<ul id="ulEmployees"></ul>

</body>

</html>

...................

s1-

Nuget pacakge

Install-Package WebApiContrib.Formatting.Jsonp

s2-

var jsonpFormatter = new JsonpMediaTypeFormatter(config.Formatters.JsonFormatter);

config.Formatters.Insert(0, jsonpFormatter);

.....................

If you want JSONP formatted data back, set Accept header to application/javascript and specify a name for the callback function in the URI. We have set it to ABC.

http://localhost:23258/api/Employees?callback=ABC

---------------------------------------------------------------------------------------2

Enabling cors-

s1-Nuget package

Install-Package Microsoft.AspNet.WebApi.Cors

s2-Include this code in webconfig.cs

EnableCorsAttribute cors = new EnableCorsAttribute("\*", "\*", "\*");

config.EnableCors();.

-----------------------------------------------------------------------------------------------

16 lecture kudvekant-

press F4 key for properties of main project

enable ssl property to true

-----------------------------------------------------------------------------------------

Web API enable HTTPS

RequireHttpsAttribute-->Add class to your project

using System;

using System.Net;

using System.Net.Http;

using System.Text;

using System.Web.Http.Controllers;

using System.Web.Http.Filters;

namespace EmployeeService

{

public class RequireHttpsAttribute : AuthorizationFilterAttribute

{

public override void OnAuthorization(HttpActionContext actionContext)

{

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");

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

uriBuilder.Scheme = Uri.UriSchemeHttps;

uriBuilder.Port = 44337;

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

}

else

{

base.OnAuthorization(actionContext);

}

}

}

}

.............

Add this in webconfig-

config.Filters.Add(new RequireHttpsAttribute());

-------------------------------------------------------------------------------

Implementing basic authentication in ASP.NET Web API

.......

craete a table of users

Create Table Users

(

Id int identity primary key,

Username nvarchar(100),

Password nvarchar(100)

)

Insert into Users values ('male','male')

Insert into Users values ('female','female')

.............

Create a class that checks if the username and password are valid

EmployeeSecurity.cs

using EmployeeDataAccess;

using System;

using System.Linq;

namespace EmployeeService

{

public class EmployeeSecurity

{

public static bool Login(string username, string password)

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

return entities.Users.Any(user =>

user.Username.Equals(username, StringComparison.OrdinalIgnoreCase)

&& user.Password == password);

}

}

}

}

................

Create basic authentication filter

BasicAuthenticationAttribute.cs

using System;

using System.Net;

using System.Net.Http;

using System.Security.Principal;

using System.Text;

using System.Threading;

using System.Web.Http.Controllers;

using System.Web.Http.Filters;

namespace EmployeeService

{

public class BasicAuthenticationAttribute : AuthorizationFilterAttribute

{

public override void OnAuthorization(HttpActionContext actionContext)

{

if (actionContext.Request.Headers.Authorization == null)

{

actionContext.Response = actionContext.Request

.CreateResponse(HttpStatusCode.Unauthorized);

}

else

{

string authenticationToken = actionContext.Request.Headers

.Authorization.Parameter;

string decodedAuthenticationToken = Encoding.UTF8.GetString(

Convert.FromBase64String(authenticationToken));

string[] usernamePasswordArray = decodedAuthenticationToken.Split(':');

string username = usernamePasswordArray[0];

string password = usernamePasswordArray[1];

if (EmployeeSecurity.Login(username, password))

{

Thread.CurrentPrincipal = new GenericPrincipal(

new GenericIdentity(username), null);

}

else

{

actionContext.Response = actionContext.Request

.CreateResponse(HttpStatusCode.Unauthorized);

}

}

}

}

}

...............

Add code on controller

EmployeesController

[BasicAuthentication]

public HttpResponseMessage Get(string gender = "All")

{

string username = Thread.CurrentPrincipal.Identity.Name;

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

switch (username.ToLower())

{

case "male":

return Request.CreateResponse(HttpStatusCode.OK,

entities.Employees.Where(e => e.Gender.ToLower() == "male").ToList());

case "female":

return Request.CreateResponse(HttpStatusCode.OK,

entities.Employees.Where(e => e.Gender.ToLower() == "female").ToList());

default:

return Request.CreateResponse(HttpStatusCode.BadRequest);

}

}

}

to check this when you run the get api

http://localhost:54390/api/employees you need to choose the type-basic auth then pass the username and password then you get filtered data according to it.

----------------------------------------------------------------------------

ASP.NET Web API token authentication-

This only access by authorised user if you use Authorize attribute

it gives an error 401 unauthoriser

http://localhost:54390/api/employees

[Authorize]

public IEnumerable<Employee> Get()

{

using (EmployeeDBEntities entities = new EmployeeDBEntities())

{

return entities.Employees.ToList();

}

}

If you call this api then you do not get any data.

------------------------------------------------------------------It is post api

create a new WebApi Project-

Add a file Register.html

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta charset="utf-8" />

<link href="Content/bootstrap.min.css" rel="stylesheet" />

</head>

<body style="padding-top:20px">

<div class="col-md-10 col-md-offset-1">

<div class="well">

<!--This table contains the fields that we want to capture to register a new user-->

<table class="table table-bordered">

<thead>

<tr class="success">

<th colspan="2">

New User Registration

</th>

</tr>

</thead>

<tbody>

<tr>

<td>Email</td>

<td><input type="text" id="txtEmail" placeholder="Email" /> </td>

</tr>

<tr>

<td>Password</td>

<td>

<input type="password" id="txtPassword"

placeholder="Password" />

</td>

</tr>

<tr>

<td>Confirm Password</td>

<td>

<input type="password" id="txtConfirmPassword"

placeholder="Confirm Password" />

</td>

</tr>

<tr class="success">

<td colspan="2">

<input id="btnRegister" class="btn btn-success"

type="button" value="Register" />

</td>

</tr>

</tbody>

</table>

<!--Bootstrap modal dialog that shows up when regsitration is successful-->

<div class="modal fade" tabindex="-1" id="successModal"

data-keyboard="false" data-backdrop="static">

<div class="modal-dialog modal-sm">

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal">

&times;

</button>

<h4 class="modal-title">Success</h4>

</div>

<div class="modal-body">

<form>

<h2 class="modal-title">Registration Successful!</h2>

</form>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-success"

data-dismiss="modal">

Close

</button>

</div>

</div>

</div>

</div>

<!--Bootstrap alert to display any validation errors-->

<div id="divError" class="alert alert-danger collapse">

<a id="linkClose" href="#" class="close">&times;</a>

<div id="divErrorText"></div>

</div>

</div>

</div>

<script src="Scripts/jquery-1.10.2.min.js"></script>

<script src="Scripts/bootstrap.min.js"></script>

<script type="text/javascript">

$(document).ready(function () {

//Close the bootstrap alert

$('#linkClose').click(function () {

$('#divError').hide('fade');

});

// Save the new user details

$('#btnRegister').click(function () {

$.ajax({

url: '/api/account/register',

method: 'POST',

data: {

email: $('#txtEmail').val(),

password: $('#txtPassword').val(),

confirmPassword: $('#txtConfirmPassword').val()

},

success: function () {

$('#successModal').modal('show');

},

error: function (jqXHR) {

$('#divErrorText').text(jqXHR.responseText);

$('#divError').show('fade');

}

});

});

});

</script>

</body>

</html>

http://localhost:62594/api/account/register

Change the type to post request and add the parameters in x-wwww-from-encoded

key value

email-sun@gmail.com

password-$UUny123

confirmPassword-$UUny123

It add the the data in app\_data folder in project

It created a default database in the visual studio its string is

<connectionStrings>

<add name="DefaultConnection" connectionString="Data Source=(LocalDb)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\aspnet-qweer-20170718053936.mdf;Initial Catalog=aspnet-qweer-20170718053936;Integrated Security=True" providerName="System.Data.SqlClient" />

</connectionStrings>

To see this data add click on the showfiles button in project.

........................

-----------------------------------------------------------------------------------------------------------------------

change this url to this to make a different database with dbname userdb

<add name="DefaultConnection"

connectionString="Data Source=(LocalDb)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\UsersDB.mdf;Initial Catalog=UsersDB;Integrated Security=True"

providerName="System.Data.SqlClient" />

----------------------------------------------------------------------------------------

change the connection string to this to make db on sqlserver

<add name="DefaultConnection"

connectionString="Data Source=(local);Initial Catalog=UsersDB;Integrated Security=True"

providerName="System.Data.SqlClient" />

...

<add name="DefaultConnection"

connectionString="Data Source=(local);Initial Catalog=EmployeeDB;Integrated Security=True"

providerName="System.Data.SqlClient" />

----------------------------------------------------------------------------------------------------

How To access Authiorize attribute data-

s1-First is you need to create token

s2-Issue a POST request to /token

http://localhost:62594/token

s3-In the request body include username and the password.

username=test@gmail.com&password=Test123!&grant\_type=password

We also need to set grant\_type=password. This indicates that we are presenting password for acquiring access token.

s4-Then it generated the access token

when you access the api

http://localhost:54390/api/employees then tou get error 401 unauthorised

To access this -pass get request

http://localhost:54390/api/employees write the api

and pass the header

Authorization:Bearer Accesstokendata(dscfdscd)

---------------------------------------------------------------------

signin and signout

1. sessionStorage data is lost when the browser window is closed.

2. To store an item in the browser session storage use setItem() method

Example : sessionStorage.setItem("accessToken", response.access\_token)

3. To retrieve an item from the browser session storage use getItem() method

Example : sessionStorage.getItem("accessToken")

4. To remove an item from the browser session storage use removeItem() method

Example : sessionStorage.removeItem('accessToken')

------------------------------------------------------

url encode decode

https://console.developers.google.com

----------------------------------------------

Attribute routing in ASP.NET Web API 2-

StudentsController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

using demoapi.Models;

namespace demoapi.Controllers

{

public class StudentsController : ApiController

{

static List<Student> students = new List<Student>()

{

new Student() { Id = 1, Name = "Tom" },

new Student() { Id = 2, Name = "Sam" },

new Student() { Id = 3, Name = "John" }

};

public IEnumerable<Student> Get()

{

return students;

}

public Student Get(int id)

{

return students.FirstOrDefault(s => s.Id == id);

}

[Route("api/students/{id}/courses")] //Attribute routing

public IEnumerable<string> GetStudentCourses(int id)

{

if (id == 1)

return new List<string>() { "C#", "ASP.NET", "SQL Server" };

else if (id == 2)

return new List<string>() { "ASP.NET Web API", "C#", "SQL Server" };

else

return new List<string>() { "Bootstrap", "jQuery", "AngularJs" };

}

}

}

........................

student.cs

public int Id { get; set; }

public string Name { get; set; }

/api/students/1

/api/students/1/courses

--------------------------------------------------------------------------