# BÀI 14: THỰC HÀNH 06 - BẢO MẬT ỨNG DỤNG WEB API

***Mục tiêu***

* Trình bày được một số phương pháp bảo mật trên WebAPI
* Trình bày được các bước thực hiện bảo mật WebAPI với token
* Xây dựng được Web API bảo mật với token
* Vận dụng và tổng hợp được để xây dựng được ứng dụng WebAPI theo yêu cầu
* Nghiêm túc, chăm chỉ, nhiệt tình, tích cực trong học tập

Hướng dẫn thực hiện

Bài 1 Xây dựng ứng dụng bảo mật với lớp BasicAuthentication

#### Bước 1: Tạo dự án ASP.NET Web API

Khởi động Visual Studio → tạo một project ASP.NET Web Application và chọn template Web API.

Đặt tên cho ứng dụng, chọn nơi lưu trữ và FrameWork rồi chọn Create

Chọn template WebAPI

#### Bước 2: Tạo một lớp BasicAuthenticationAttribute

Lớp BasicAuthenticationAttribu kế thừa từ AuthorizationFilterAttribute (khai báo System.Web.Http.Filters;) và ghi đè phương thức OnAuthorization từ lớp cơ sở (AuthorizationFilterAttribute).

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Net;  using System.Net.Http;  using System.Security.Principal;  using System.Threading;  using System.Web;  using System.Web.Http.Controllers;  using System.Web.Http.Filters;  namespace DemoToken.Models  {  public class BasicAuthenticationAttribute : AuthorizationFilterAttribute  {  public override void OnAuthorization(HttpActionContext actionContext)  {  base.OnAuthorization(actionContext);  if (actionContext.Request.Headers.Authorization != null)  {  var authToken = actionContext.Request.Headers  .Authorization.Parameter;  // decoding authToken we get decode value in 'Username:Password' format  var decodeauthToken = System.Text.Encoding.UTF8.GetString(  Convert.FromBase64String(authToken));  // spliting decodeauthToken using ':'  var arrUserNameandPassword = decodeauthToken.Split(':');  // at 0th postion of array we get username and at 1st we get password  if (IsAuthorizedUser(arrUserNameandPassword[0], arrUserNameandPassword[1]))  {  // setting current principle  Thread.CurrentPrincipal = new GenericPrincipal(  new GenericIdentity(arrUserNameandPassword[0]), null);  }  else  {  actionContext.Response = actionContext.Request  .CreateResponse(HttpStatusCode.Unauthorized);  }  }  else  {  actionContext.Response = actionContext.Request  .CreateResponse(HttpStatusCode.Unauthorized);  }  }  public static bool IsAuthorizedUser(string Username, string pass)  {  // In this method we can handle our database logic here...  return Username == "fit" && pass == "123";  }  }  } |

Và cấu hình trong App\_Start/WebApiConfig.cs

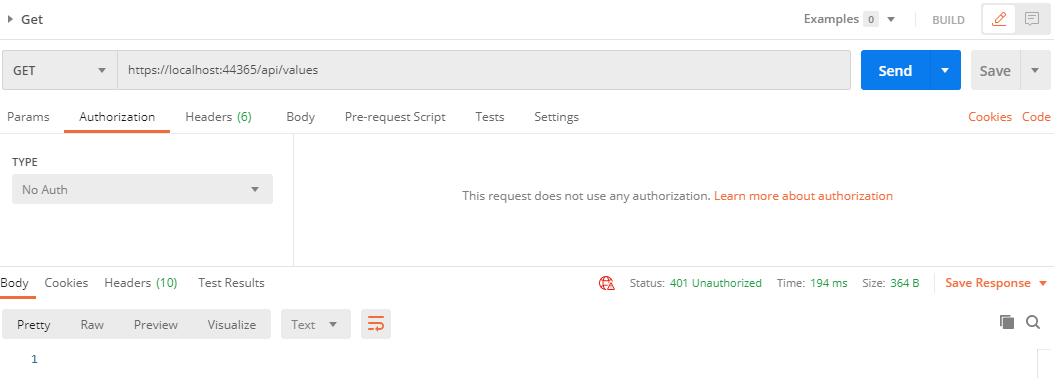
|  |
| --- |
| App\_Start/WebApiConfig.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Net.Http;  using System.Web.Http;  using BasicAuthentication.Models;  namespace BasicAuthentication  {  public static class WebApiConfig  {  public static void Register(HttpConfiguration config)  {  config.Filters.Add(new BasicAuthenticationAttribute());  // Web API routes  config.MapHttpAttributeRoutes();  config.Routes.MapHttpRoute(  name: "DefaultApi",  routeTemplate: "api/{controller}/{id}",  defaults: new { id = RouteParameter.Optional }  );    }  }  } |

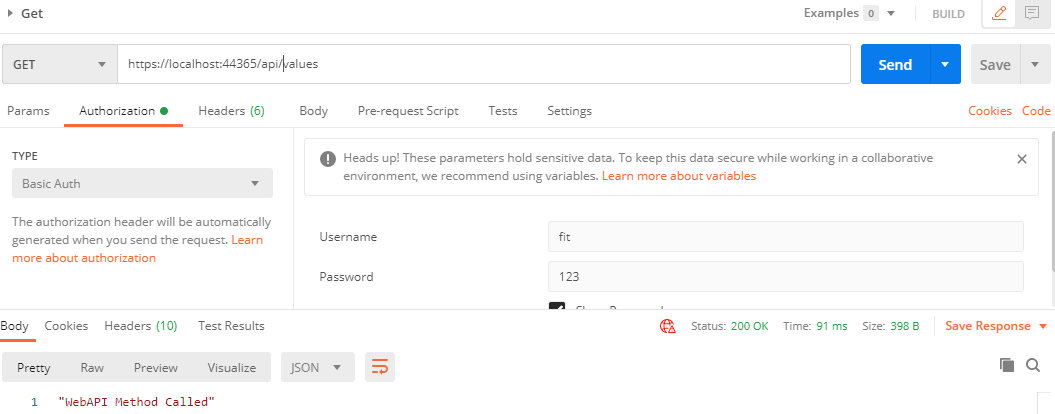
#### Bước 3: Viết Web API Controller

|  |
| --- |
| ValuesController.cs |
| using BasicAuthentication.Models;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Net;  using System.Net.Http;  using System.Web.Http;  namespace BasicAuthentication.Controllers  {  // [Authorize]  public class ValuesController : ApiController  {  [BasicAuthentication]  public string Get()  {  return "WebAPI Method Called";  }  }  } |

#### Bước 4: Chạy và kiểm tra với công cụ

Phương thức HttpGet Get với Route mặc định global trong WebApiConfig.cs là ("api/values")] và chỉ cho phép người dùng truy cập thỏa mãn [BasicAuthentication] đã định nghĩa



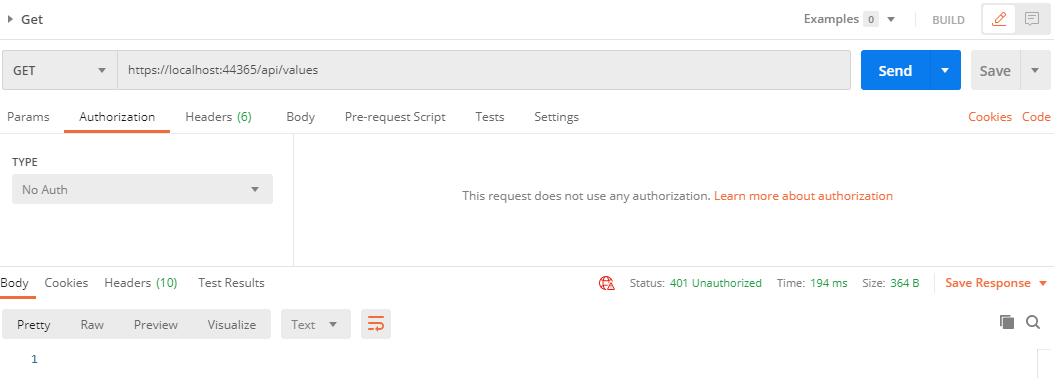


Bài 2 Ứng dụng trong bài 1, kiểm tra ứng dụng bảo mật với thuộc tính [Authorize] và [AllowAnonymous]

|  |
| --- |
| ValuesController.cs |
| using BasicAuthentication.Models;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Net;  using System.Net.Http;  using System.Web.Http;  namespace BasicAuthentication.Controllers  {  [Authorize]  public class ValuesController : ApiController  {  [AllowAnonymous]  [Route("values/AllowAnonymous")]  public string Get1()  {  return "WebAPI Method Called with AllowAnonymous (-:";  }  }  } |

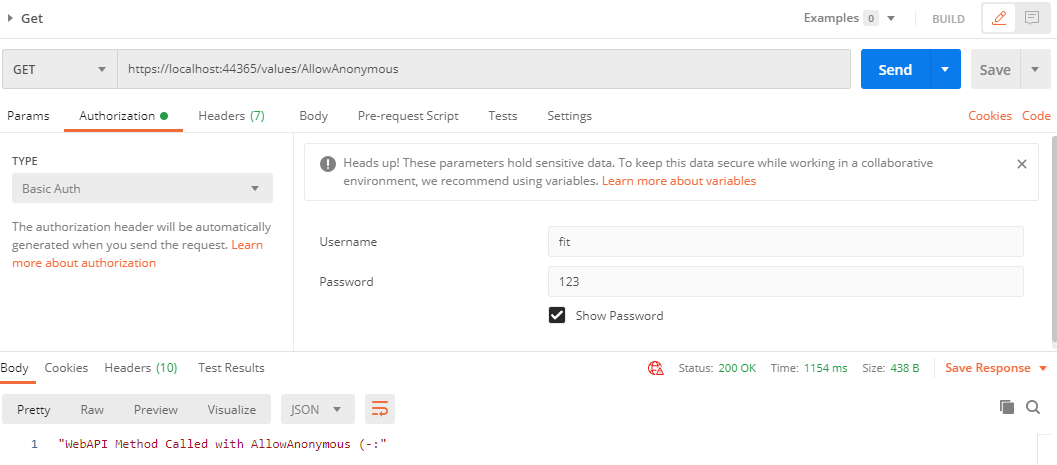
Run và test

Phương thức HttpGet Get với Route mặc định global trong WebApiConfig.cs là ("api/values")] và chỉ cho phép người dùng truy cập thỏa mãn [BasicAuthentication] đã định nghĩa

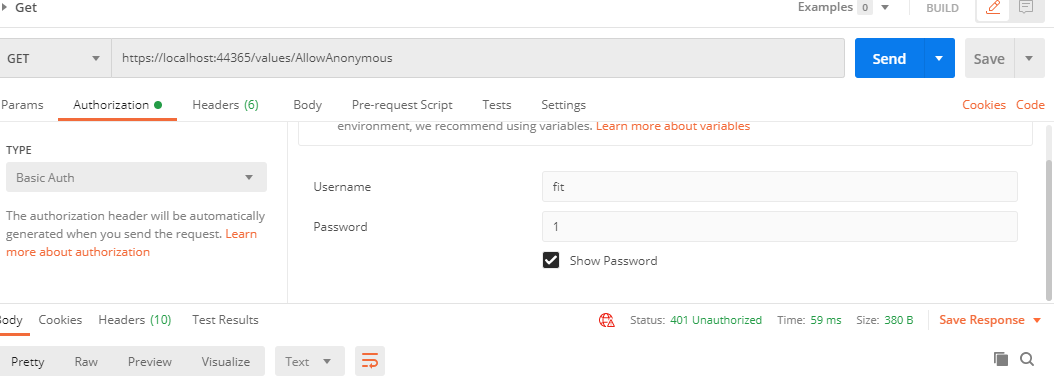


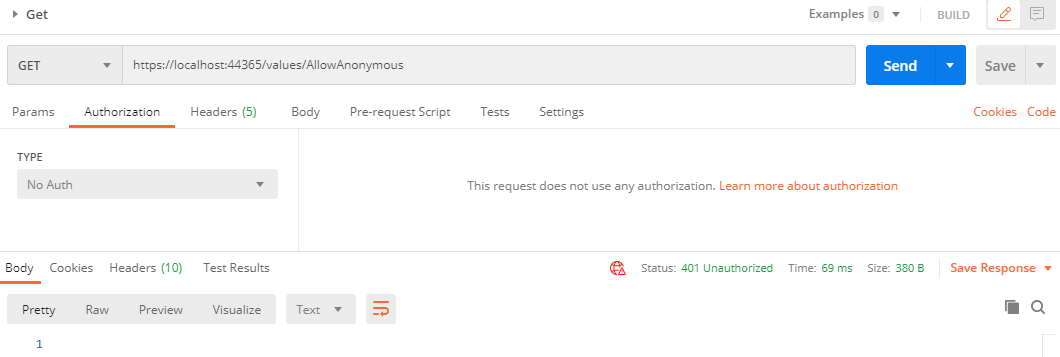
Phương thức HttpGet Get1 với [Route("values/AllowAnonymous")] và cho phép tất cả người dùng truy cập với thuộc tính [AllowAnonymous]

Đúng username và password



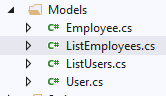
Code 401 nếu cấu hình Basic authorication mật khẩu hoặc tài khoản sai hoặc no authorication





## Bài 3 Xây dựng ứng dụng webAPI Token Based Authentication

### B1 Models



|  |
| --- |
| Model/**User.cs** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Web;  namespace DemoAuthor1\_attribute.Models  {  public class User  {  public int ID { get; set; }  public string UserName { get; set; }  public string Password { get; set; }  }  } |

|  |
| --- |
| Model/ **Employee.cs** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Web;  namespace DemoAuthor1\_attribute.Models  {  public class Employee  {  public int ID { get; set; }  public string Name { get; set; }  public string Gender { get; set; }  public string Dept { get; set; }  public int Salary { get; set; }  }  } |

|  |
| --- |
| ListUsers.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Web;  namespace DemoAuthor1\_attribute.Models  {  public class ListUsers  {  public List<User> GetUsers()  {  // In Real-time you need to get the data from any persistent storage  // For Simplicity of this demo and to keep focus on Basic Authentication  // Here we are hardcoded the data  List<User> userList = new List<User>();  userList.Add(new User()  {  ID = 100,  UserName = "MaleUser",  Password = "123456"  });  userList.Add(new User()  {  ID = 02027,  UserName = "diepnh",  Password = "123"  });  userList.Add(new User()  {  ID = 101,  UserName = "FemaleUser",  Password = "abcdef"  });  return userList;  }  }  } |

|  |
| --- |
| ListEmployees.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Web;  namespace DemoAuthor1\_attribute.Models  {  public class ListEmployees  {  public List<Employee> GetEmployees()  {  // In Real-time you need to get the data from any persistent storage  // For Simplicity of this demo and to keep focus on Basic Authentication  // Here we hardcoded the data  List<Employee> empList = new List<Employee>();  for (int i = 0; i < 10; i++)  {  if (i > 5)  {  empList.Add(new Employee()  {  ID = i,  Name = "Name" + i,  Dept = "IT",  Salary = 1000 + i,  Gender = "Male"  });  }  else  {  empList.Add(new Employee()  {  ID = i,  Name = "Name" + i,  Dept = "HR",  Salary = 1000 + i,  Gender = "Female"  });  }  }  return empList;  }  }  } |

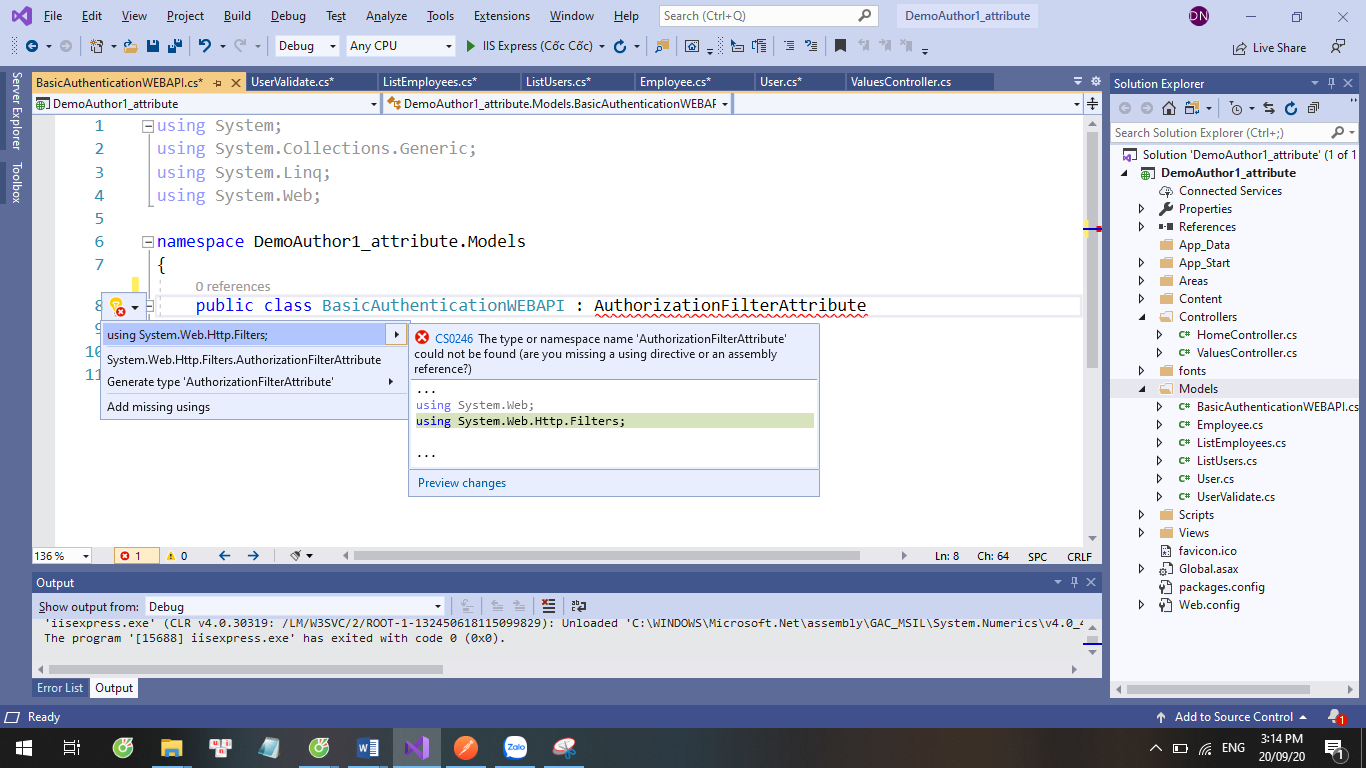
Tạo một lớp **UserValidate** để kiểm tra tên người dùng và mật khẩu có hợp lệ hay không. Nhấp chuột phải vào thư mục Mô hình và thêm tệp lớp có Tên UserValidate như sau:

|  |
| --- |
| **UserValidate.cs** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Web;  namespace DemoAuthor1\_attribute.Models  {  public class UserValidate  {  public static bool Login(string username, string password)  {  ListUsers userBL = new ListUsers();  var UserLists = userBL.GetUsers();  return UserLists.Any(user =>  user.UserName.Equals(username, StringComparison.OrdinalIgnoreCase)  && user.Password == password);  }  }  } |

### B2: Create a Basic Authentication Filter in Web API

Trong Models, add a class **BasicAuthenticationAttribute** kế thừa từ AuthorizationFilterAttribute trong gói System.Web.Http.Filters;

Sau đo tạo override phương thức OnAuthorization như sau



|  |
| --- |
| **BasicAuthenticationAttribute.cs** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Net;  using System.Net.Http;  using System.Security.Principal;  using System.Text;  using System.Threading;  using System.Web;  using System.Web.Http.Controllers;  using System.Web.Http.Filters;  namespace DemoAuthor1\_attribute.Models  {  public class BasicAuthenticationWEBAPI : AuthorizationFilterAttribute  {  private const string Realm = "My Realm";  public override void OnAuthorization(HttpActionContext actionContext)  {  //If the Authorization header is empty or null  //then return Unauthorized  if (actionContext.Request.Headers.Authorization == null)  {  actionContext.Response = actionContext.Request  .CreateResponse(HttpStatusCode.Unauthorized);  // If the request was unauthorized, add the WWW-Authenticate header  // to the response which indicates that it require basic authentication  if (actionContext.Response.StatusCode == HttpStatusCode.Unauthorized)  {  actionContext.Response.Headers.Add("WWW-Authenticate",  string.Format("Basic realm=\"{0}\"", Realm));  }  }  else  {  //Get the authentication token from the request header  string authenticationToken = actionContext.Request.Headers  .Authorization.Parameter;  //Decode the string  string decodedAuthenticationToken = Encoding.UTF8.GetString(  Convert.FromBase64String(authenticationToken));  //Convert the string into an string array  string[] usernamePasswordArray = decodedAuthenticationToken.Split(':');  //First element of the array is the username  string username = usernamePasswordArray[0];  //Second element of the array is the password  string password = usernamePasswordArray[1];  //call the login method to check the username and password  if (UserValidate.Login(username, password))  {  var identity = new GenericIdentity(username);  IPrincipal principal = new GenericPrincipal(identity, null);  Thread.CurrentPrincipal = principal;  if (HttpContext.Current != null)  {  HttpContext.Current.User = principal;  }  }  else  {  actionContext.Response = actionContext.Request  .CreateResponse(HttpStatusCode.Unauthorized);  }  }  }  }  } |

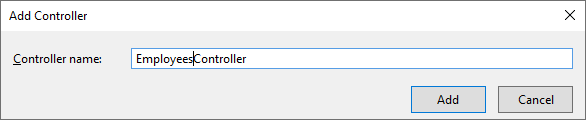
|  |
| --- |
| WebApiConfig |
| using DemoAuthor1\_attribute.Models;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Web.Http;  namespace DemoAuthor1\_attribute  {  public static class WebApiConfig  {  public static void Register(HttpConfiguration config)  {  // Web API configuration and services  // Web API routes  config.MapHttpAttributeRoutes();  config.Filters.Add(new BasicAuthenticationWEBAPI());  config.Routes.MapHttpRoute(  name: "DefaultApi",  routeTemplate: "api/{controller}/{id}",  defaults: new { id = RouteParameter.Optional }  );  }  }  } |

### B3: Controller

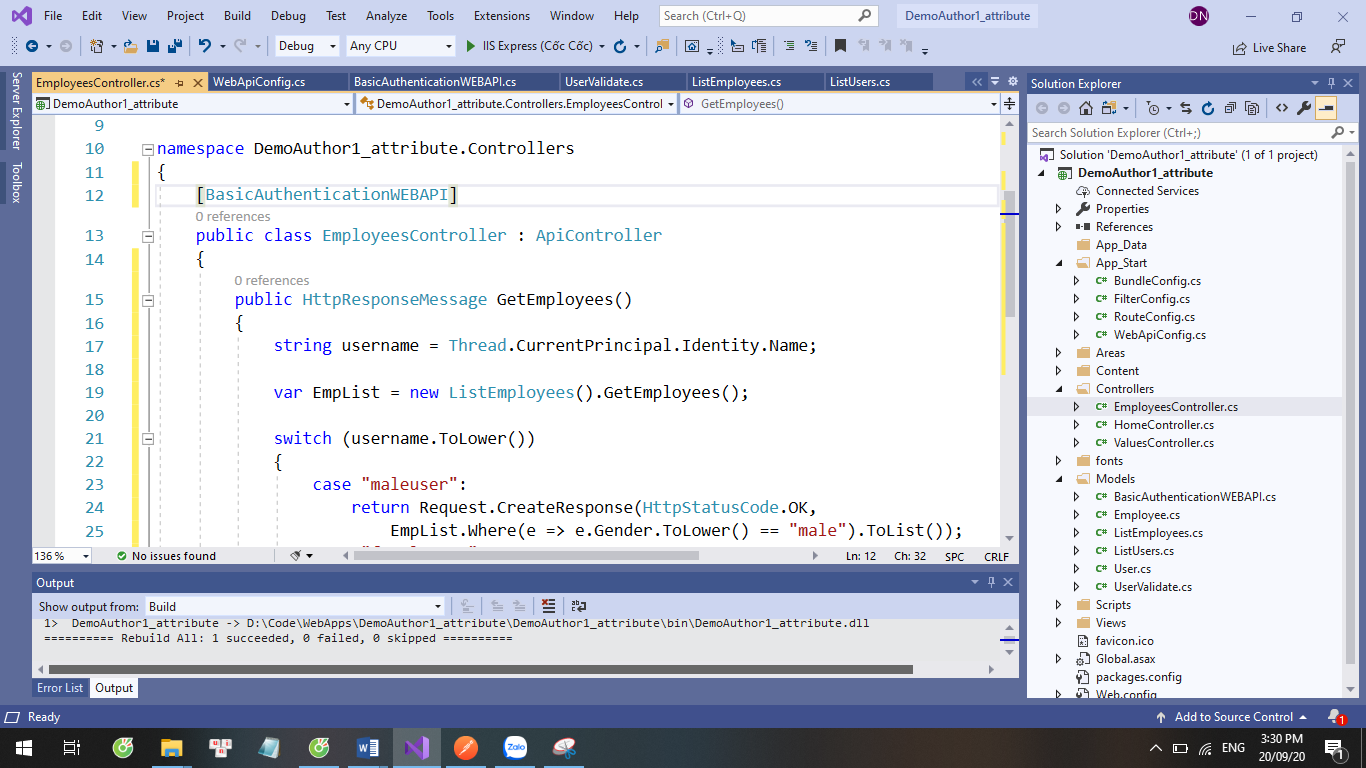
Ta có thể áp dụng thuộc tính BasicAuthenticationAttribute trên một controller cụ thể, điều này sẽ cho phép xác thực cơ bản cho tất cả các phương thức có trong controller đó.

Ta có thể kích hoạt basic authentication bằng cách khai báo tại action method (chỉ áp dụng cho particular action method which is decorated with the BasicAuthenticationAttribute).

Yêu cầu: Trong ListUsers, ta có hai người dùng là MaleUser và FemaleUser và nếu người dùng đăng nhập bằng tên người dùng MaleUser, ta hiển thị tất cả nhân viên "nam" và nếu người dùng đăng nhập bằng tên người dùng FemaleUser, ta hiển thị tất cả nhân viên nữ; người dùng là diepnh thì hiển thị tất cả nhân viên; còn lại ko đúng là 1 trong 3 tài khoản trên thì trả lại HttpStatusCode.BadRequest.

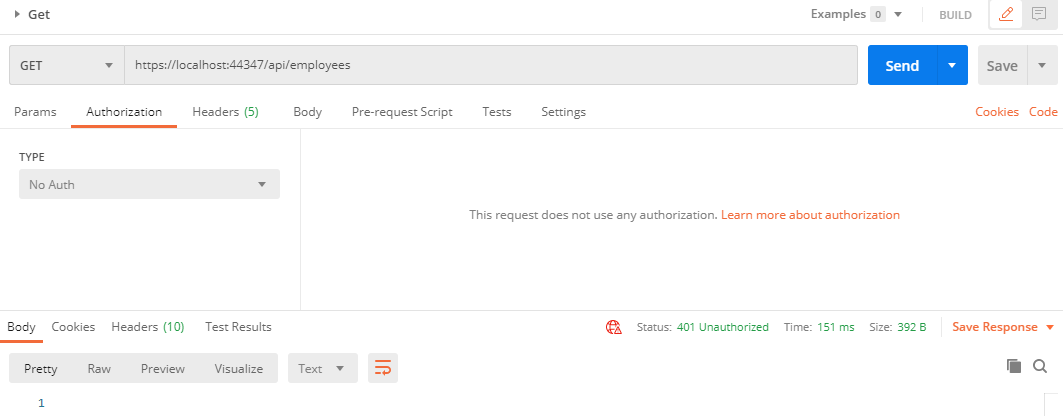


|  |
| --- |
| **EmployeesController.cs** |
| using DemoAuthor1\_attribute.Models;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Net;  using System.Net.Http;  using System.Threading;  using System.Web.Http;  namespace DemoAuthor1\_attribute.Controllers  {  [BasicAuthenticationWEBAPI]  public class EmployeesController : ApiController  {  public HttpResponseMessage GetEmployees()  {  string username = Thread.CurrentPrincipal.Identity.Name;  var EmpList = new ListEmployees().GetEmployees();  switch (username.ToLower())  {  case "maleuser":  return Request.CreateResponse(HttpStatusCode.OK,  EmpList.Where(e => e.Gender.ToLower() == "male").ToList());  case "femaleuser":  return Request.CreateResponse(HttpStatusCode.OK,  EmpList.Where(e => e.Gender.ToLower() == "female").ToList());  case "diepnh":  return Request.CreateResponse(HttpStatusCode.OK,  EmpList.ToList());  default:  return Request.CreateResponse(HttpStatusCode.BadRequest);  }  }  }  } |



### Test

TH1: authorization header



Lý do

Xác thực

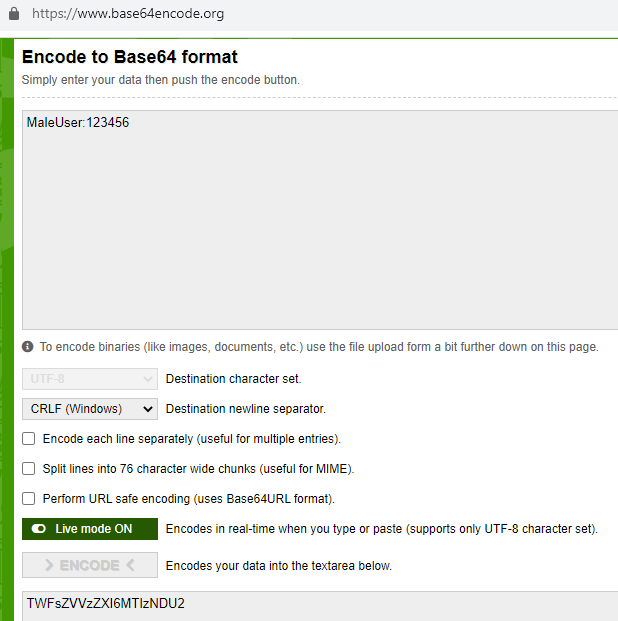
Header thuộc tính Authorization với value là “BASIC “và usename và password sau khi mã hóa

Để lấy chuỗi mã hóa hoặc chuỗi giải mã vào <https://www.base64encode.org/>

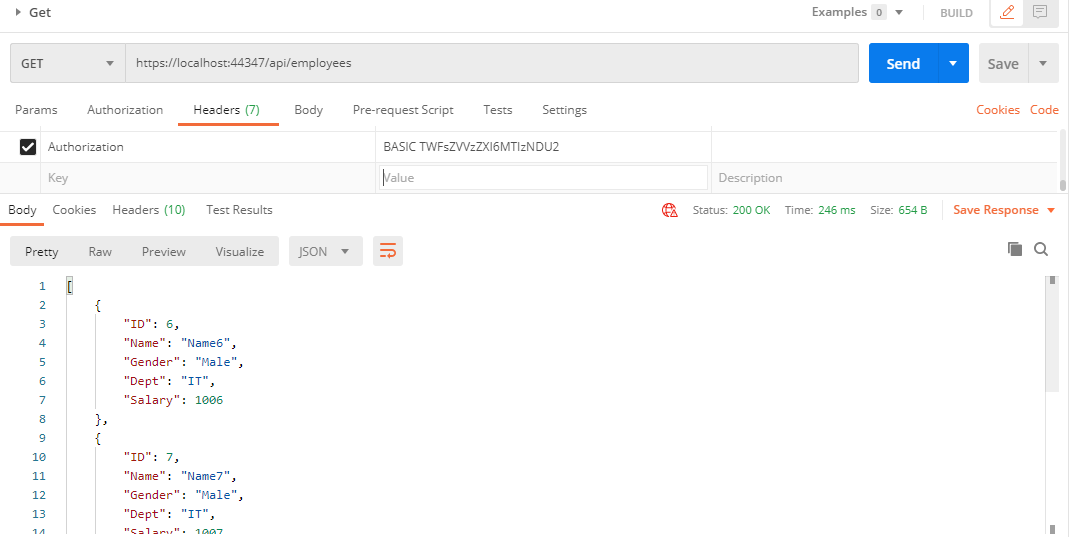
Nhập usename và password cách nhau bởi dấu :

|  |  |  |
| --- | --- | --- |
| usename:password | encode | Authorization value tương ứng Encode to Base64 format |
| diepnh:123 | ZGllcG5oOjEyMw== | BASIC ZGllcG5oOjEyMw== |
| MaleUser:123456 | TWFsZVVzZXI6MTIzNDU2 | BASIC TWFsZVVzZXI6MTIzNDU2 |
| FemaleUser:abcdef | RmVtYWxlVXNlcjphYmNkZWY= | BASIC RmVtYWxlVXNlcjphYmNkZWY= |
| FemaleUser:ab | RmVtYWxlVXNlcjphYg== |  |
| Tk154:101174 | VGsxNTQ6MTAxMTc0 |  |
| 101175:tk155 | MTAxMTc1OnRrMTU1 |  |
| tk157:abc123 | dGsxNTc6YWJjMTIz |  |
| tk154:123 | dGsxNTQ6MTIz |  |
| Tk155:abc | VGsxNTU6YWJj |  |

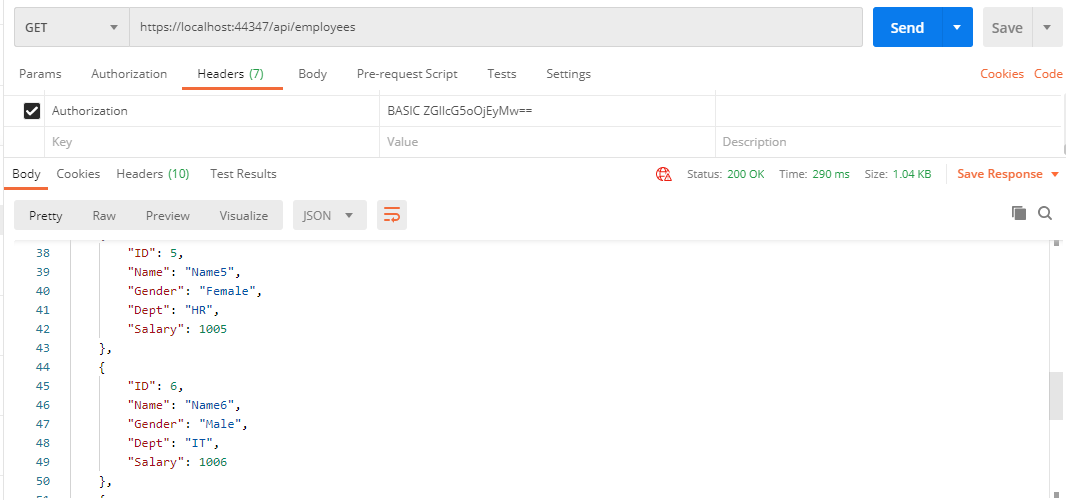
Copy encode string tương ứng usename:password cho thuộc tính Authorization trong request header

Tài khoản MaleUser:123456, trả về all male employee

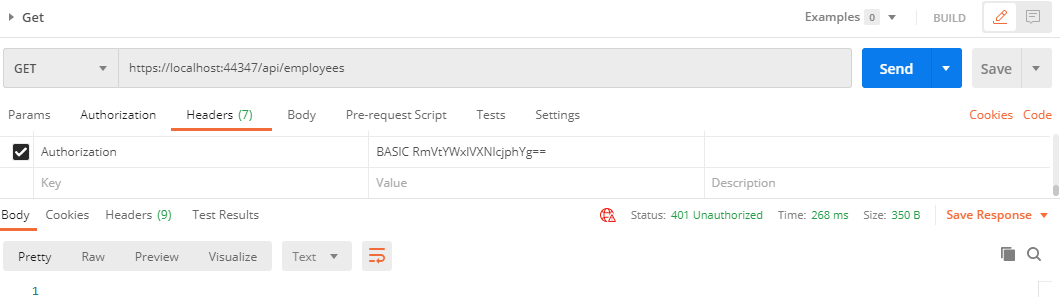


Tài khoản diepnh:123, trả về all employee



Không phải 1 trong 3 tài khoản ví dụ FemaleUser:ab trả về code 401 không được phép

Không phải đúng encode của 1 xâu thì sẽ bị lỗi trả về code (nếu ko try catch)



Bài tập tự làm

**Bài tập** Xây dựng WebAPI với token authorize với

Tất cả người dùng được phép lấy thông tin theo mã, không có quyền lấy thông tin toàn bộ đối tượng

Người dùng quyền admin có quyền lấy thông tin toàn bộ đối tượng, thêm, sửa xóa

Người dùng quyền user có quyền lấy thông tin toàn bộ đối tượng và thêm; nhưng không có quyền sửa, xóa.

Thực hành trên các ứng dụng WebAPI với sau

1 Cho các CSDL QLSinhVien gồm các quan hệ

**SinhVien** (MASV, HOTEN)

**LopHoc**(KYHIEU, TENMONHOC, THOIGIAN)

**GiaoVien** (MAGV, TENGV)

2 Cho các CSDL QLSinhVien gồm các quan hệ

**BangDiem** (MaSV, Ma\_mon, Diem)

**SinhVien** (MaSV, Ten\_SV, Gioi\_tinh, Dia\_chi, Ngay\_sinh)

**MonHoc** (MaMH, Ten\_mon, Chuyen\_nganh, So\_hoc\_trinh)

**GiaoVien** (MaGV, Ten\_GV, Chuyen\_nganh, Dia\_chi, Dien\_thoai)

**GV\_DAY** (MaGV, MaMH)

3 Cho các CSDL QLNhanVien gồm các quan hệ

**Nhanvien** (MANV, HOTEN, NGAYSINH, PHAI, DIACHI, MAPB)

**Phongban** (MAPB, TENPB)

**Cong** (MACT, MANV, SLNGAYCONG)

**Congtrinh** (MACT, TENCT, DIADIEM, NGAYCAPGP, NGAYKC)



4 Cho các CSDL QLThuVien gồm các quan hệ

**Docgia** (MADG, HOTEN, NGAYSINH, DIACHI, NGHENGHIEP)

**Phieumuon** (SOPM, NGAYMUON, MADG)

**Chitietmuon** (SOPM, MADAUSACH, NGAYTRA)

**Dausach** (MADAUSACH, BAN, TAP, MASH)

**Sach** (MASH, TENSACH, TACGIA, NHAXB, NAMXB)

5 Cho các CSDL QLdự án gồm các quan hệ

