WEB API Session 04

By CODEMIND Technology

Contact us +91 96650 44698

Enable HTTPS

- Authentication - process of identifying the user
- Authorization - Process of deciding whether allowed to perform an action on specific resource

Use ‘**AuthorizationFilterAttribute**’

Enable HTTPS

```
public class RequiredHttpsAttribute : AuthorizationFilterAttribute {

        public override void OnAuthorization(HttpActionContext actionContext) {

                if(actionContext.Request.RequestUri.Scheme != Uri.UriSchemeHttps) {
                    throw new Exception(“Not Authorised");
                } else {
                        base.onAuthorization(actionContext);
                }
        }
}
```

Enable HTTPS

- Add above filter in ReqisterHttpConfiguration to enable for all controllers.

```
config.Filters.Add(new RequiredHttpsAttribute);
```

Enable HTTPS

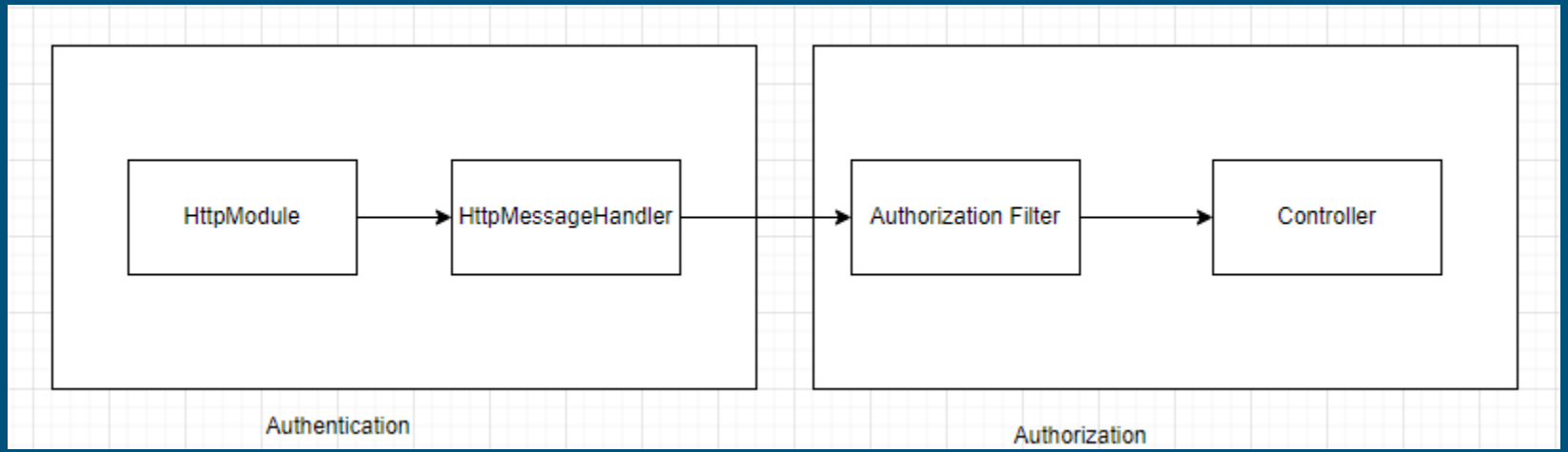
- If we need to add for Specific controller then:

```
[RequireHttps]
Public class EmployeeController : ApiController {
}
```

Authorization in Web API

- The authorization process is going to happen before executing the controller Action method which provide you the flexibility to decide whether you want to grant access to that resource or not.

Authorization in Web API



Authorization in Web API

- Use [Authorize] Attribute -> 401 for unauthorize
- At Globally:

```
Public static void Register(HttpConfiguration config) {
        config.Filters.Add(new AuthorizeAttribute());
}
```

Authorization in Web API

- At Controller level:

```
[Authorize]
Public class EmployeeController : ApiController {
}
```

Authentication in Web API

1. Http Basic Authentication
2. API Keys
3. OAuth

1. Http Basic Authentication

- HTTP agent simply provides a username & password to prove their authentication
- Not require cookies, session ID, login pages & other such specialty solution because it uses the HTTP header itself.

1. Http Basic Authentication

- The problem with this, unless process strictly enforced throughout the entire data cycle to SSL for security; the authentication is transmitted in open insecure lines - it lends itself to “man in the middle attack”

2. API Keys

- In this approach, a unique generated value is assigned to each first time user, signifying that the user is know.
- When the user re-enter the system, this unique key [Sometimes generated from H/W combination & i/p data ^ other random generated by server] is used to prove that they’re the same user as before.
- This is very fast

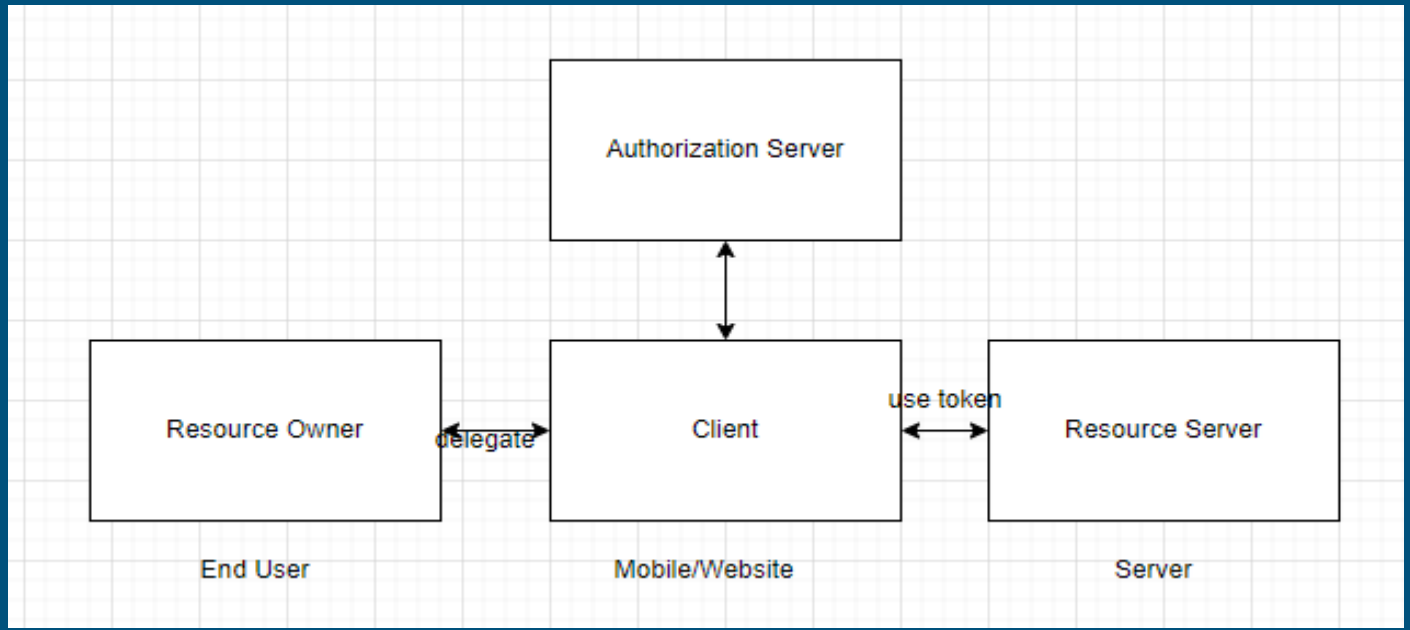
2. API Keys

- The ability to prove identity once & move on is very agile.
- The Problem is that API keys often used for what they’re not an API key is not method of authorization, it’s an **authentication**.

3. OAuth: [Both Authentication & authorization]

- In this approach, the user login into system, that system will then request authentication, usually in the form of a token.
- User will then forward this request to an authentication server, which will either reject/allows this authentication. From here, the token is provided to user & then to the requester.
- Such token can then be checked at any time independently of the user by requestor for validation & can be used overtime with strictly limited scope & age of validity.

3. OAuth: [Both Authentication & authorization]



3. OAuth: [Both Authentication & authorization]

1. Resource Owner [RO] - End user
2. Client - mobile app, web site etc that wants to access data on behalf of RO
3. Authorization Server - The security token Services (STS) or the OAuth server that issues token.
4. Resource Server [RS] - Service that expose the data i.e. the API

3. OAuth: [Both Authentication & authorization]

In OAuth, there two kinds of tokens :

1. Access token - that are presented to API.
2. Refresh token - Used by client to get a new access token from the authorization server.

Types of token

1. WS-Security token, especially SAML token
2. JWT token
3. Legacy token
4. Custom tokens

JWT Token [JSON Web Token]

- JSON web token(JWT) is an opened standard (RFS 7519) that defines a compact & self-contained way for security transmitting information between parties as a JSON object.
- This information can be verified & trusted because it is digitally signed.
- JWT can be signed using a secret(with HMAC algorithm) or public/private key pair using RSA or ECDSA

JWT Token [JSON Web Token]

1. Header:

   Signed algorithm being used, such as HMAC, SHA256 or RSA

   Eg.

   {

   “alg” : “HS256”,

   “typ” : “JWT”

   }

JWT Token [JSON Web Token]

2. Payload: - which contain claims. Claims are the statements about an entity(typically the user & additional data)

Eg.

```
{
        “sub” : “123456”,
        “name” : “John”,
        “admi” : true
}
```

JWT Token [JSON Web Token]

2. Signature: - To create the signature part you have to take the encoded header, the encoded payload, a secret, the algorithm specified in header & sign that

Eg. if you want to use HMAC SHA256 algorithm, the signature will be created in following way:

HMACSHA256(

base64urlEncode(header) + “.” + base64urlEncode(payload) + “.” + secret

)

CODEMIND
Your Success Partner

Thank you

Contact us - 96650 44698

CODEMIND
Your Success Partner