**Secure a Web API with Login**

**Introduction:**

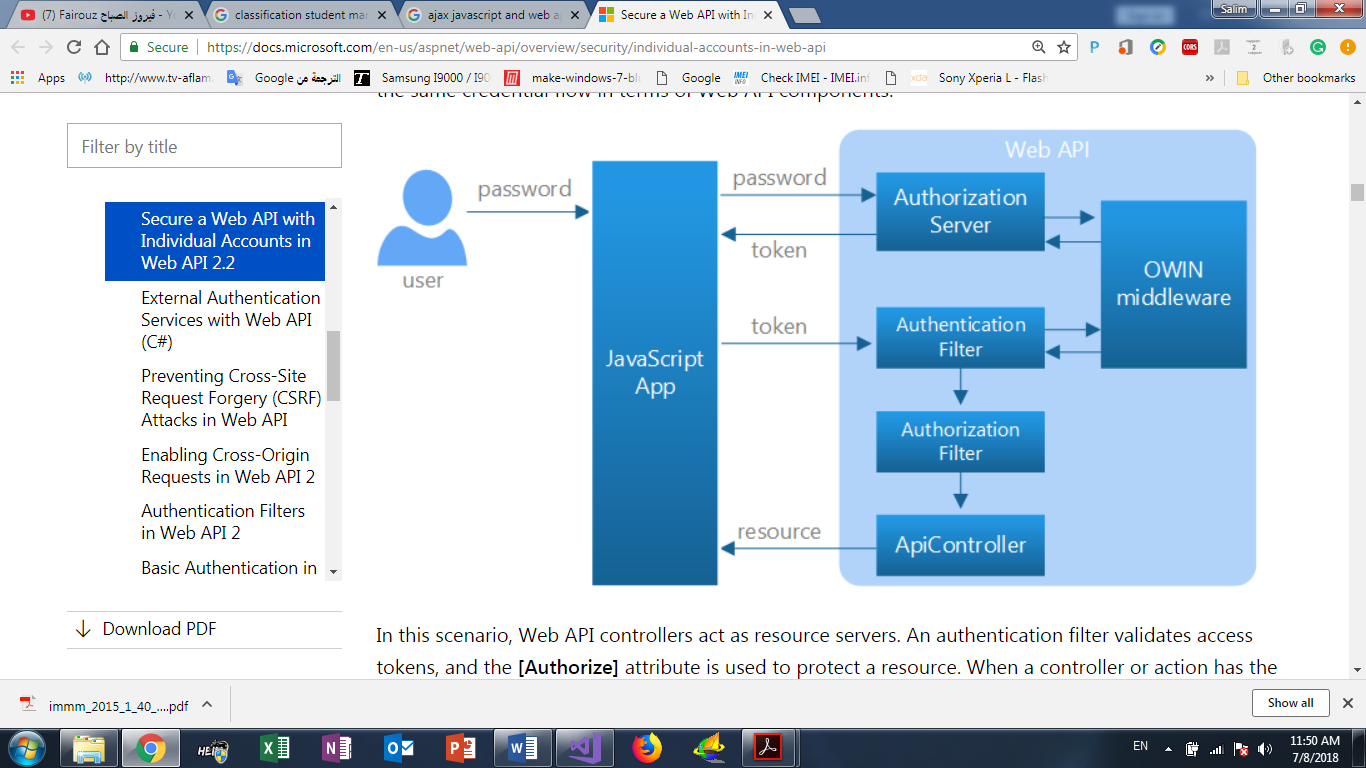
Web API uses OAuth2 to authenticate requests, so we need to Define some OAuth2 terminology:

* *Resource*. Some piece of data that can be protected.
* *Resource server*. The server that hosts the resource.
* *Resource owner*. The entity that can grant permission to access a resource. (Typically, the user.)
* *Client*: The app that wants access to the resource. In this article, the client is a web browser.
* *Access token*. A token that grants access to a resource.
* *Bearer token*. A particular type of access token, with the property that anyone can use the token. In other words, a client doesn't need a cryptographic key or other secret to use a bearer token. For that reason, bearer tokens should only be used over a HTTPS, and should have relatively short expiration times.
* *Authorization server*. A server that gives out access tokens.

## **Local Login Credential Flow:**

For local login, Web API uses the resource owner password flow defined in OAuth2.

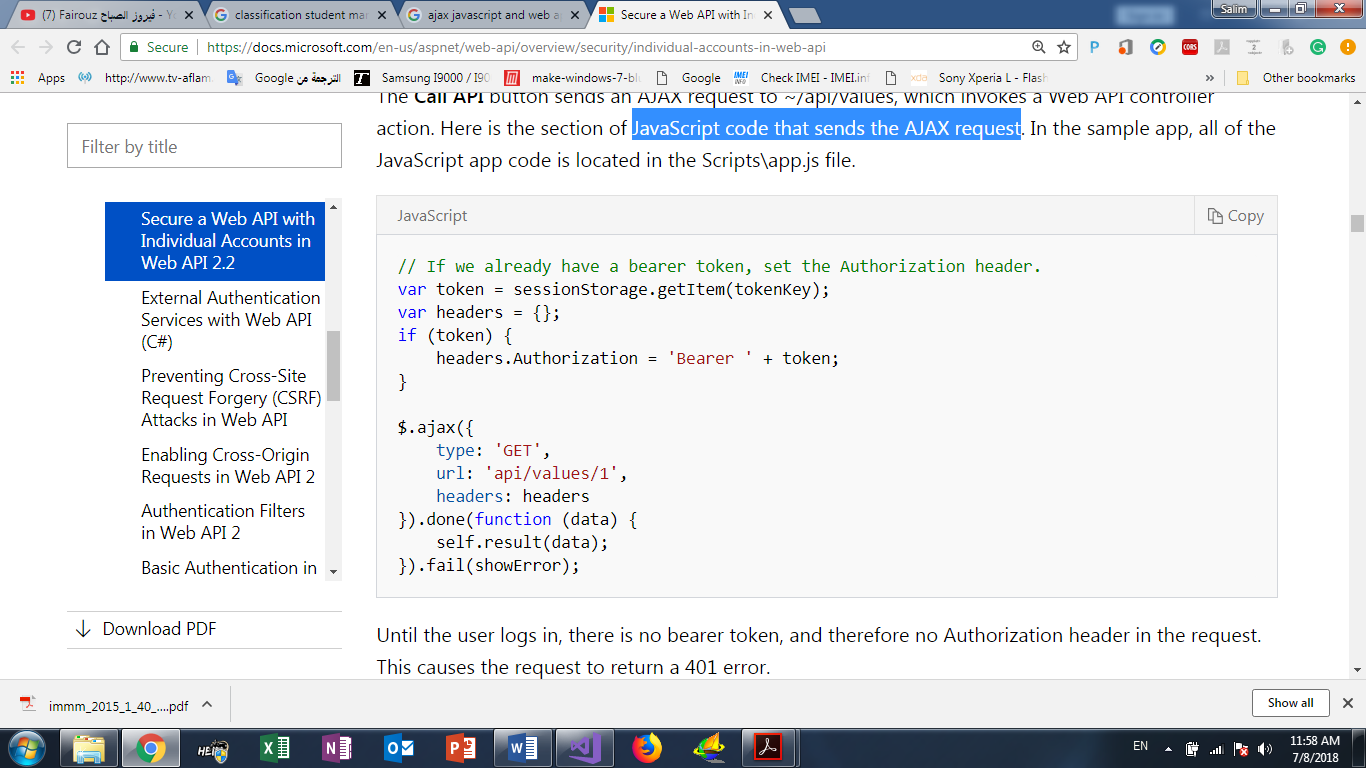
1. The user enters a name and password into the client.
2. The client sends these credentials to the authorization server.
3. The authorization server authenticates the credentials and returns an access token.
4. To access a protected resource, the client includes the access token in the Authorization header of the HTTP request.

When you select Individual accounts in the Web API project template, the project includes an authorization server that validates user credentials and issues tokens. The following diagram shows the same credential flow in terms of Web API components.

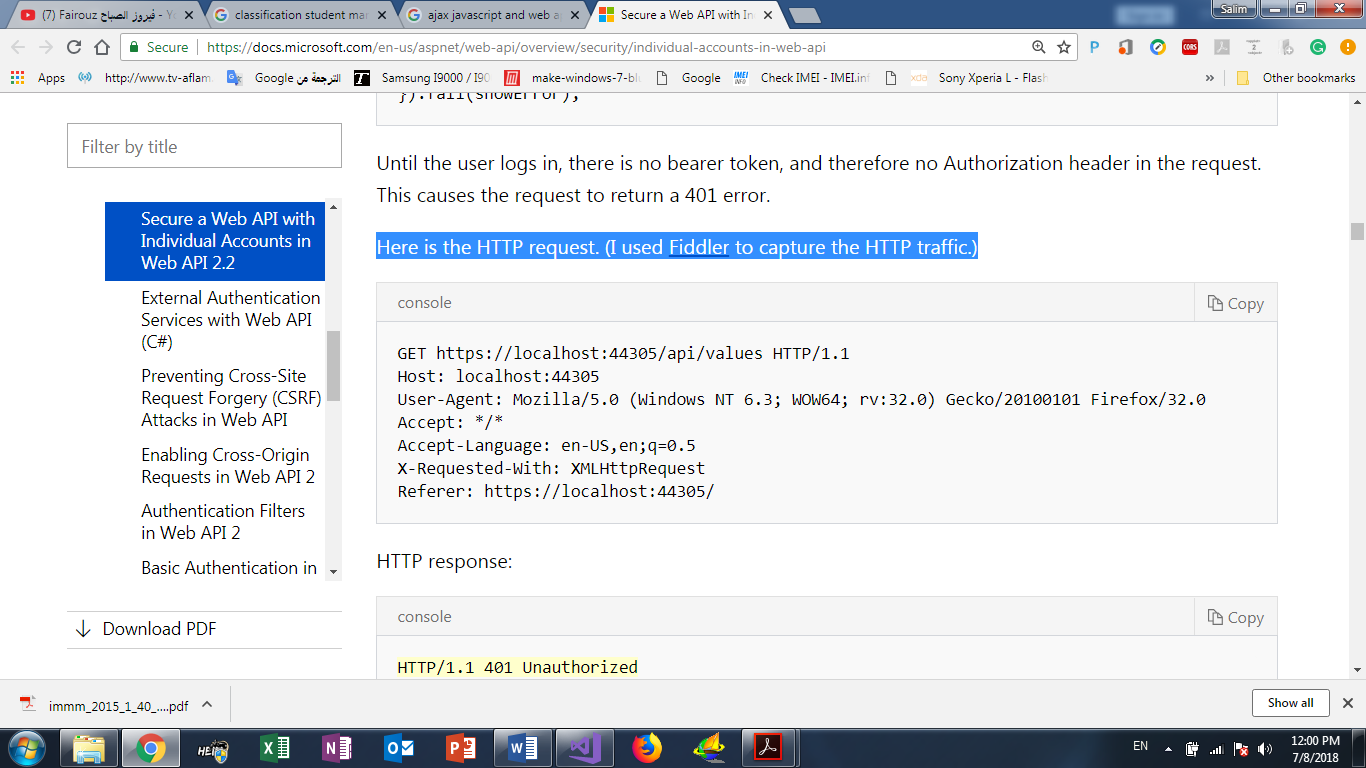
In this scenario, Web API controllers act as resource servers. An authentication filter validates access tokens, and the **[Authorize]** attribute is used to protect a resource. When a controller or action has the **[Authorize]** attribute, all requests to that controller or action must be authenticated. Otherwise, authorization is denied, and Web API returns a 401 (Unauthorized) error.

## **Sending an Unauthorized Request:**

JavaScript code that sends the AJAX request:



Until the user logs in, there is no bearer token, and therefore no Authorization header in the request. This causes the request to return a 401 error. Here is the HTTP request. (we used Fiddler to capture the HTTP traffic.)



HTTP response:

