

Report Title

REPORT SUBTITLE

Name | Course Title | Date

1. Authentication
2. [Implementation Redis Cache In the .NET Core API](https://www.c-sharpcorner.com/article/implementation-of-the-redis-cache-in-the-net-core-api/)

# Token Based Authentication

# **JWT Token Authentication Using The .NET Core 6 Web API**

1. [JWT Token Authentication for ASP.NET Core Web API](https://www.c-sharpcorner.com/article/jwt-token-authentication-using-the-net-core-6-web-api/)
2. [Azure AD Authentication for ASP.NET Core Web API](https://www.c-sharpcorner.com/article/enable-azure-ad-authentication-using-net-5-0-web-api/)

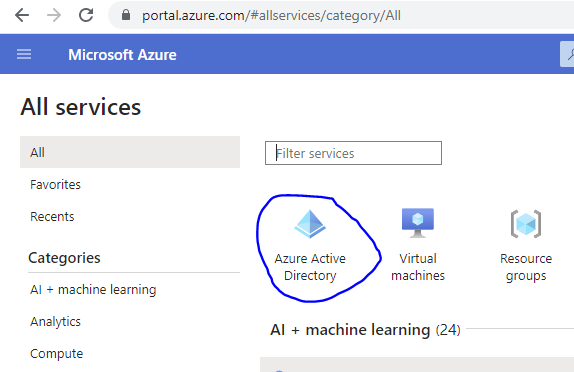
# [Secure Web API Azure AD B2C](https://learn.microsoft.com/en-us/samples/azure-samples/active-directory-aspnetcore-webapp-openidconnect-v2/how-to-secure-a-web-api-built-with-aspnet-core-using-the-azure-ad-b2c/)

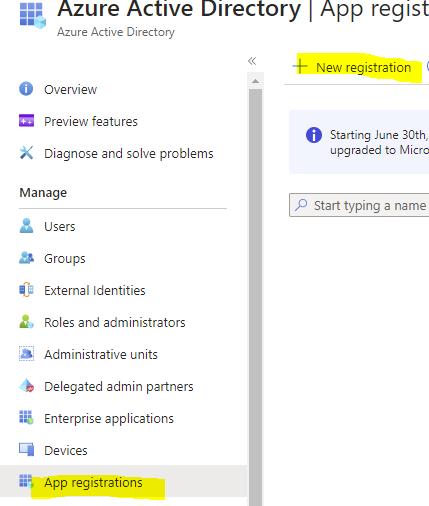
Steps to Configure this are,

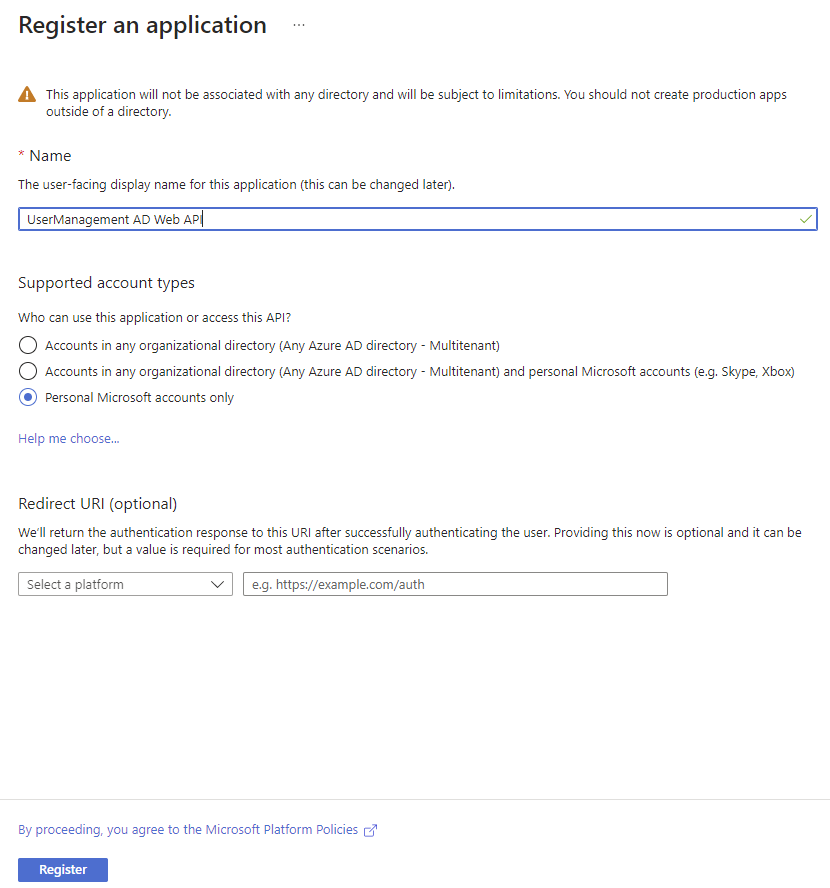
1. Create a Web API project with **Microsoft Identity Platform** - **Authentication type**
2. Register an Azure AD (AAD) app for the Web API.
3. Create a Scope for App registration (API)
4. Update the Web API Project to use Azure AD Authentication.
5. Configure the Redirect URL's (If you are testing with Postman)
6. Create a Client Secret.
7. Change the Authentication Type - Microsoft Identity Platform

## **Register an Azure AD (AAD) app for the Web API**

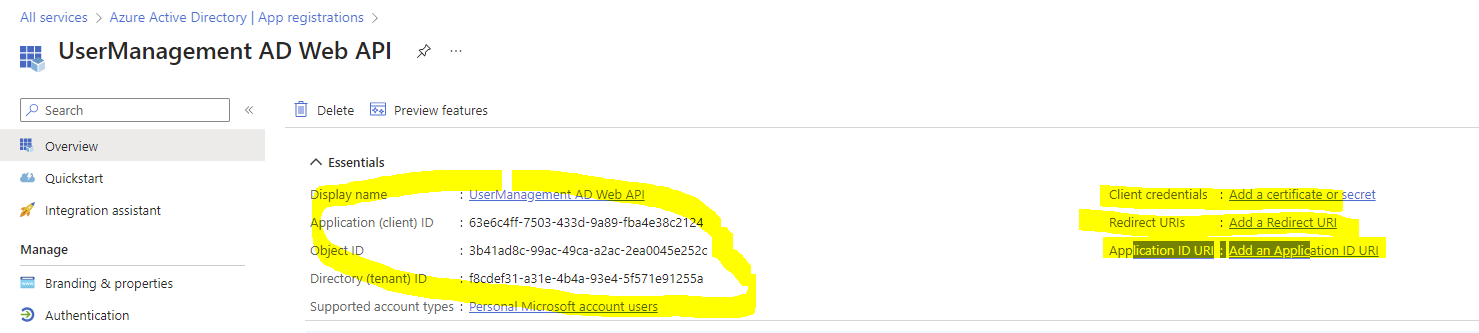
To authenticate against Azure AD you need to add the Azure AD app registration and this can be done through the Azure portal at http://portal.azure.com  >  Azure Active Directory > App registrations > New application registration.

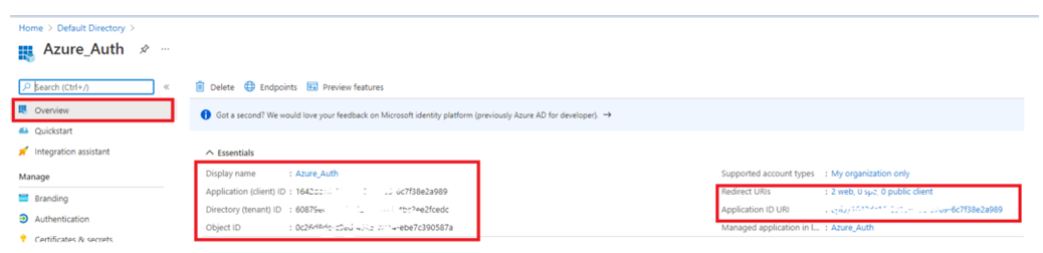






When the app registration is complete we can see the Client Id and Tenant Id in the Azure overview and copy those ID'd we will need in configuration setup.

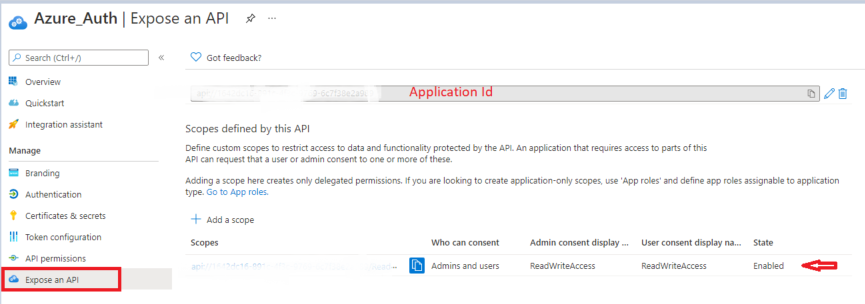


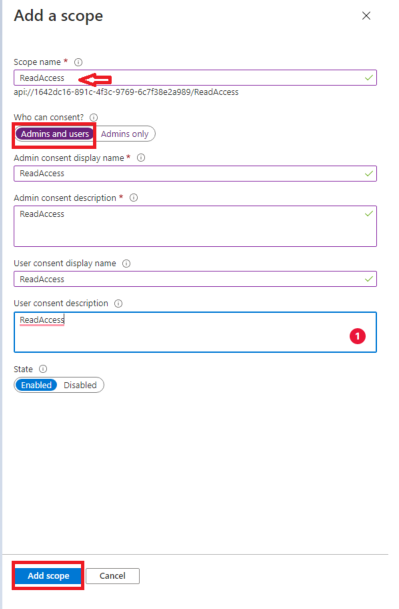


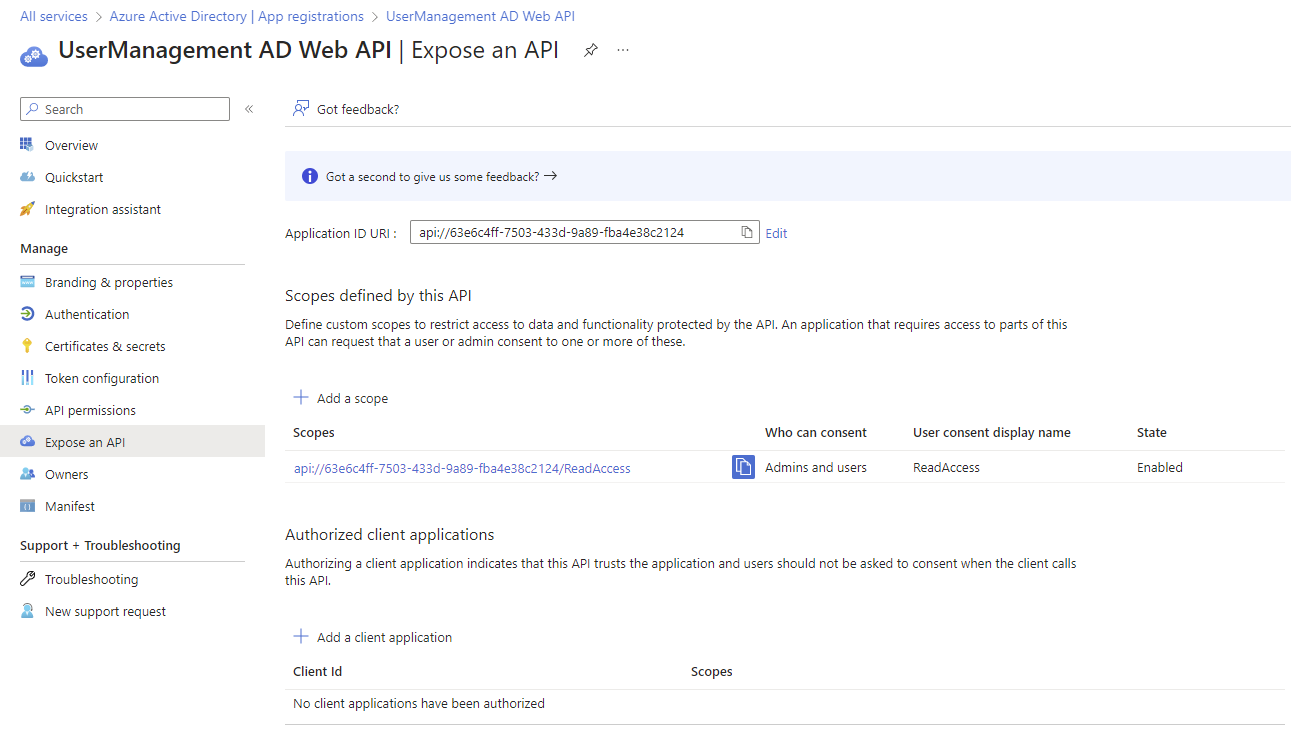
## **Create Scope for App registration**

The scope is required for authorizing the API like read and write access so we can define multiple scopes for our APIs if we are dealing with multiple projects or Microservices.

For that select the Expose an API  from the Azure \_Auth application. Click on the + Add Scope button. where it will open the popup to create the scopes.







## **Update the Web API Project to use Azure AD Authentication**

We need to configure the Azure active directory setup in appsettings.json like adding the application (client ID) and Tenant Id.

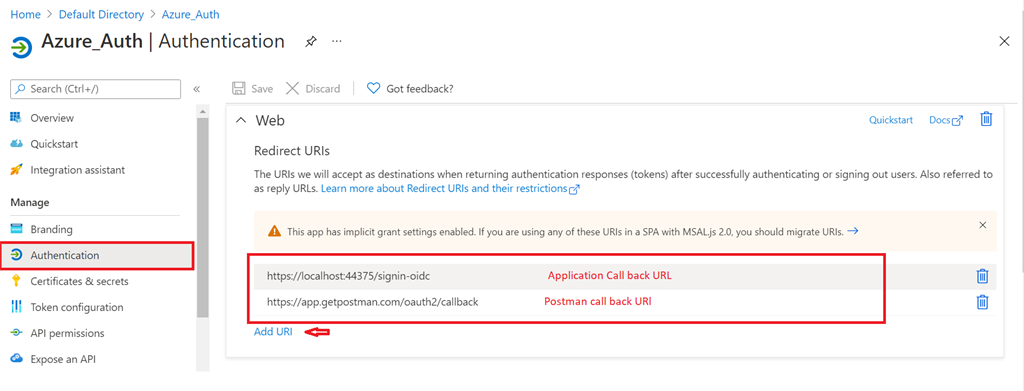
The Authentication type **Microsoft.identity.platform**helps in the integration of Azure AD and it will add all the basic configuration setup in the respective files.

**app.settings.json**

1. "AzureAd": {
2. "Instance": "https://login.microsoftonline.com/",
3. "Domain": "\*Your domain name\*", //Domain name configured in Azure
4. "TenantId": "0000-00000-00000-0000", // Tenant Id configured in Azure
5. "ClientId": "0000-00000-00000-0000", //  Client Id configured in Azure
6. "CallbackPath": "/signin-oidc"

## **Configure the Redirect URL's (If you are testing with Postman)**

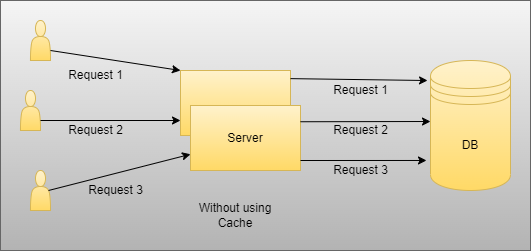
To test the APIs with Postman we need to configure the Callback URLs. Click on the Authentication menu, under the Platform Configurations, add the Redirect URLs - add the postman call back URL's - https://app.getpostman.com/oauth2/callback and application callback URL.

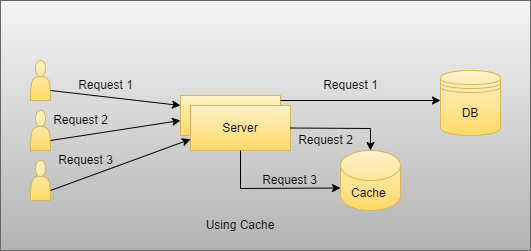


Implementation Of The Redis Cache In The .NET Core API

## **What is Caching?**

The cache is the memory storage that is used to store the frequent access data into the temporary storage, it will improve the performance drastically and avoid the unnecessary database hit and store frequently used data into the buffer whenever we need it.





As you see in the above image there are two scenarios, one is without using cache and another is with cache. So here when we do not use the cache, in that case, suppose users want data then they will hit each time database and it will increase the time complexity and reduce performance in case there is some static data users want and it is the same for all users. In that case when we do not use cache then each one hits the unnecessary database to fetch data. On the other side as you can see we use cache, and in that case if there is the same static and the same data for all users then only the first user will hit the database and fetch data and store it into the cache memory and then other two users used that from the cache without unnecessarily hit database to fetch data.

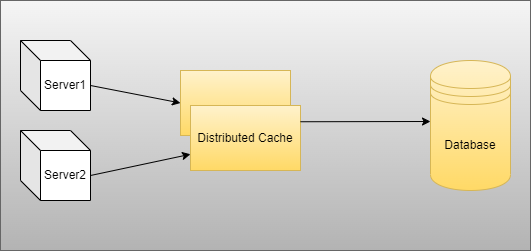
## **Types of Cache**

Basically, there are two types of caching .NET Core supports

1. In-Memory Caching
2. Distributed Caching

When we use In-Memory Cache then in that case data is stored in the application server memory and whenever we need then we fetch data from that and use it wherever we need it. And in Distributed Caching there are many third-party mechanisms like Redis and many others. But in this section, we look into the Redis Cache in detail and how it works in the .NET Core

### **Distributed Caching**



* Basically, in the distributed cachin,g data are stored and shared between multiple servers
* Also, it’s easy to improve scalability and performance of the application after managing the load between multiple servers when we use multi-tenant application
* Suppose, In the future, if one server is crashed and restarted then the application does not have any impact because multiple servers are as per our need if we want

Redis is the most popular cache which is used by many companies nowadays to improve the performance and scalability of the application. So, we are going to discuss Redis and usage one by one.

### **Redis Cache**

* Redis is an Open Source (BSD Licensed) in-memory Data Structure store used as a database.
* Basically, it is used to store the frequently used and some static data inside the cache and use and reserve that as per user requirement.
* There are many data structures present in the Redis which we are able to use like List, Set, Hashing, Stream, and many more to store the data.





