Use Redis Caching in ASP.NET Core Backend

Use Case:

Your **React app** makes API calls to your **ASP.NET Core Web API**, which caches responses using **Redis**.

Architecture Overview

```
[React Frontend]
       ↓
[ASP.NET Core Web API]
       ↓
[Redis Cache] ⟷ [SQL Server / DB]
```

Step-by-Step Guide

1. Install Required Packages

Install the Redis package:

```
dotnet add package StackExchange.Redis
```

---

2. Configure Redis in `Program.cs`

```
using StackExchange.Redis;

var builder = WebApplication.CreateBuilder(args);

// Add Redis
builder.Services.AddSingleton<IConnectionMultiplexer>(sp =>
{
    var configuration = builder.Configuration.GetConnectionString("Redis"
    return ConnectionMultiplexer.Connect(configuration);
});
```

```
builder.Services.AddControllers();
var app = builder.Build();
app.MapControllers();
app.Run();
```

---

3. Add Redis Connection String in `appsettings.json`

```
"ConnectionStrings": {
  "Redis": "localhost:6379"
}
```

> Replace `localhost:6379` with your Redis server details if running on Docker or cloud.

---

4. Sample Controller Using Redis Cache

```
using Microsoft.AspNetCore.Mvc;
using StackExchange.Redis;
using System.Text.Json;

[Route("api/[controller]")]
[ApiController]
public class ProductsController : ControllerBase
{
    private readonly IConnectionMultiplexer _redis;

    public ProductsController(IConnectionMultiplexer redis)
    {
        _redis = redis;
    }

    [HttpGet("{id}")]
    public async Task<IActionResult> GetProduct(int id)
    {
        var db = _redis.GetDatabase();
        string cacheKey = $"product:{id}";

        // 1. Check cache
        var cachedData = await db.StringGetAsync(cacheKey);
        if (cachedData.HasValue)
        {
```

```
            Console.WriteLine("From Cache");
            var product = JsonSerializer.Deserialize<Product>(cachedData)
            return Ok(product);
        }

        // 2. Simulate DB call
        var productFromDb = new Product { Id = id, Name = "Laptop", Price

        // 3. Save to cache with TTL
        await db.StringSetAsync(cacheKey, JsonSerializer.Serialize(produc

        Console.WriteLine("From DB, Saved to Cache");
        return Ok(productFromDb);
    }
}

public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public int Price { get; set; }
}
```

---

5. React Frontend (Same as before)

Just call your ASP.NET API:

```
axios.get("http://localhost:5000/api/products/1")
```

Run & Test

1. Start Redis server (e.g., `redis-server`)
2. Run your ASP.NET Core backend
3. Call the API from React or Postman
4. First call: logs "From DB, Saved to Cache"
5. Second call: logs "From Cache" — fast!

Summary

Step	Description
Install `StackExchange.Redis`	Official Redis client
Configure in `Program.cs`	Setup singleton connection
Use in Controller	Read/write using `StringGetAsync`, `StringSetAsync`
Set TTL	Expire old data automatically