**Walkthrough – Using Redis Cache in Microservices**

**Project Setup**

1. Right Click on Services Folder 🡪 Add 🡪 **New Project**
2. Select ASP.NET Core Web Application, Project name=**ShoppingBasket.API**, Location=D:\DemoSolution\**Services\ShoppingBasket** 🡪 Create
3. Select **API** 🡪 Create
4. Add NuGet Package: **StackExchange.Redis**
5. Edit appSettings.json

{

"**RedisConnectionString**": "localhost:6300"

}

Note: The above setting assumes that the Redis service is going to run in a Docker container whose port is mapped to 6300 on localhost as below

1. Start the Redis cache docker container

docker run -d -p **6300:6379** redis

1. **Edit Program.cs and add the following code below:** builder.Services.AddSwaggerGen();

builder.Services.AddSingleton<**ConnectionMultiplexer**>(sp =>

{

var configuration = ConfigurationOptions.Parse(Configuration["RedisConnectionString"], true);

configuration.ResolveDns = true;

// OR

//configuration.ResolveDns = false; // If Azure Redis Cache is used.

configuration.AbortOnConnectFail = false;

return ConnectionMultiplexer.Connect(configuration);

});

1. Add Models folder to the project.

**Adding Models**

1. **Add Models/BasketItems.cs**

public class BasketItem

{

public string Id { get; set; }

public int ProductId { get; set; }

public string ProductName { get; set; }

public decimal UnitPrice { get; set; }

public int Quantity { get; set; }

public string PictureUrl { get; set; }

}

1. **Add Models/Basket.cs**

public class Basket

{

public string BuyerId { get; set; }

public List<BasketItem> Items { get; set; }

}

1. **Add Services/IBasketRepository.cs**

public interface IBasketRepository

{

Task<Basket> GetBasketAsync(string buyerId);

IEnumerable<string> GetBuyers();

Task<Basket> UpdateBasketAsync(Basket basket);

Task<bool> DeleteBasketAsync(string id);

}

1. **Add Services/BasketRepository.cs**

public class RedisBasketRepository : IBasketRepository

{

private readonly ConnectionMultiplexer \_redis;

private readonly IDatabase \_database;

public RedisBasketRepository(**ConnectionMultiplexer** redis)

{

\_redis = redis;

\_database = redis.GetDatabase();

}

public async Task<bool> DeleteBasketAsync(string id)

{

return await \_database.KeyDeleteAsync(id);

}

public async Task<Basket> GetBasketAsync(string buyerId)

{

var data = await \_database.StringGetAsync(buyerId);

if (data.IsNullOrEmpty)

{

return null;

}

return JsonConvert.DeserializeObject<Basket>(data);

}

public IEnumerable<string> GetBuyers()

{

var endpoint = \_redis.GetEndPoints();

var server = \_redis.GetServer(endpoint.First());

var data = server.Keys();

return data?.Select(k => k.ToString());

}

public async Task<Basket> UpdateBasketAsync(Basket basket)

{

var created = await \_database.StringSetAsync(basket.BuyerId, JsonConvert.SerializeObject(basket));

if (!created)

{

throw new ApplicationException("Basket cannot be saved...");

}

return await GetBasketAsync(basket.BuyerId);

}

}

1. Edit ConfigureService method in Starup.cs

services.**AddTransient**<IBasketRepository, RedisBasketRepository>();

**Adding Controller Service**

1. Right click on Controllers Folder 🡪 Add 🡪 Controller…
2. Select **MVC Controller - Empty** Controller 🡪 Name = **BasketController** 🡪 Add
3. Edit BasketController.cs

[Route("[controller]")]

[ApiController]

public class BasketController : ControllerBase

{

private IBasketRepository \_repository;

public BasketController(**IBasketRepository** repository)

{

\_repository = repository;

}

// GET basket/5

[HttpGet("{id}")]

[ProducesResponseType(typeof(Basket), (int)HttpStatusCode.OK)]

public async Task<IActionResult> **Get**(string id)

{

var basket = await \_repository.GetBasketAsync(id);

return Ok(basket);

}

// POST basket

[HttpPost]

[ProducesResponseType(typeof(Basket), (int)HttpStatusCode.OK)]

public async Task<IActionResult> **Post**([FromBody]Basket value)

{

var basket = await \_repository.UpdateBasketAsync(value);

return Ok(basket);

}

// DELETE bakset/5

[HttpDelete("{id}")]

public void **Delete**(string id)

{

\_repository.DeleteBasketAsync(id);

}

}

1. Run and test the API using <http://localhost:5200/swagger>
   1. Add a BasketItem and then Get the BasketItem.

**Docker-compose**

1. ShoppingBasket.API 🡪 Properties 🡪 launchsettings.json

Change HTTP Port No = 5200

1. Append the following to Docker-compose.yml

redis:

image: redis

networks:

- backend

1. **Run docker-compose up and ensure it doesn’t have any error**

**Editing WebMVC (Client) Application**

1. Edit appsettings.json

{

"CatalogBaseUrl": "http://localhost:5100",

**"CartBaseUrl":** "http://localhost:5200"

}

1. Add **Models\BasketItem.cs**

public class BasketItem

{

public string Id { get; set; }

public int ProductId { get; set; }

public string ProductName { get; set; }

public decimal UnitPrice { get; set; }

public int Quantity { get; set; }

public string PictureUrl { get; set; }

}

1. Add **Models\Basket.cs**

public class Basket

{

public List<BasketItem> Items { get; set; } = new List<BasketItem>();

public string BuyerId { get; set; }

public decimal Total()

{

return Math.Round(Items.Sum(x => x.UnitPrice \* x.Quantity), 2);

}

}

**Adding BasketService**

1. **Add Services\IBasketService.cs**

public interface IBasketService

{

Task<Basket> GetBasket(string userId);

Task AddItemToBasket(string userId, BasketItem product);

Task<Basket> UpdateBasket(Basket basket);

Task ClearBasket(string userId);

}

1. **Add Services\BasketService.cs**

public class **BasketService** : IBasketService

{

private readonly string \_remoteServiceBaseUrl;

public BasketService(IConfiguration config)

{

\_remoteServiceBaseUrl = config["CartBaseUrl"];

}

public async Task **AddItemToBasket**(string userId, BasketItem product)

{

var basket = await GetBasket(userId);

var basketItem = basket.Items

.Where(p => p.ProductId == product.ProductId)

.FirstOrDefault();

if (basketItem == null)

{

basket.Items.Add(product);

}

else

{

basketItem.Quantity += 1;

}

await UpdateBasket(basket);

}

public async Task **ClearBasket**(string userId)

{

var client = new HttpClient();

await client.DeleteAsync(\_remoteServiceBaseUrl + "/basket/" + userId);

}

public async Task<Basket> **GetBasket**(string userId)

{

var client = new HttpClient();

var dataString = await client.GetStringAsync(\_remoteServiceBaseUrl + "/basket/" + userId);

var response = JsonConvert.DeserializeObject<Basket>(dataString.ToString());

if (response == null)

{

response = new Basket()

{

BuyerId = userId,

Items = new List<BasketItem>()

};

}

return response;

}

public async Task<Basket> **UpdateBasket**(Basket basket)

{

var client = new HttpClient();

HttpContent content = new StringContent(JsonConvert.SerializeObject(basket), System.Text.Encoding.UTF8, "application/json");

var response = await client.PostAsync(\_remoteServiceBaseUrl + "/basket/", content);

response.EnsureSuccessStatusCode();

return basket;

}

}

1. **Add the following to ConfigureService method in StartUp.cs**

services.AddTransient<IBasketService, BasketService>();

**Adding BasketController:**

1. **Right Click on Controllers Folder 🡪 Add MVC Controller – Empty, Name = BasketController.cs**

public class BasketController : Controller

{

IBasketService \_basketService;

ICatalogService \_catalogService;

public BasketController(IBasketService basketService, ICatalogService catalogService)

{

\_basketService = basketService;

\_catalogService = catalogService;

}

public async Task<IActionResult> Index()

{

var userId = "DemoUser";

var basket = await \_basketService.GetBasket(userId);

return View(basket);

}

public async Task<IActionResult> AddToBasket(int productId)

{

CatalogItem catItem = await \_catalogService.GetItemDetails(productId);

if (catItem != null)

{

var userId = "DemoUser";

var product = new BasketItem()

{

Id = Guid.NewGuid().ToString(),

Quantity = 1,

ProductName = catItem.Name,

PictureUrl = catItem.PictureUrl,

UnitPrice = catItem.Price,

ProductId = catItem.Id

};

await \_basketService.AddItemToBasket(userId, product);

}

return RedirectToAction("Index", "Catalog");

}

}

1. Right Click on **Index** method of BasketController 🡪 Add View… 🡪 View name = Index, Template="**Details**", Model class = **Basket** (WebMvc.Models) 🡪 Add
2. Right Click on **Index** method of BasketController 🡪 Add View… 🡪 View name = BasketItems, Template="**List**", Model class = **BasketItem** (WebMvc.Models) 🡪 **Check Create as Partial View** 🡪 Add
3. To Views\Basket\Index.html append

<div>

<**partial** **name**="BasketItems" **model**="Model.Items" />

</div>

1. Edit Views\Catalog\Index.html and add the following for each row

Below this line = @Html.ActionLink("Details", "Details", new { id = item.Id })

@Html.ActionLink("AddToCart", "AddToBasket","Basket", new { productId=item.Id })

1. Edit Views\Shared\\_Layout.cshtml

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Basket" **asp-action**="Index">My Basket</a>

</li>

1. Run the applications and Test "Add To Cart" functionality