Configuring the Trading microservice

Script start

Let’s now add the proper startup configuration to the Trading microservice so that it is ready to receive requests and also to enable our state machine to start interacting with MongoDB and RabbitMQ.

In Trading repo

1. Update appsettings.json:

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "ServiceSettings": {
    "ServiceName": "Trading",
    "Authority": "https://localhost:5003"
  },
  "MongoDbSettings": {
    "Host": "localhost",
    "Port": "27017"
  },
  "RabbitMQSettings": {
    "Host": "localhost"
  },
  "AllowedHosts": "*"
}
```

2. Update appsettings.Development.json:

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Debug",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  }
}
```

3. Add NuGet packages:

```
dotnet add package Play.Common
dotnet add package MassTransit.AspNetCore
dotnet add package MassTransit.RabbitMQ
dotnet add package MassTransit.MongoDB
```

4. Update Startup.cs:

```
public class Startup
{
  …
  public void ConfigureServices(IServiceCollection services)
  {
    services.AddMongo()
           .AddJwtBearerAuthentication();
    AddMassTransit(services);

    services.AddControllers();
    services.AddSwaggerGen(c =>
    {
      c.SwaggerDoc("v1", new OpenApiInfo { Title = "Play.Trading.Service", Version = "v1" });
    });
  }

  public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
  {
    …
    app.UseRouting();

    app.UseAuthentication();
    app.UseAuthorization();
    …
  }

  private void AddMassTransit(IServiceCollection services)
  {
    services.AddMassTransit(configure =>
    {
      configure.UsingPlayEconomyRabbitMq();
      configure.AddSagaStateMachine<PurchaseStateMachine, PurchaseState>()
        .MongoDbRepository(r =>
        {
                var serviceSettings =
Configuration.GetSection(nameof(ServiceSettings)).Get<ServiceSettings>();
                var mongoDbSettings =
Configuration.GetSection(nameof(MongoDbSettings)).Get<MongoDbSettings>();

                r.Connection = mongoDbSettings.ConnectionString;
                r.DatabaseName = serviceSettings.ServiceName;
            });
        });

        services.AddMassTransitHostedService();
    }
}
```

In Identity repo

5. Update appserttings.json:

```
"IdentityServerSettings": {
  "ApiScopes": [
    ...
    {
      "Name": "inventory.fullaccess"
    },
    {
      "Name": "trading.fullaccess"
    },
    {
      "Name": "IdentityServerApi"
    }
  ],
  "ApiResources": [
    {
    ...
    {
      "Name": "Inventory",
      "Scopes": [
        "inventory.fullaccess"
      ],
      "UserClaims": [
        "role"
      ]
    },
    {
      "Name": "Trading",
      "Scopes": [
        "trading.fullaccess"
      ],
      "UserClaims": [
        "role"
      ]
    }
  ]
},
```

6. Update appsettings.Development.json:

```
{
  "ClientId": "postman",
   …
  "AllowedScopes": [
    …
    "inventory.fullaccess",
    "trading.fullaccess",
    "IdentityServerApi",
    …
  ],
  …
}
```

In the next lesson we will post a purchase request to our controller and trigger the purchase state machine for the first time.