Generalizing the Open Telemetry configuration

Start

Let's generalize the Open Telemetry tracing configuration so that it becomes easy for each of our microservices to send distributed traces to Jaeger.

In Common repo

1. Add the OpenTelemetry package references to Play.common.csproj:

```
<Project Sdk="Microsoft.NET.Sdk">
  …
  <ItemGroup>
    …
    <PackageReference Include="MongoDB.Driver" Version="2.11.6" />
    <PackageReference Include="OpenTelemetry" Version="1.2.0" />
    <PackageReference Include="OpenTelemetry.Exporter.Jaeger" Version="1.2.0" />
    <PackageReference Include="OpenTelemetry.Extensions.Hosting" Version="1.0.0-rc9.2" />
    <PackageReference Include="OpenTelemetry.Instrumentation.AspNetCore" Version="1.0.0-rc9.2" />
    <PackageReference Include="OpenTelemetry.Instrumentation.Http" Version="1.0.0-rc9.2" />
    <PackageReference Include="Seq.Extensions.Logging" Version="6.0.0" />
  </ItemGroup>
</Project>
```

2. Copy JaegerSettings.cs from Trading to Common, under the Settings directory (fix namespace)

3. Create an OpenTelemetry directory

4. Add Extensions.cs under new directory:

```
namespace Play.Common.OpenTelemetry
{
    public static class Extensions
    {
        public static IServiceCollection AddTracing(this IServiceCollection services, IConfiguration config)
        {
            services.AddOpenTelemetryTracing(builder =>
            {
                var serviceSettings = config.GetSection(nameof(ServiceSettings))
                                            .Get<ServiceSettings>();
                builder.AddSource(serviceSettings.ServiceName)
                    .AddSource("MassTransit")
                    .SetResourceBuilder(
                        ResourceBuilder.CreateDefault()
```

```
                        .AddService(serviceName: serviceSettings.ServiceName))
                .AddHttpClientInstrumentation()
                .AddAspNetCoreInstrumentation()
                .AddJaegerExporter(options =>
                {
                    var jaegerSettings = config.GetSection(nameof(JaegerSettings)).Get<JaegerSettings>();
                    options.AgentHost = jaegerSettings.Host;
                    options.AgentPort = jaegerSettings.Port;
                });
        });

        return services;
    }
  }
}
```

5. Add ConsumeObserver.cs under MassTransit:

```
namespace Play.Common.MassTransit
{
  public class ConsumeObserver : IConsumeObserver
  {
    public Task ConsumeFault<T>(ConsumeContext<T> context, Exception exception) where T : class
    {
      Activity.Current.SetStatus(Status.Error.WithDescription(exception.Message));
      return Task.CompletedTask;
    }

    public Task PostConsume<T>(ConsumeContext<T> context) where T : class
    {
      return Task.CompletedTask;
    }

    public Task PreConsume<T>(ConsumeContext<T> context) where T : class
    {
      return Task.CompletedTask;
    }
  }
}
```

6. Register the observer in the just created AddTracing method:

```
public static IServiceCollection AddTracing(this IServiceCollection services, IConfiguration config)
{
```

```
    services.AddOpenTelemetryTracing(builder =>
    {
        …
    })
    .AddConsumeObserver<ConsumeObserver>();

    return services;
}
```

7. Commit and push

8. Wait for GitHub workflow to complete

In the next lesson you will use the new NuGet package in all your microservices