Extending the Message Pipeline with MassTransit Middleware

Roland Guijt
INDEPENDENT SOFTWARE DEVELOPER AND TRAINER

@rolandguijt www.rmgsolutions.nl

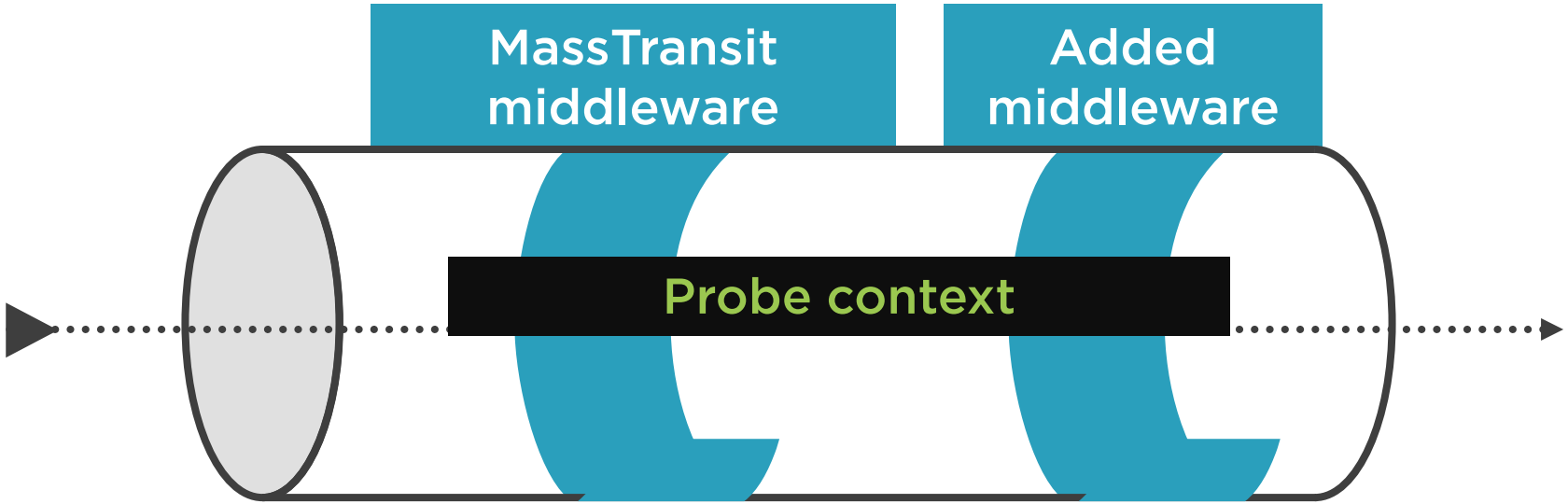
Module Overview

- Middleware and pipeline
- Composition of middleware
- Circuit breaker
- Rate limiter
- Latest filter

Pipelines

- Used to process messages
- Consist of asynchronous middleware
- Configure using configurator
- Extend the pipeline
- Out-of-the-box or custom-made

A Pipeline



Composition of Middleware (1/3)

```
public class MyFilter<T> : IFilter<T>
    where T : class, PipeContext
{
        public void Probe(ProbeContext context)
        {
            //Manipulate probe context
        }

         public async Task Send(T context, IPipe<T> next)
        {
            //do something before next middlewares
            await next.Send(context);
            //do something after next middlewares
        }
}
```

Composition of Middleware (2/3)

```
public class MyFilterSpec<T>: IPipeSpecification<T>
   where T : class, PipeContext
{
      public IEnumerable<ValidationResult> Validate()
      {
         //perform validation
      }

      public void Apply(IPipeBuilder<T> builder)
      {
         builder.AddFilter(new MyFilter<T>())
      }
}
```

Composition of Middleware (3/3)

```
public static class ExampleMiddlewareConfiguratorExtensions
{
   public static void UseMyFilter<T>(
       this IPipeConfigurator<T> configurator)
           where T : class, PipeContext
   {
       configurator.AddPipeSpecification(
           new MyFilterSpec<T>());
   }
}
```

Adding Middleware to the Pipeline

```
var bus = BusConfigurator.ConfigureBus((cfg, host) =>
{
   cfg.ReceiveEndpoint(host,queuename, e =>
   {
      e.Consumer<ConsumerType>();
   });
});
```

Adding Middleware to the Pipeline

```
var bus = BusConfigurator.ConfigureBus((cfg, host) =>
{
   cfg.UseMyFilter();
   cfg.ReceiveEndpoint(host,queuename, e =>
   {
      e.Consumer<ConsumerType>();
   });
});
```

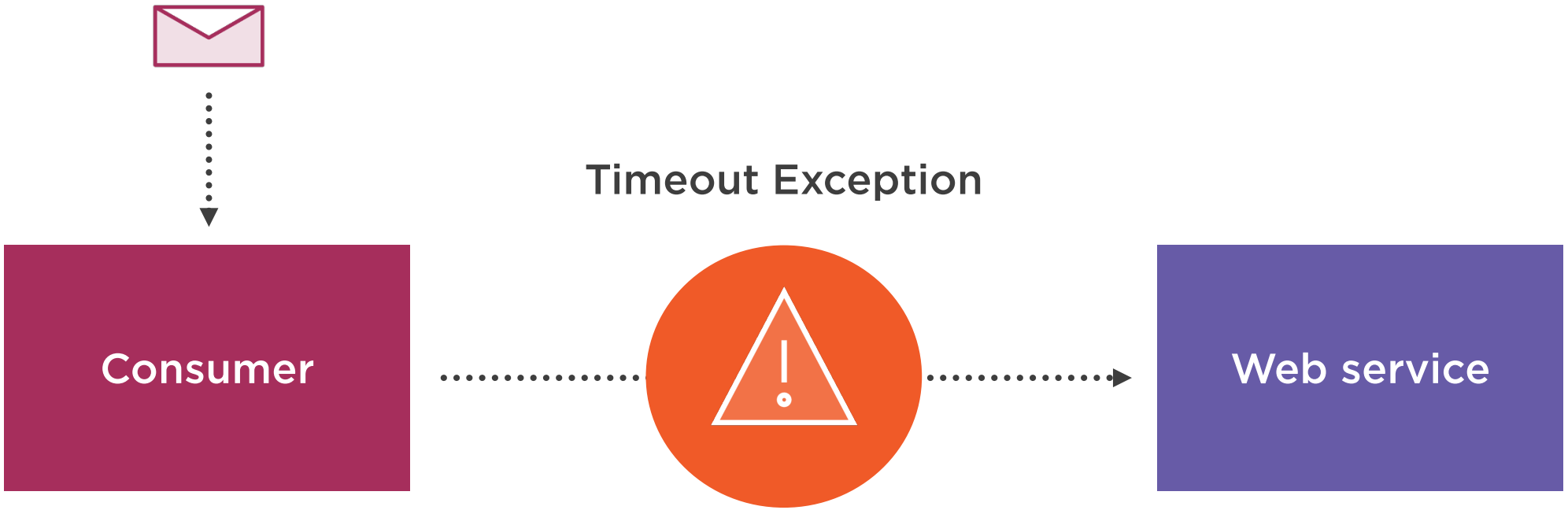
Adding Middleware to the Pipeline

```
var bus = BusConfigurator.ConfigureBus((cfg, host) =>
{
   cfg.ReceiveEndpoint(host,queuename, e =>
   {
      e.UseMyFilter();
      e.Consumer<ConsumerType>();
   });
});
```

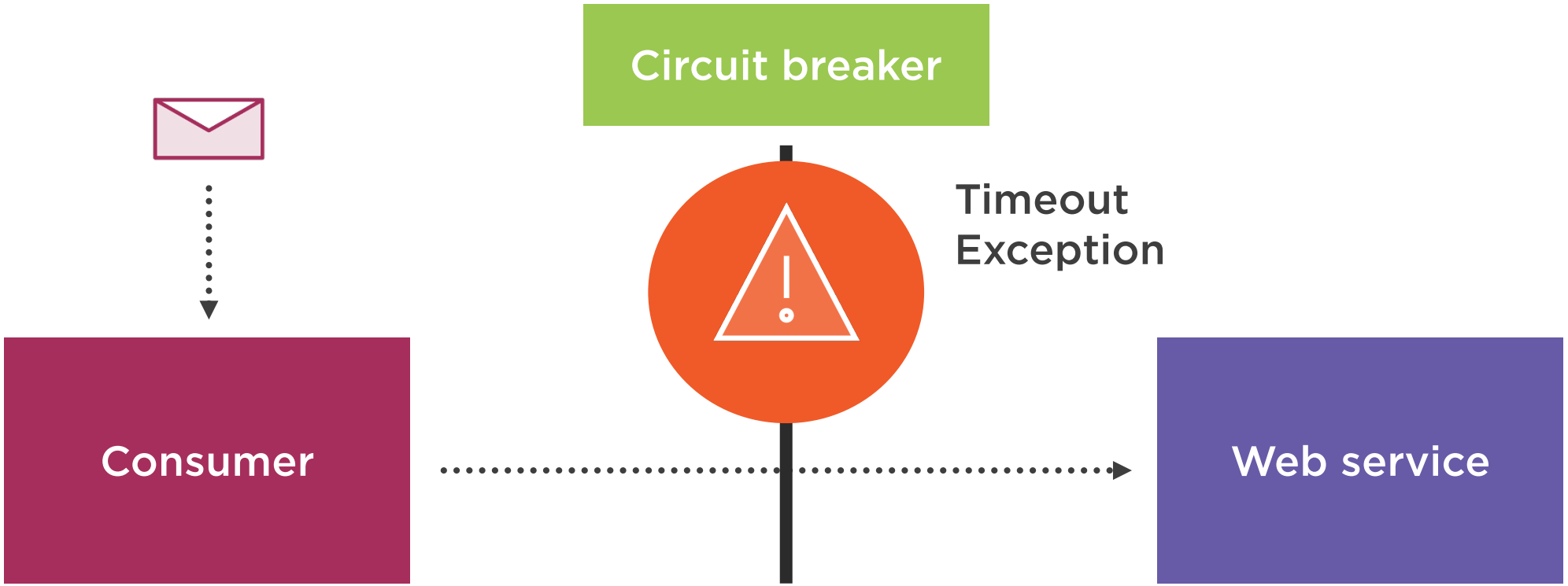
Adding Middleware to the Pipeline

```
var bus = BusConfigurator.ConfigureBus((cfg, host) =>
{
   cfg.ReceiveEndpoint(host,queuename, e =>
   {
      e.Consumer<ConsumerType>(c => c.UseMyFilter());
   });
});
```

Circuit Breaker



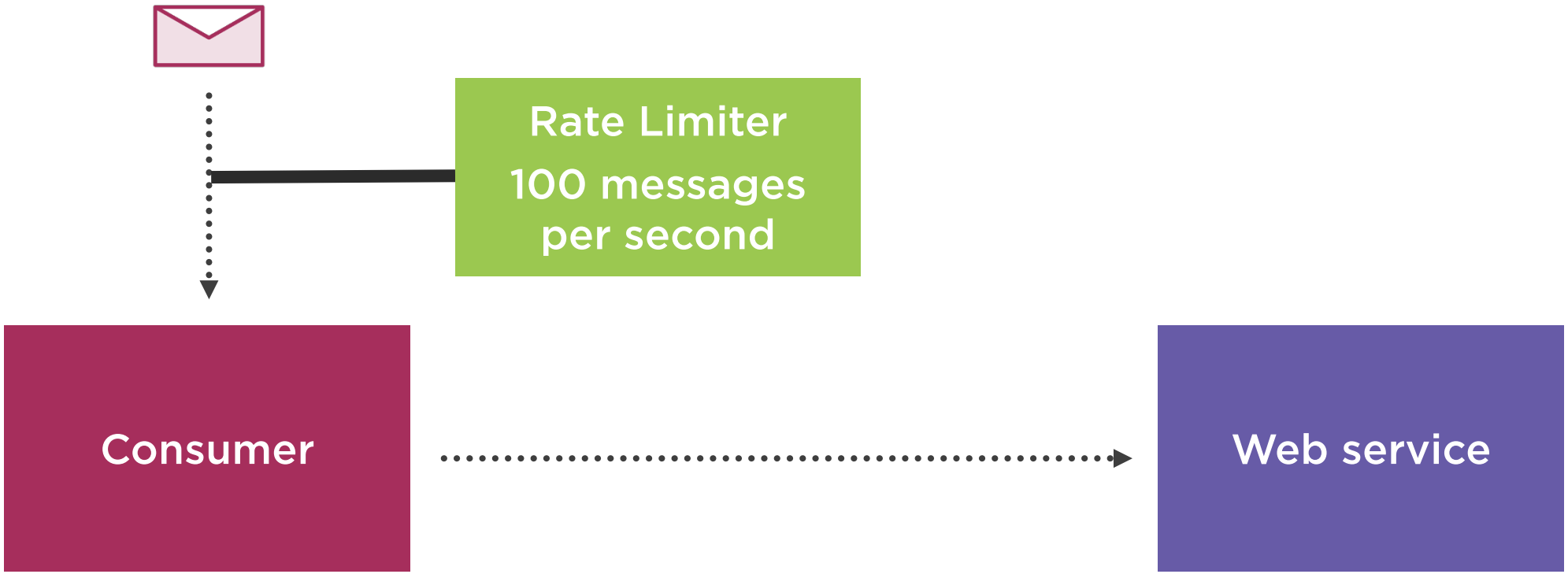
Circuit Breaker



Circuit Breaker in Code

```
cfg.ReceiveEndpoint(host, queue, e =>
{
    e.UseCircuitBreaker(cb =>
    {
        cb.TripThreshold = 15;
        cb.ResetInterval = TimeSpan.FromMinutes(5);
        cb.TrackingPeriod = TimeSpan.FromMinutes(1);
        cb.ActiveThreshold = 10;
    });
}
```

Rate Limiter



Rate Limiter in Code

```
cfg.ReceiveEndpoint(host, queuename, e =>
{
    e.UseRateLimit(100, TimeSpan.FromSeconds(1));
});
```

Latest Filter in Code

```
//class level
private ILatestFilter<ConsumeContext> latestContext;


cfg.ReceiveEndpoint(host, queuename, e =>

{

    e.UseLatest(lg => lg.Create =

        filter => latestContext = filter);

});
```

Summary

- How middleware and pipeline fit together
- Middleware composition
- Circuit breaker
- Rate limiter
- Latest filter