

ПОВАР ЯКОВ

СИСТЕМЫ ОБМЕНА СООБЩЕНИЯМИ

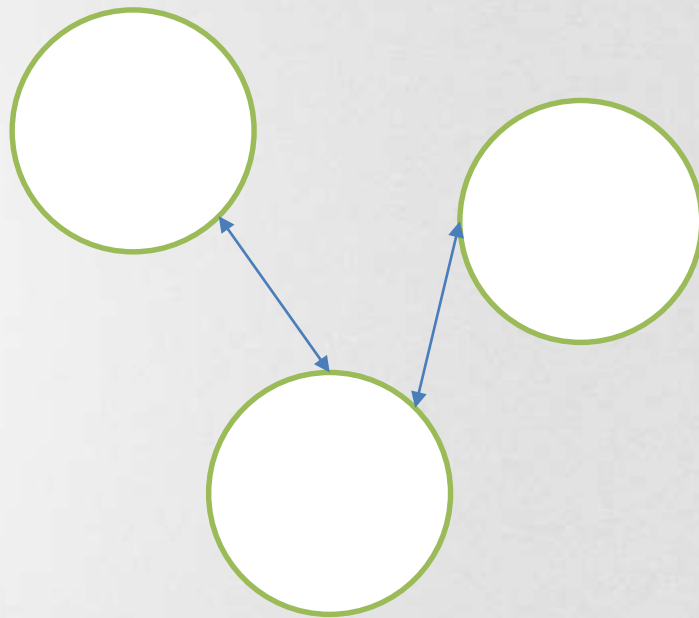
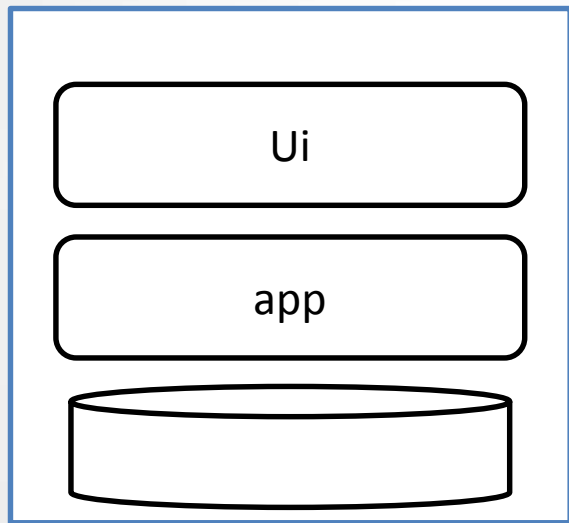
НА ПРИМЕРЕ БИБЛИОТЕКИ MASSTRANSIT



СОДЕРЖАНИЕ

- ТИПЫ ИНТЕГРАЦИИ
- ОБМЕН СООБЩЕНИЯМИ
- RABBITMQ
- MASSTRANSIT: ПРОСТОЕ
- MASSTRANSIT: ПОСЛОЖНЕЕ

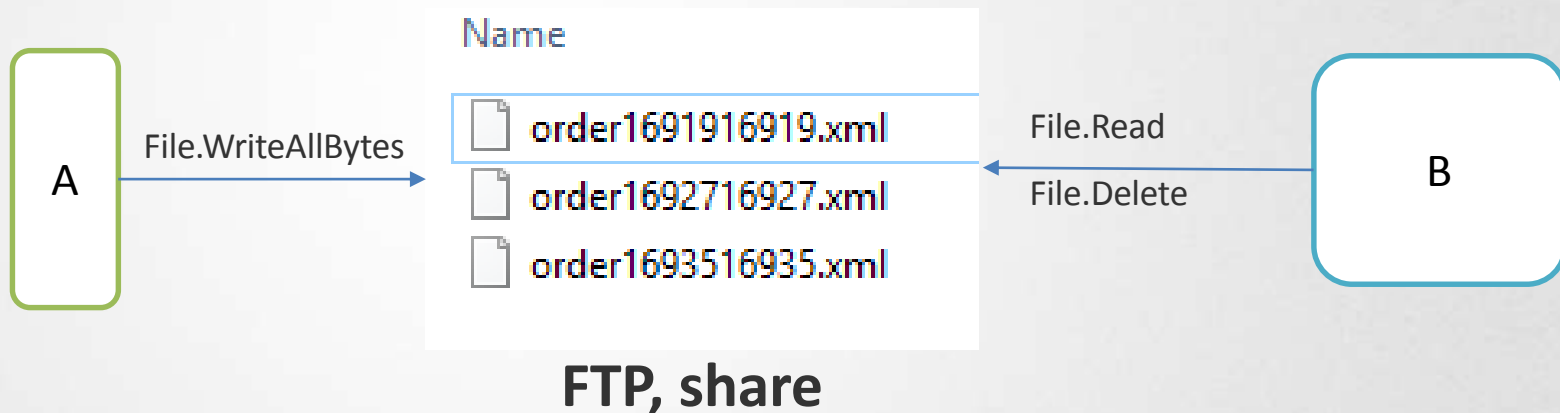
ЗАЧЕМ?



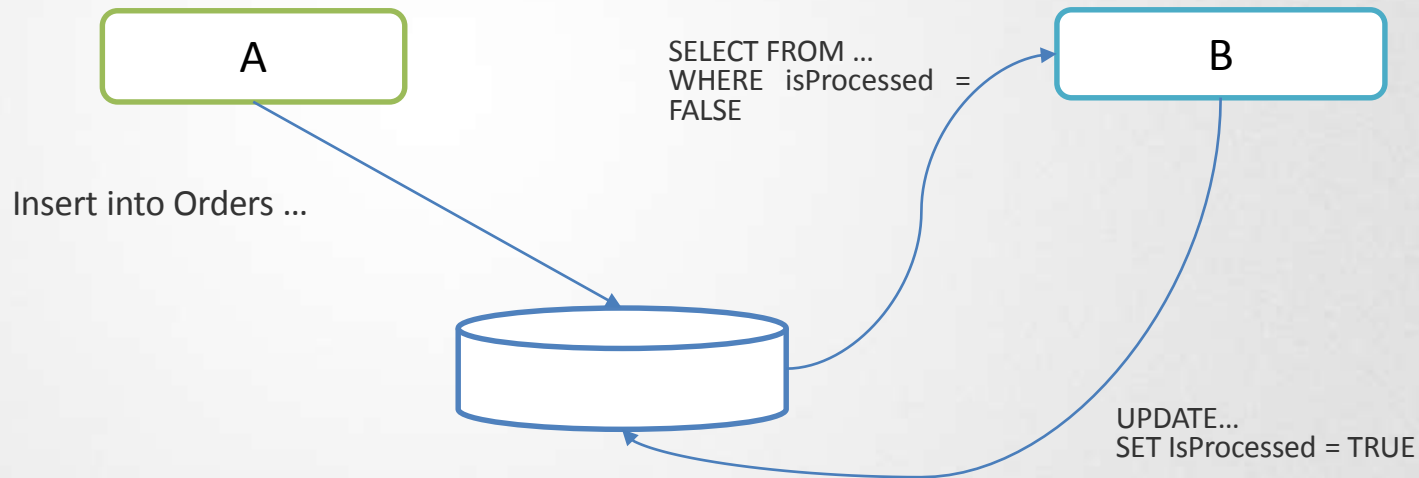
ИНТЕГРАЦИЯ ПРИЛОЖЕНИЙ

- РАСПРЕДЕЛЕНИЕ НАГРУЗКИ
- ОТЛОЖЕННАЯ ОБРАБОТКА
- РАЗНЕСЕНИЕ ОТВЕТСТВЕННОСТИ
- СУЩЕСТВУЮЩИЕ ПРИЛОЖЕНИЯ

ОБМЕН ФАЙЛАМИ

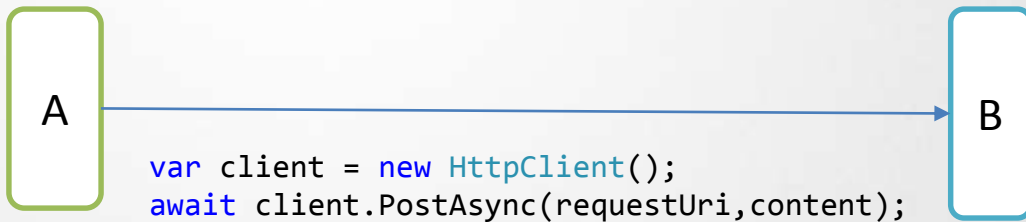


ОБЩАЯ БАЗА

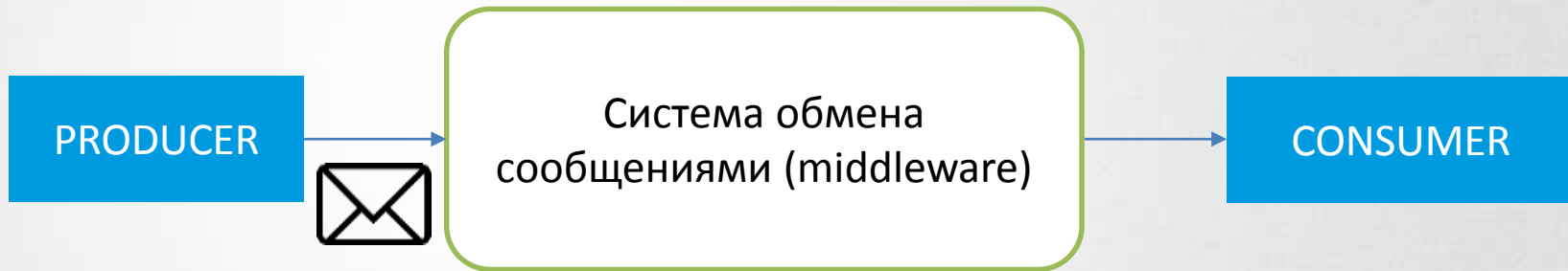


RPC/REST

- WEB-API
- WCF
- ServiceStack
- ...



MESSAGING



MESSAGING

КАКИЕ ПЛЮСЫ

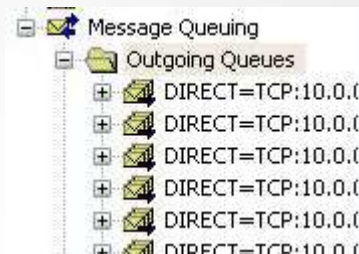
- АСИНХРОННОСТЬ ВЗАИМОДЕЙСТВИЯ
- ЧЕТКО ОПРЕДЕЛЕННЫЙ КОНТРАКТ
- МИНИМИЗИРОВАНА СВЯЗАННОСТЬ (COUPLING)
- УСТОЙЧИВОСТЬ К ОШИБКАМ (FAULT TOLERANCE)
- СВЕЖЕСТЬ ДАННЫХ (DATA FRESHNESS)

НЕМНОГО ИСТОРИИ



1980 - 1990e

Первые messaging системы – **TIBCO**, **IBM MQ**.



1997e

MSMQ - messaging system от Microsoft



2000e

Стандартизация – **JMS**, **AMQP**, **MQTT**.



2000 - 2010

RabbitMQ, Apache QPID, Kafka, Azure Service Bus



NServiceBus
in Particular

2000e – наше время

NServiceBus



2007, 2015 – версия 3.0

MassTransit

MASSTRANSIT

LIGHTWEIGHT ESB

MassTransit / **MassTransit**

Watch

120

Star

834

Fork

544

Code

Issues 47

Pull requests 3

Wiki

Pulse

Graphs

Distributed Application Framework for .NET <http://masstransit-project.com/>

3,658 commits

10 branches

56 releases

88 contributors

Branch: develop


New pull request

Create new file

Upload files

Find file

Clone or download

 **phatboyg** Routing slip variables can be removed by setting to null, and keys ar... Latest commit 6b51f3e 39 minutes ago

doc

Update quickstart.rst (#622)27 days ago

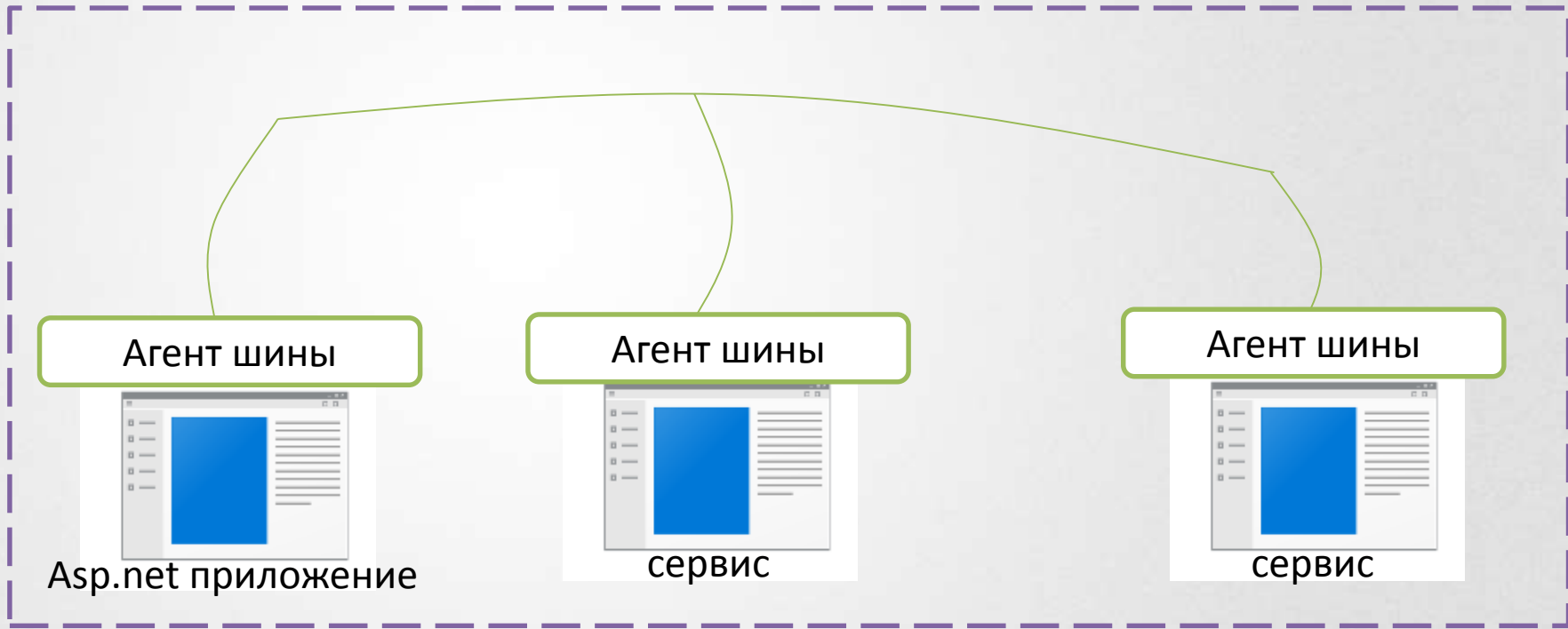
host

Enable logging output and set project startup when installing MassTra...7 months ago

src

Routing slip variables can be removed by setting to null, and keys ar...39 minutes ago

ШИНА



БРОКЕРЫ

MASSTRANSIT



RABBITMQ

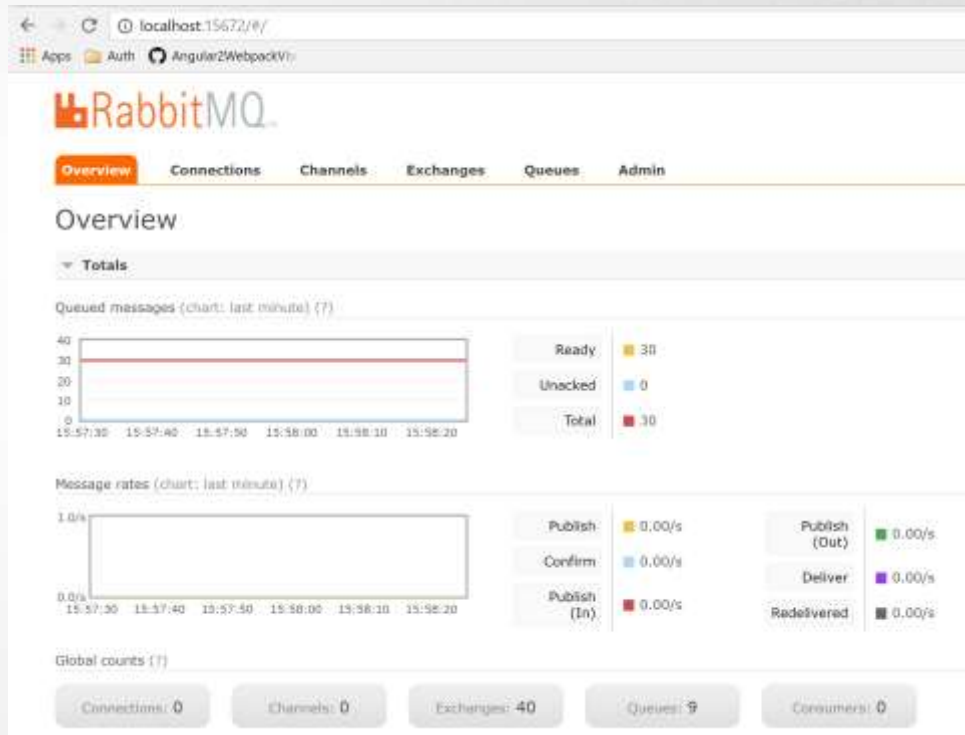
ERLANG



RABBITMQ

ERLANG

- RABBITMQ_MANAGEMENT
- LOCALHOST:15672
- AMQP 0-9-1



DEMO

КОМПОНЕНТЫ

СООБЩЕНИЯ (КОНТРАКТЫ) – КОМАНДЫ И СОБЫТИЯ

CreateCustomer, WorkItemCreated

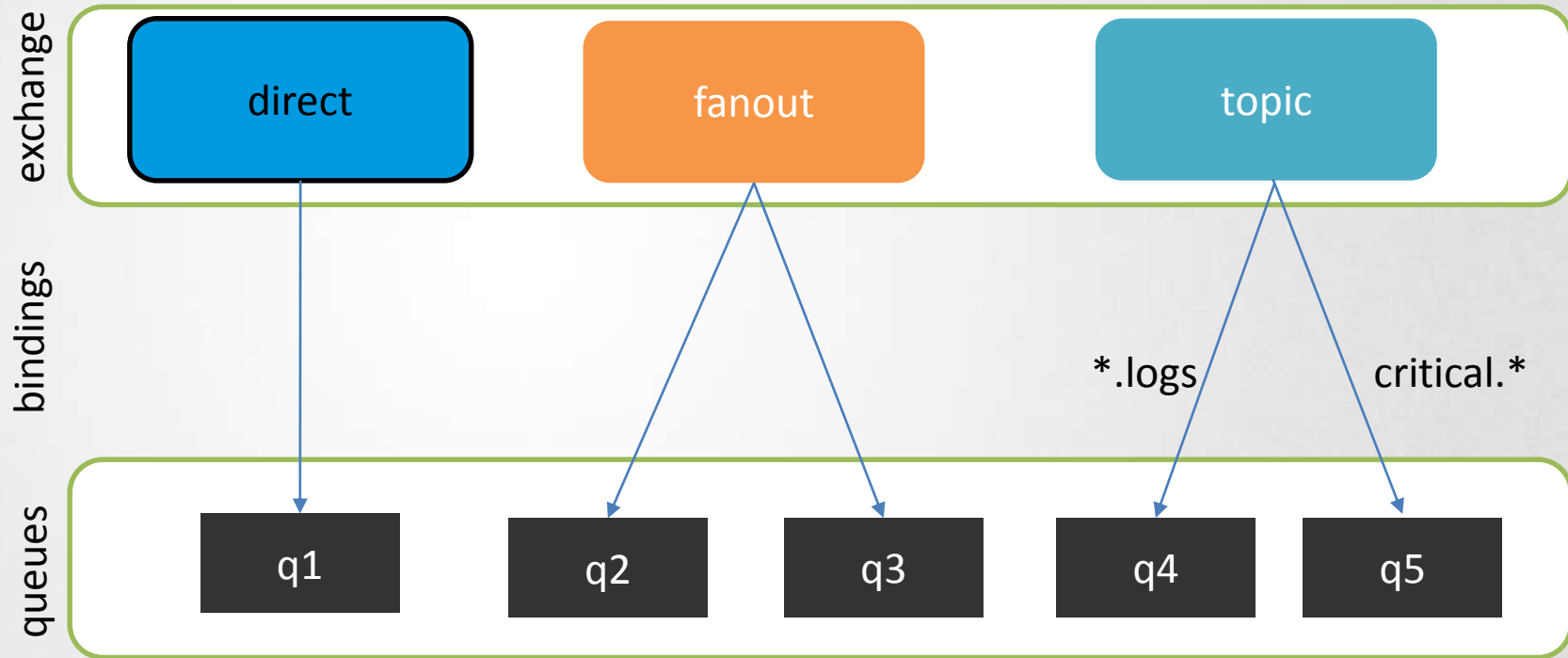
IBUS – ШИНА, СОЗДАЕТСЯ ПРИ СТАРТЕ

bus.Publish(...)

CONSUMERS – ПОЛУЧАЮТ И ОБРАБАТЫВАЮТ СООБЩЕНИЯ

IConsumer(ConsumeContext)

RABBITMQ



РАБОТА НАД ОШИБКАМИ

RETRY POLICIES

- None
- Immediate
- Intervals
- Exponential
- Incremental
- ...

РАБОТА НАД ОШИБКАМИ

RETRY POLICIES

```
Bus.Factory.CreateUsingInMemory(cfg =>
{
    cfg.ReceiveEndpoint("queue_name", ep =>
    {
        ep.Handler(async cxt => {});
        ep.Handler(async cxt => {}, endpointConfig =>
        {
            endpointConfig.Retry(Retry.None);
        });
    });
});
```

РАБОТА НАД ОШИБКАМИ

МОЖНО ПОДПИСЫВАТЬСЯ НА ОШИБКИ

```
public class FaultConsumer : IConsumer<Fault<UpdateCustomerAddress>>
{
    public async Task Consume(ConsumeContext<Fault<UpdateCustomerAddress>> context)
    {
        var originalMessage = context.Message.Message;
        var exceptions = context.Message.Exceptions;

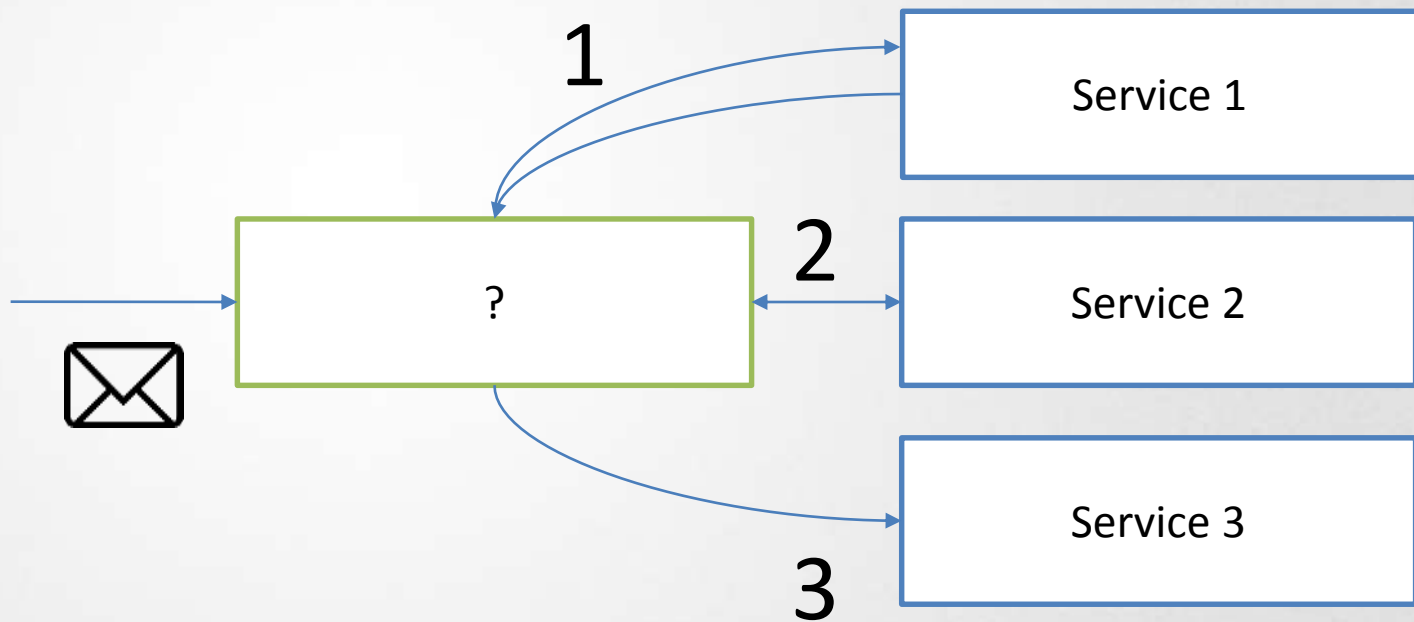
        //Do something interesting.
    }
}
```

РАБОТА НАД ОШИБКАМИ

CIRCUIT BREAKER



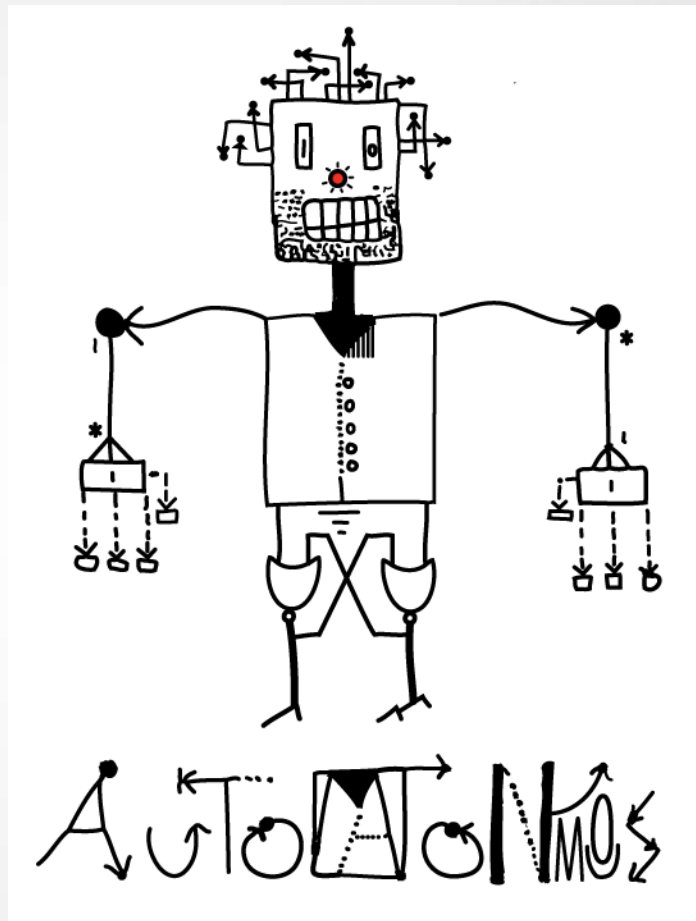
DEMO



- Saga – распределенный процесс с **общим координатором**
- Состоят из **действий, событий и состояния**
- Сохраняются в **репозитории**
- Описывается как **конечный автомат**

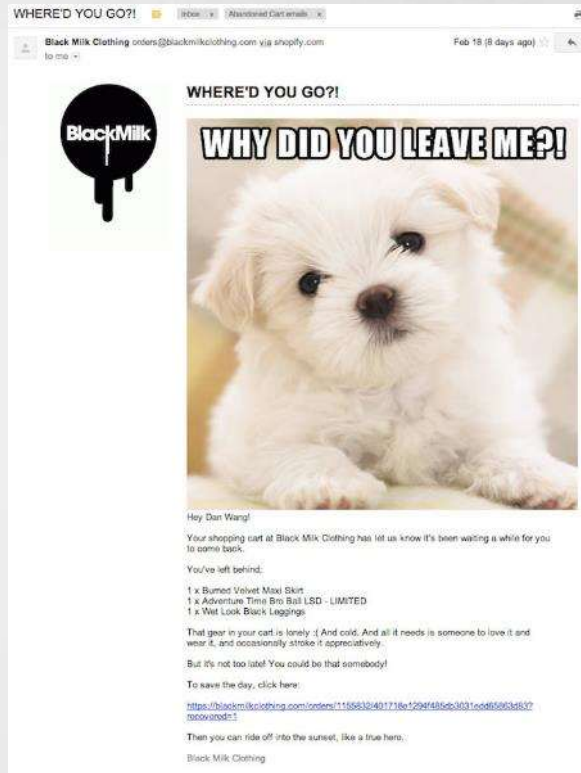
САГИ

LONG RUNNING PROCESS



САГИ

ПРИМЕР



DEMO

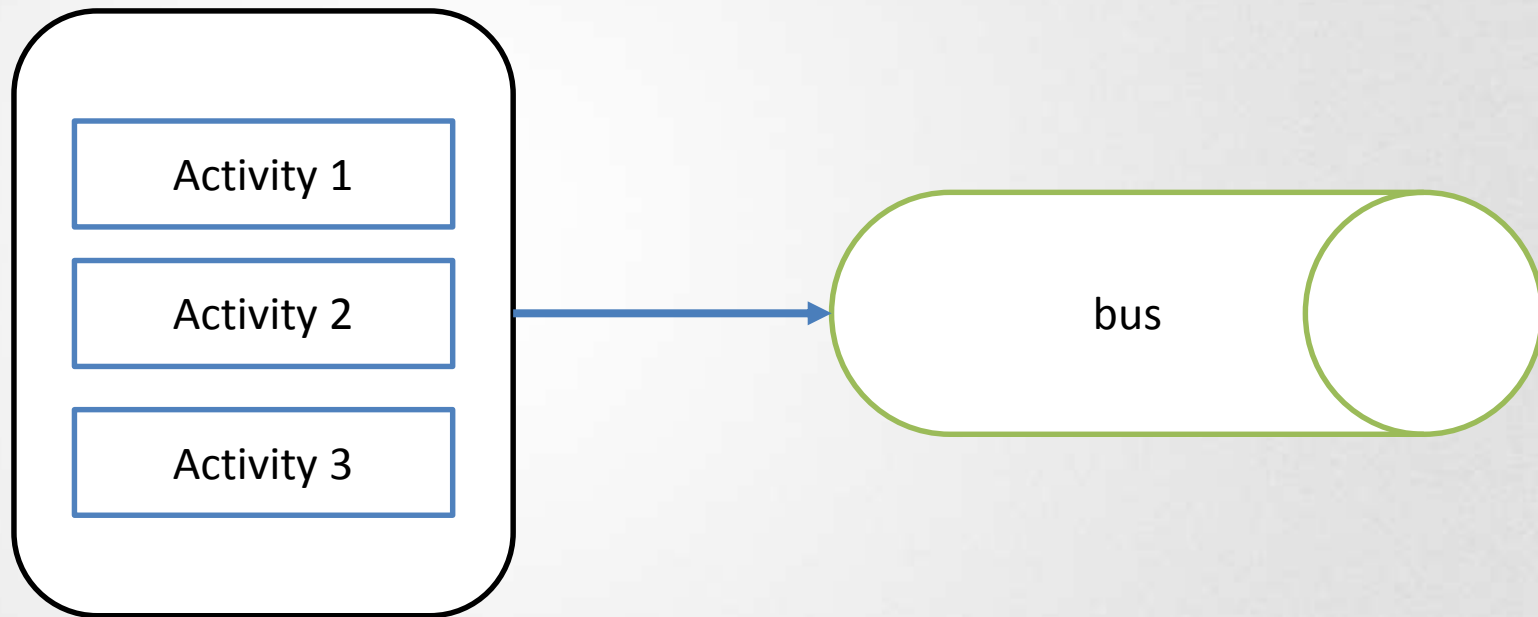
COURIER

ROUTING SLIP

- Карта маршрутизации (EIP)
- Позволяет динамически определять маршрут обработки сообщения
- Routing slip включает с себя маршрут (Itinerary). Itinerary - это список Activity
- Этот маршрут вместе с сообщением отправляется по шине
- Каждая activity имеет два метода: `execute`, `compensate`.

COURIER

ROUTING SLIP



DEMO

РАЗВЕРТЫВАНИЕ



TOPSHELF

```
HostFactory.Run(x =>
{
    x.Service<TownCrier>(s =>
    {
        s.ConstructUsing(name=> new TownCrier());
        s.WhenStarted(tc => tc.Start());
        s.WhenStopped(tc => tc.Stop());
    });
    x.RunAsLocalSystem();

    x.SetDescription("Sample Topshelf Host");
    x.SetDisplayName("Stuff");
    x.SetServiceName("Stuff");
});
```

ЧТО ЕЩЕ

1. Отложенная отправка (scheduling) через Quartz .Net
2. Сообщения как Streams через Reactive Extensions
3. Unit tests через встроенный framework для тестирования
4. Счетчики производительности
5. Turnouts – долгоживущие consumers
6. Request-reply – двухстороннее взаимодействие
7. Greenpipes – pipes & filters
8. Поддержка ioc контейнеров

В ЗАКЛЮЧЕНИИ

MESSAGING VS RPC

1. 😊 Расширяемость
2. 😊 Просто обеспечить надежность
3. 😊 Легко масштабировать
4. 😊😞 Другая модель взаимодействия
5. 😞 В RPC клиент может контролировать что ему нужно
6. 😞 RPC есть практически везде, messaging нет
7. 😞 Не знаем когда работает или нет
8. 😊 Можно комбинировать

ЧТО ПОЧИТАТЬ

1. Enterprise integration patterns (Addison-Wesley, 2004)
2. SOA Patterns (manning, 2012)
3. Building microservices (2015, O'Reilly Media)
4. DotNetRocks подкасты (<https://www.dotnetrocks.com>, 798, 1228, 1242, 1055)
5. <http://blog.phatboyg.com>, <https://lostechies.com/chrispatterson>
6. <https://github.com/MassTransit/MassTransit>
7. <https://github.com/jacobpovar/DotNetMsk>
8. Jacob.povar@gmail.com

**СПАСИБО ЗА
ВНИМАНИЕ**

ДО НОВЫХ ВСТРЕЧ