Очереди сообщений

Определение

Очередь сообщений – это некая структура данных, которая обеспечивает хранение и передачу двоичных данных между различными участниками системы.

Преимущества

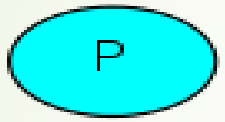
- Независимость компонентов системы друг от друга
- Экономия ресурсов
- Надежность
- Гарантия последовательной обработки

RabbitMQ

RabbitMQ – платформа, реализующая систему обмена сообщениями между компонентами программной системы на основе стандарта AMQP.

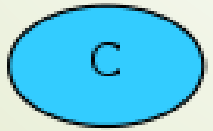
Терминология в RabbitMQ

- **Producer** (поставщик) – программа, отправляющая сообщения.

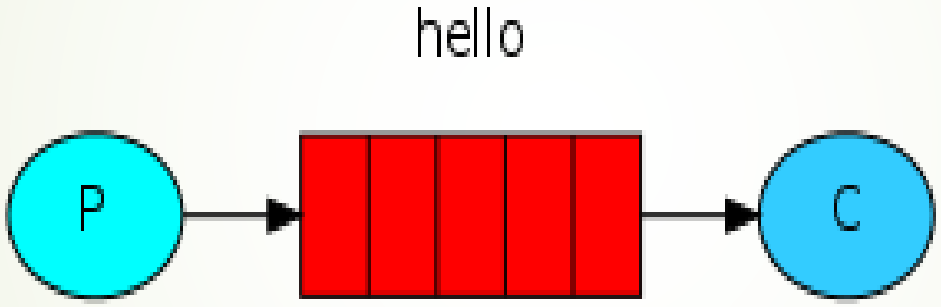


- **Queue** (очередь). Хотя сообщения проходят через RabbitMQ и приложения, хранятся они только в очередях.

- **Consumer** (подписчик) – программа, принимающая сообщения.

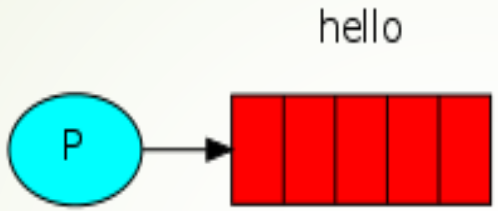


Hello, World!



Библиотека RabbitMQ

Отправка сообщений




```
class Send
{
    public static void Main()
    {
        var factory = new ConnectionFactory() { HostName = "localhost" };
        using (var connection = factory.CreateConnection())
        {
            using (var channel = connection.CreateModel())
            {
                ...
            }
        }
    }
}
```

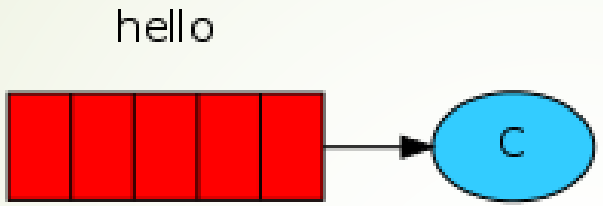
Отправка сообщений

```
class Send
{
    public static void Main()
    {
        var factory = new ConnectionFactory() { HostName = "localhost" };
        using (var connection = factory.CreateConnection())
        {
            using (var channel = connection.CreateModel())
            {
                channel.QueueDeclare("hello", false, false, false, null);

                string message = "Hello World!";
                var body = Encoding.UTF8.GetBytes(message);

                channel.BasicPublish("", "hello", null, body);
                Console.WriteLine(" [x] Sent {0}", message);
            }
        }
    }
}
```

Получение сообщений




```
class Receive
{
    public static void Main()
    {
        var factory = new ConnectionFactory() { HostName = "localhost" };
        using (var connection = factory.CreateConnection())
        {
            using (var channel = connection.CreateModel())
            {
                channel.QueueDeclare("hello", false, false, false, null);
                ...
            }
        }
    }
}
```

Получение сообщений

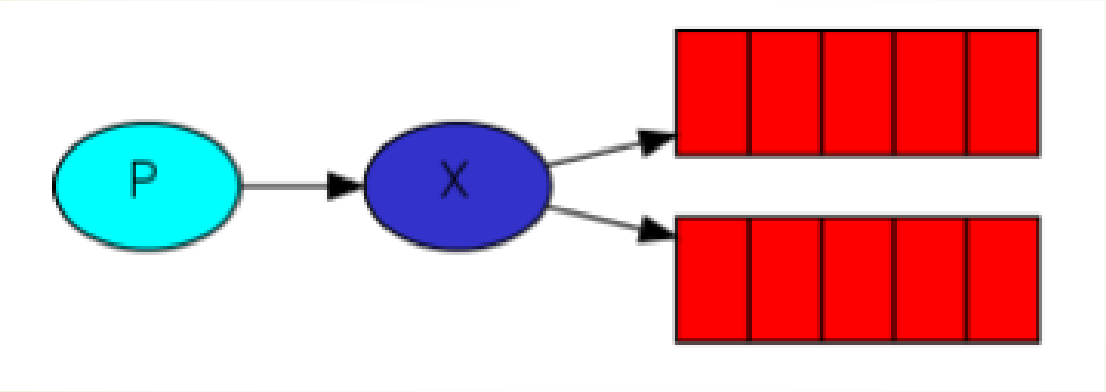
```
class Receive
{
    public static void Main()
    {
        var factory = new ConnectionFactory() { HostName = "localhost" };
        using (var connection = factory.CreateConnection())
        {
            using (var channel = connection.CreateModel())
            {
                channel.QueueDeclare("hello", false, false, false, null);

                var consumer = new QueueingBasicConsumer(channel);
                channel.BasicConsume("hello", true, consumer);

                Console.WriteLine(" [*] Waiting for messages." +
                                  "To exit press CTRL+C");
                while (true)
                {
                    var ea = (BasicDeliverEventArgs)consumer.Queue.Dequeue();

                    var body = ea.Body;
                    var message = Encoding.UTF8.GetString(body);
                    Console.WriteLine(" [x] Received {0}", message);
                }
            }
        }
    }
}
```

Обмен (exchange)




```
channel.ExchangeDeclare("logs", "fanout");
```

```
using System;
using RabbitMQ.Client;
using System.Text;

class EmitLog
{
    public static void Main(string[] args)
    {
        var factory = new ConnectionFactory() { HostName = "localhost" };
        using (var connection = factory.CreateConnection())
        using (var channel = connection.CreateModel())
        {
            channel.ExchangeDeclare("logs", "fanout");

            var message = GetMessage(args);
            var body = Encoding.UTF8.GetBytes(message);
            channel.BasicPublish("logs", "", null, body);
            Console.WriteLine(" [x] Sent {0}", message);
        }
    }

    private static string GetMessage(string[] args)
    {
      return ((args.Length > 0) ? string.Join(" ", args) : "info: Hello World!");
    }
}
```

```
class ReceiveLogs
{
    public static void Main()
    {
        var factory = new ConnectionFactory() { HostName = "localhost" };
        using (var connection = factory.CreateConnection())
        {
            using (var channel = connection.CreateModel())
            {
                channel.ExchangeDeclare("logs", "fanout");

                var queueName = channel.QueueDeclare();

                channel.QueueBind(queueName, "logs", "");
                var consumer = new QueueingBasicConsumer(channel);
                channel.BasicConsume(queueName, true, consumer);

                Console.WriteLine(" [*] Waiting for logs." +
                                  "To exit press CTRL+C");
                while (true)
                {
                    var ea = (BasicDeliverEventArgs)consumer.Queue.Dequeue();

                    var body = ea.Body;
                    var message = Encoding.UTF8.GetString(body);
                    Console.WriteLine(" [x] {0}", message);
                }
            }
        }
    }
}
```

Связывание