



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.

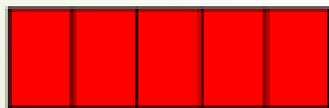
Терминология в RabbitMQ

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



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

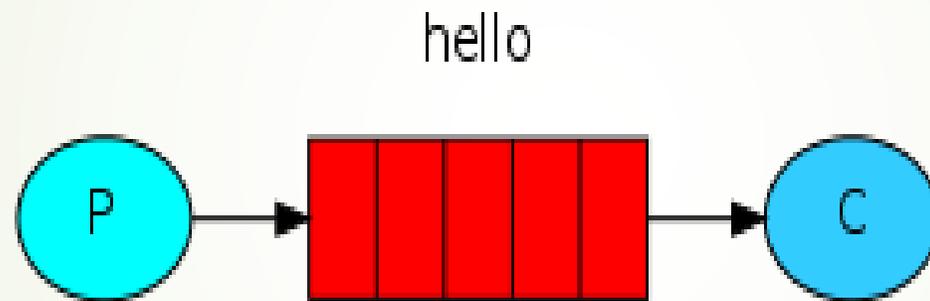
queue_name



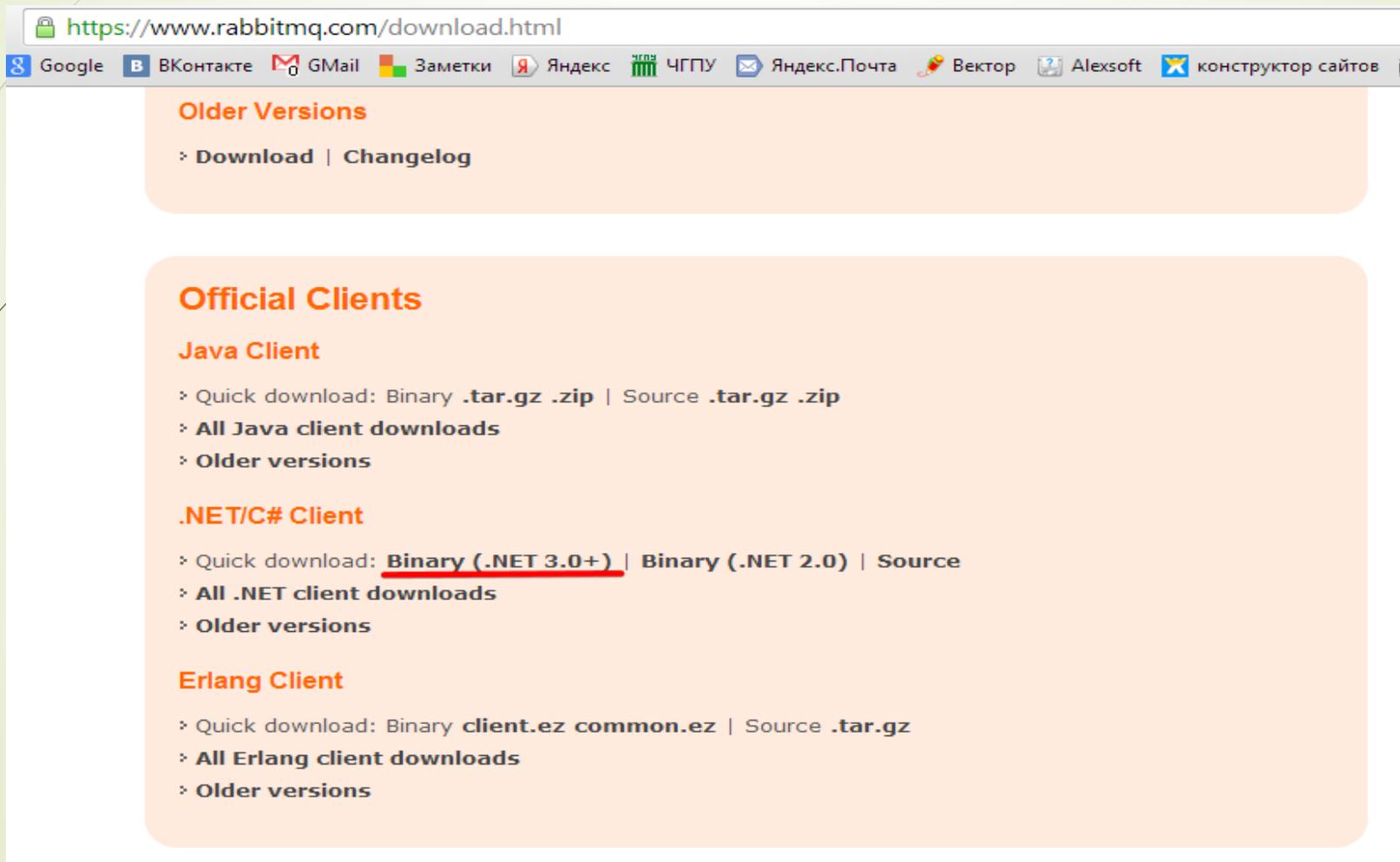
- ➔ **Consumer** (подписчик) – программа, принимающая сообщения.



Hello, World!



Библиотека RabbitMQ

A screenshot of a web browser displaying the RabbitMQ download page. The browser's address bar shows the URL 'https://www.rabbitmq.com/download.html'. The browser's toolbar includes icons for Google, ВКонтакте, GMail, Заметки, Яндекс, ЧГПУ, Яндекс.Почта, Вектор, Alexsoft, and конструктор сайтов. The page content is organized into sections with orange headers and light orange backgrounds. The first section is 'Older Versions' with links for 'Download' and 'Changelog'. The second section is 'Official Clients', which includes three sub-sections: 'Java Client' with links for 'Quick download: Binary .tar.gz .zip | Source .tar.gz .zip', 'All Java client downloads', and 'Older versions'; '.NET/C# Client' with links for 'Quick download: Binary (.NET 3.0+) | Binary (.NET 2.0) | Source', 'All .NET client downloads', and 'Older versions'; and 'Erlang Client' with links for 'Quick download: Binary client.ez common.ez | Source .tar.gz', 'All Erlang client downloads', and 'Older versions'.

<https://www.rabbitmq.com/download.html>

Google ВКонтакте GMail Заметки Яндекс ЧГПУ Яндекс.Почта Вектор Alexsoft конструктор сайтов

Older Versions

- › [Download](#) | [Changelog](#)

Official Clients

Java Client

- › Quick download: [Binary .tar.gz .zip](#) | [Source .tar.gz .zip](#)
- › [All Java client downloads](#)
- › [Older versions](#)

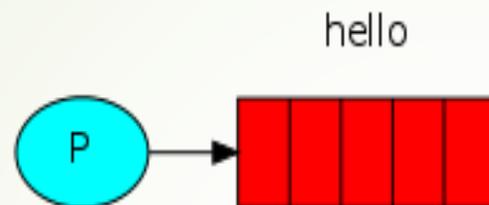
.NET/C# Client

- › Quick download: [Binary \(.NET 3.0+\)](#) | [Binary \(.NET 2.0\)](#) | [Source](#)
- › [All .NET client downloads](#)
- › [Older versions](#)

Erlang Client

- › Quick download: Binary [client.ez common.ez](#) | [Source .tar.gz](#)
- › [All Erlang client downloads](#)
- › [Older versions](#)

Отправка сообщений



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

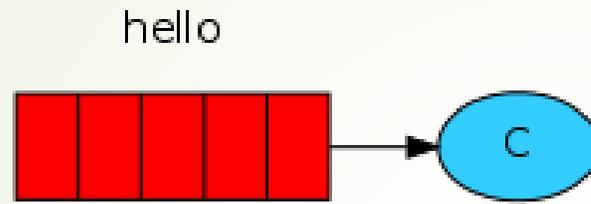
Отправка сообщений

```
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);

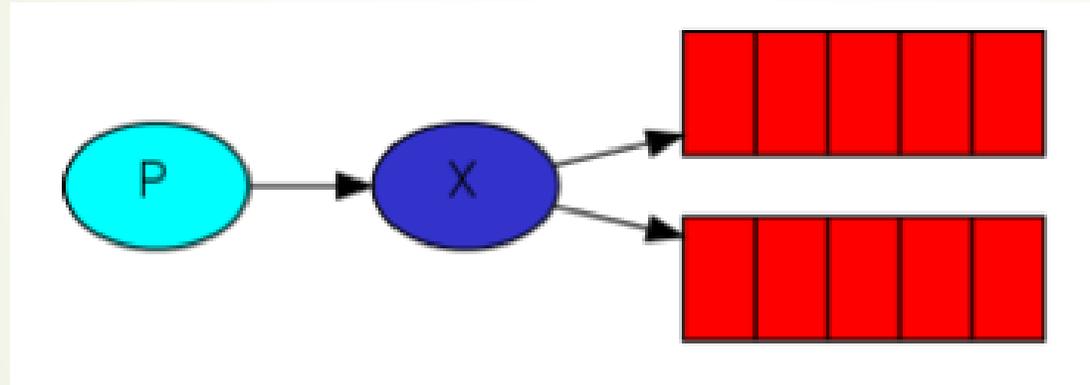
                var consumer = new QueueingBasicConsumer(channel);
                channel.BasicConsume("hello", true, consumer);

                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", ""); ← Связывание
                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);
                }
            }
        }
    }
}
```