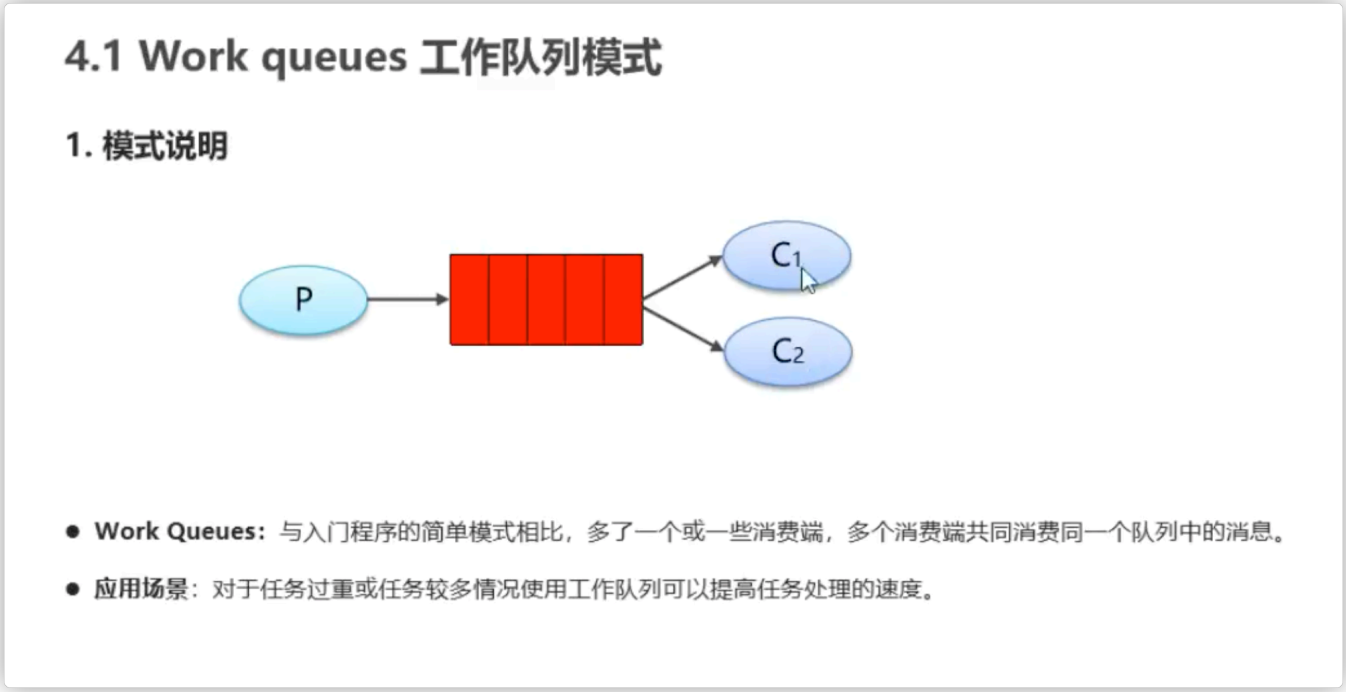
rabbitMq的工作模式

https://www.rabbitmq.com/getstarted.html

简单模式和work模式也有交换机，只不过用的是默认的



Producer_WorkQueues

```
package com.itheima.producer;

import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.ConnectionFactory;

import java.io.IOException;
import java.util.concurrent.TimeoutException;

/**
```

```
 * 发送消息
 */
public class Producer_WorkQueues {
    public static void main(String[] args) throws IOException, TimeoutException {

        //1.创建连接工厂
        ConnectionFactory factory = new ConnectionFactory();
        //2. 设置参数
        factory.setHost("172.16.98.133");//ip  默认值 localhost
        factory.setPort(5672); //端口  默认值 5672
        factory.setVirtualHost("/itcast");//虚拟机 默认值/
        factory.setUsername("heima");//用户名 默认 guest
        factory.setPassword("heima");//密码 默认值 guest
        //3. 创建连接 Connection
        Connection connection = factory.newConnection();
        //4. 创建Channel
        Channel channel = connection.createChannel();
        //5. 创建队列Queue
        /*
        queueDeclare(String queue, boolean durable, boolean exclusive, boolean autoDelete,
Map<String, Object> arguments)
        参数:
            1. queue:队列名称
            2. durable:是否持久化，当mq重启之后，还在
            3. exclusive:
                * 是否独占。只能有一个消费者监听这队列
                * 当Connection关闭时，是否删除队列
                *
            4. autoDelete:是否自动删除。当没有Consumer时，自动删除掉
            5. arguments:参数。

         */
        //如果没有一个名字叫hello_world的队列，则会创建该队列，如果有则不会创建
        channel.queueDeclare("work_queues",true,false,false,null);
        /*
        basicPublish(String exchange, String routingKey, BasicProperties props, byte[] body)
        参数:
            1. exchange:交换机名称。简单模式下交换机会使用默认的 ""
            2. routingKey:路由名称
            3. props:配置信息
            4. body:发送消息数据

         */
        for (int i = 1; i <= 10; i++) {
            String body = i+"hello rabbitmq~~~";

            //6. 发送消息
            channel.basicPublish("","work_queues",null,body.getBytes());
        }
```

```
        //7.释放资源
        channel.close();
        connection.close();

    }
}
```

Consumer_WorkQueues1

```
package com.itheima.producer;

import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.ConnectionFactory;

import java.io.IOException;
import java.util.concurrent.TimeoutException;

/**
 * 发送消息
 */
public class Producer_WorkQueues {
    public static void main(String[] args) throws IOException, TimeoutException {

        //1.创建连接工厂
        ConnectionFactory factory = new ConnectionFactory();
        //2. 设置参数
        factory.setHost("172.16.98.133");//ip  默认值 localhost
        factory.setPort(5672); //端口  默认值 5672
        factory.setVirtualHost("/itcast");//虚拟机 默认值/
        factory.setUsername("heima");//用户名 默认 guest
        factory.setPassword("heima");//密码 默认值 guest
        //3. 创建连接 Connection
        Connection connection = factory.newConnection();
        //4. 创建Channel
        Channel channel = connection.createChannel();
        //5. 创建队列Queue
        /*
        queueDeclare(String queue, boolean durable, boolean exclusive, boolean autoDelete,
Map<String, Object> arguments)
        参数:
            1. queue：队列名称
            2. durable:是否持久化，当mq重启之后，还在
            3. exclusive:
                * 是否独占。只能有一个消费者监听这队列
                * 当Connection关闭时，是否删除队列
                *
            4. autoDelete:是否自动删除。当没有Consumer时，自动删除掉
            5. arguments：参数。
```

```
         */
        //如果没有一个名字叫hello_world的队列，则会创建该队列，如果有则不会创建
        channel.queueDeclare("work_queues",true,false,false,null);
        /*
        basicPublish(String exchange, String routingKey, BasicProperties props, byte[] body)
        参数：
            1. exchange：交换机名称。简单模式下交换机会使用默认的 ""
            2. routingKey：路由名称
            3. props：配置信息
            4. body：发送消息数据

         */
        for (int i = 1; i <= 10; i++) {
            String body = i+"hello rabbitmq~~~";

            //6. 发送消息
            channel.basicPublish("","work_queues",null,body.getBytes());
        }



        //7.释放资源
      channel.close();
      connection.close();

    }
}
```

Consumer_WorkQueues1

```
package com.itheima.consumer;

import com.rabbitmq.client.*;

import java.io.IOException;
import java.util.concurrent.TimeoutException;

public class Consumer_WorkQueues1 {
    public static void main(String[] args) throws IOException, TimeoutException {

        //1.创建连接工厂
        ConnectionFactory factory = new ConnectionFactory();
        //2. 设置参数
        factory.setHost("172.16.98.133");//ip  默认值 localhost
        factory.setPort(5672); //端口  默认值 5672
        factory.setVirtualHost("/itcast");//虚拟机 默认值/
        factory.setUsername("heima");//用户名 默认 guest
        factory.setPassword("heima");//密码 默认值 guest
        //3. 创建连接 Connection
        Connection connection = factory.newConnection();
        //4. 创建Channel
```

```
        Channel channel = connection.createChannel();
        //5. 创建队列Queue
        /*
        queueDeclare(String queue, boolean durable, boolean exclusive, boolean autoDelete,
Map<String, Object> arguments)
        参数:
            1. queue:队列名称
            2. durable:是否持久化，当mq重启之后，还在
            3. exclusive:
                * 是否独占。只能有一个消费者监听这队列
                * 当Connection关闭时，是否删除队列
                *
            4. autoDelete:是否自动删除。当没有Consumer时，自动删除掉
            5. arguments:参数。

         */
        //如果没有一个名字叫hello_world的队列，则会创建该队列，如果有则不会创建
        channel.queueDeclare("work_queues",true,false,false,null);

        /*
        basicConsume(String queue, boolean autoAck, Consumer callback)
        参数:
            1. queue:队列名称
            2. autoAck:是否自动确认
            3. callback:回调对象

         */
        // 接收消息
        Consumer consumer = new DefaultConsumer(channel){
            /*
                回调方法，当收到消息后，会自动执行该方法

                1. consumerTag:标识
                2. envelope:获取一些信息，交换机，路由key...
                3. properties:配置信息
                4. body:数据

             */
            @Override
            public void handleDelivery(String consumerTag, Envelope envelope,
AMQP.BasicProperties properties, byte[] body) throws IOException {
              /*  System.out.println("consumerTag:"+consumerTag);
                System.out.println("Exchange:"+envelope.getExchange());
                System.out.println("RoutingKey:"+envelope.getRoutingKey());
                System.out.println("properties:"+properties);*/
                System.out.println("body:"+new String(body));
            }
        };
        channel.basicConsume("work_queues",true,consumer);


        //关闭资源? 不要
```

```
    }
}
```

Consumer_WorkQueues2

```
package com.itheima.consumer;

import com.rabbitmq.client.*;

import java.io.IOException;
import java.util.concurrent.TimeoutException;

public class Consumer_WorkQueues2 {
    public static void main(String[] args) throws IOException, TimeoutException {

        //1.创建连接工厂
        ConnectionFactory factory = new ConnectionFactory();
        //2. 设置参数
        factory.setHost("172.16.98.133");//ip  默认值 localhost
        factory.setPort(5672); //端口  默认值 5672
        factory.setVirtualHost("/itcast");//虚拟机 默认值/
        factory.setUsername("heima");//用户名 默认 guest
        factory.setPassword("heima");//密码 默认值 guest
        //3. 创建连接 Connection
        Connection connection = factory.newConnection();
        //4. 创建Channel
        Channel channel = connection.createChannel();
        //5. 创建队列Queue
        /*
        queueDeclare(String queue, boolean durable, boolean exclusive, boolean autoDelete,
Map<String, Object> arguments)
        参数:
            1. queue：队列名称
            2. durable:是否持久化，当mq重启之后，还在
            3. exclusive:
                * 是否独占。只能有一个消费者监听这队列
                * 当Connection关闭时，是否删除队列
                *
            4. autoDelete:是否自动删除。当没有Consumer时，自动删除掉
            5. arguments：参数。

         */
        //如果没有一个名字叫hello_world的队列，则会创建该队列，如果有则不会创建
        channel.queueDeclare("work_queues",true,false,false,null);

        /*
        basicConsume(String queue, boolean autoAck, Consumer callback)
        参数:
            1. queue：队列名称
```

```
            2. autoAck：是否自动确认
            3. callback：回调对象

         */
        // 接收消息
        Consumer consumer = new DefaultConsumer(channel){
            /*
                回调方法，当收到消息后，会自动执行该方法

                1. consumerTag：标识
                2. envelope：获取一些信息，交换机，路由key...
                3. properties:配置信息
                4. body：数据

             */
            @Override
            public void handleDelivery(String consumerTag, Envelope envelope,
AMQP.BasicProperties properties, byte[] body) throws IOException {
              /*  System.out.println("consumerTag："+consumerTag);
                System.out.println("Exchange："+envelope.getExchange());
                System.out.println("RoutingKey："+envelope.getRoutingKey());
                System.out.println("properties："+properties);*/
                System.out.println("body："+new String(body));
            }
        };
        channel.basicConsume("work_queues",true,consumer);


        //关闭资源？不要

    }
}
```