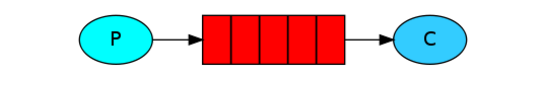
# Java操作队列

## 简单队列

功能：一个生产者P发送消息到队列Q,一个消费者C接收



P表示为生产者 、C表示为消费者 红色表示队列。

### Maven

|  |
| --- |
| <dependencies>  <dependency>  <groupId>com.rabbitmq</groupId>  <artifactId>amqp-client</artifactId>  <version>3.6.5</version>  </dependency>  </dependencies> |

### 封装Connection

|  |
| --- |
| **public** **class** MQConnectionUtils {  **public** **static** Connection newConnection() **throws** IOException, TimeoutException {  // 1.定义连接工厂  ConnectionFactory factory = **new** ConnectionFactory();  // 2.设置服务器地址  factory.setHost("127.0.0.1");  // 3.设置协议端口号  factory.setPort(5672);  // 4.设置vhost  factory.setVirtualHost("/test001\_host");  // 5.设置用户名称  factory.setUsername("test001");  // 6.设置用户密码  factory.setPassword("123456");  // 7.创建新的连接  Connection newConnection = factory.newConnection();  **return** newConnection;  }  } |

### 生产者

|  |
| --- |
| **public** **class** Producer {  **private** **static** **final** String ***QUEUE\_NAME*** = "test\_queue";  **public** **static** **void** main(String[] args) **throws** IOException, TimeoutException {  // 1.获取连接  Connection newConnection = MQConnectionUtils.*newConnection*();  // 2.创建通道  Channel channel = newConnection.createChannel();  // 3.创建队列声明  channel.queueDeclare(***QUEUE\_NAME***, **false**, **false**, **false**, **null**);  String msg = "test\_yushengjun110";  System.***out***.println("生产者发送消息:" + msg);  // 4.发送消息  channel.basicPublish("", ***QUEUE\_NAME***, **null**, msg.getBytes());  channel.close();  newConnection.close();  }  } |

### 消费者

|  |
| --- |
| **public** **class** Customer {  **private** **static** **final** String ***QUEUE\_NAME*** = "test\_queue";  **public** **static** **void** main(String[] args) **throws** IOException, TimeoutException {  System.***out***.println("002");  // 1.获取连接  Connection newConnection = MQConnectionUtils.*newConnection*();  // 2.获取通道  Channel channel = newConnection.createChannel();  channel.queueDeclare(***QUEUE\_NAME***, **false**, **false**, **false**, **null**);  DefaultConsumer defaultConsumer = **new** DefaultConsumer(channel) {  @Override  **public** **void** handleDelivery(String consumerTag, Envelope envelope, BasicProperties properties, **byte**[] body)  **throws** IOException {  String msgString = **new** String(body, "UTF-8");  System.***out***.println("消费者获取消息:" + msgString);  }  };  // 3.监听队列  channel.basicConsume(***QUEUE\_NAME***, **true**, defaultConsumer);  }  } |

## 工作队列

### 生产者

|  |
| --- |
| **public** **class** Producer {  **private** **static** **final** String ***QUEUE\_NAME*** = "test\_queue";  **public** **static** **void** main(String[] args) **throws** IOException, TimeoutException {  // 1.获取连接  Connection newConnection = MQConnectionUtils.*newConnection*();  // 2.创建通道  Channel channel = newConnection.createChannel();  // 3.创建队列声明  channel.queueDeclare(***QUEUE\_NAME***, **false**, **false**, **false**, **null**);  channel.basicQos(1);// 保证一次只分发一次 限制发送给同一个消费者 不得超过一条消息  **for** (**int** i = 1; i <= 50; i++) {  String msg = "test\_yushengjun" + i;  System.***out***.println("生产者发送消息:" + msg);  // 4.发送消息  channel.basicPublish("", ***QUEUE\_NAME***, **null**, msg.getBytes());  }  channel.close();  newConnection.close();  }  } |

### 消费者

|  |
| --- |
| **public** **class** Customer1 {  **private** **static** **final** String ***QUEUE\_NAME*** = "test\_queue";  **public** **static** **void** main(String[] args) **throws** IOException, TimeoutException {  System.***out***.println("001");  // 1.获取连接  Connection newConnection = MQConnectionUtils.*newConnection*();  // 2.获取通道  **final** Channel channel = newConnection.createChannel();  channel.queueDeclare(***QUEUE\_NAME***, **false**, **false**, **false**, **null**);  channel.basicQos(1);// 保证一次只分发一次 限制发送给同一个消费者 不得超过一条消息  DefaultConsumer defaultConsumer = **new** DefaultConsumer(channel) {  @Override  **public** **void** handleDelivery(String consumerTag, Envelope envelope, BasicProperties properties, **byte**[] body)  **throws** IOException {  String msgString = **new** String(body, "UTF-8");  System.***out***.println("消费者获取消息:" + msgString);  **try** {  Thread.*sleep*(1000);  } **catch** (Exception e) {  } **finally** {  // 手动回执消息  channel.basicAck(envelope.getDeliveryTag(), **false**);  }  }  };  // 3.监听队列  channel.basicConsume(***QUEUE\_NAME***, **false**, defaultConsumer);  }  } |