

Message Queue

Marina Gargiulo & Sascha M. Schumacher

Glossar

Grundlagen

Message Queue	☒
Message Oriented Middleware	☒
Messaging Middleware Systems	☒
Standard Open Source Protokolle	☒
Semantik des Message Passing	☒

RabbitMQ

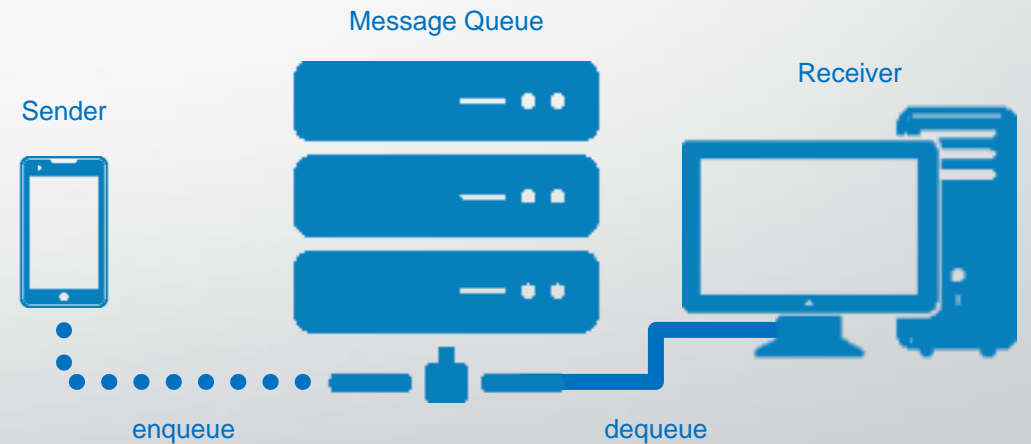
RabbitMQ	☒
Arten der RabbitMQ	☒
Allgemeines	2
Hello World: Sending	4
Hello World: Receiving	3

Grundlagen



Message Queue

- Zwischenspeicher für Objekte
- First In - First Out Prinzip
- Enqueue zum Einreihen von Objekten
- Dequeue zum Auslesen von Objekten



Message Oriented Middleware

- Eliminierung der Probleme des Remote Procedure Call
- Store and Forward Prinzip
- Asynchron, Verbindungslos, Anwendungsunabhängig
- 3 Komponenten: Konsumenten, Produzenten und Message Brokers

Messaging Middleware Systems



RabbitMQ



Oracle Advanced Queuing



Microsoft Message Queuing



WebSphere MQ

Standard Open Source Protokolle



AMQP

- **Advanced Message Queuing Protocol**
- Text Messages
- Map Messages
- Bytes Messages
- Stream Messages
- Object Messages



STOMP

- **Streaming Text Oriented Messaging Protocol**
- Text Messages

Semantik des Message Passing



RabbitMQ

RabbitMQ


Stabiler Datentransfer

Auf allen gängigen Betriebssystemen

Unterstützt auf

- Entwicklerplattformen
- Open Source und kommerziell

Arten der RabbitMQ

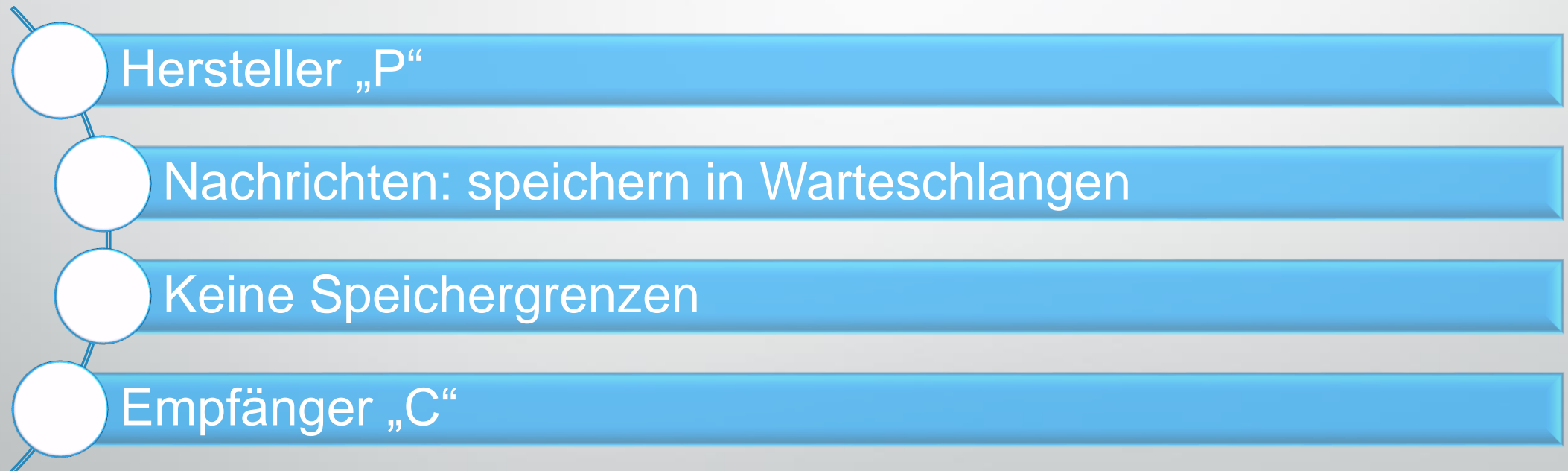
- 
- Hello World
 - Work queues

- Publish/Subscribe
- Routing

- Topics
- RPC

Allgemeines

- Ein Nachrichtenverteiler
- Akzeptiert Nachrichten vom Hersteller
- Leitet, puffert und enthält Nachrichten
- Benutzt eine Fachsprache



Hello World

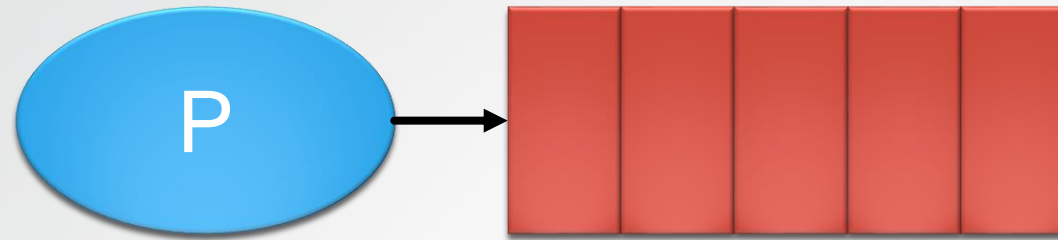


Sending

```
import com.rabbitmq.client.ConnectionFactory;
```

```
import com.rabbitmq.client.Connection;
```

```
import com.rabbitmq.client.Channel;
```



```
public class Send {  
    private final static String QUEUE_NAME = "hello";  
    public static void main(String[] argv) throws java.io.IOException { ... }  
}
```

```
ConnectionFactory factory = new ConnectionFactory();  
factory.setHost("localhost");  
Connection connection = factory.newConnection();  
Channel channel = connection.createChannel();
```



```
channel.queueDeclare(QUEUE_NAME, false, false, false, null);  
String message = "Hello World!";  
channel.basicPublish("", QUEUE_NAME, null, message.getBytes());  
System.out.println(" [x] Sent '" + message + "'");
```

```
channel.close ();  
connection.close ();
```

Receiving

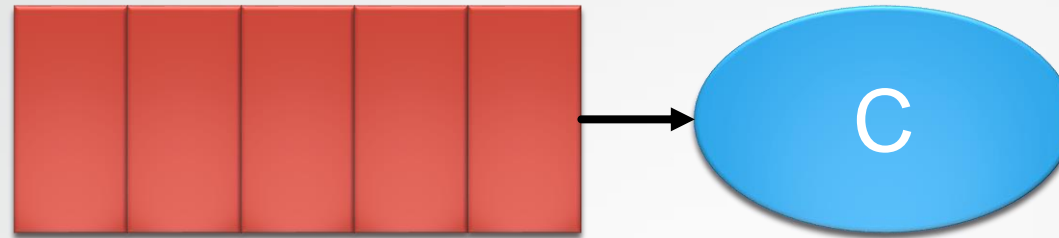
```
import com.rabbitmq.client.ConnectionFactory;
```

```
import com.rabbitmq.client.Connection;
```

```
import com.rabbitmq.client.Channel;
```

```
import com.rabbitmq.client.Consumer;
```

```
import com.rabbitmq.client.DefaultConsumer;
```



```
public class Recv {  
    private final static String QUEUE_NAME = "hello";  
    public static void main(String[] argv) throws java.io.IOException, java.lang.InterruptedException {  
        ConnectionFactory factory = new ConnectionFactory();  
        factory.setHost("localhost");  
        Connection connection = factory.newConnection();  
        Channel channel = connection.createChannel();  
        channel.queueDeclare(QUEUE_NAME, false, false, false, null);  
        System.out.println(" [*] Waiting for messages. To exit press CTRL+C");  
        ...  
    }  
}
```

```
Consumer consumer = new DefaultConsumer(channel) {  
    @Override  
    public void handleDelivery(String consumerTag, Envelope envelope,  
        AMQP.BasicProperties properties, byte[] body) throws IOException {  
        String message = new String(body, "UTF-8");  
        System.out.println(" [x] Received '" + message + "'");  
    }  
};  
channel.basicConsume(QUEUE_NAME, true, consumer);
```





Vielen Dank für Ihre
Aufmerksamkeit!

Haben Sie noch Fragen?