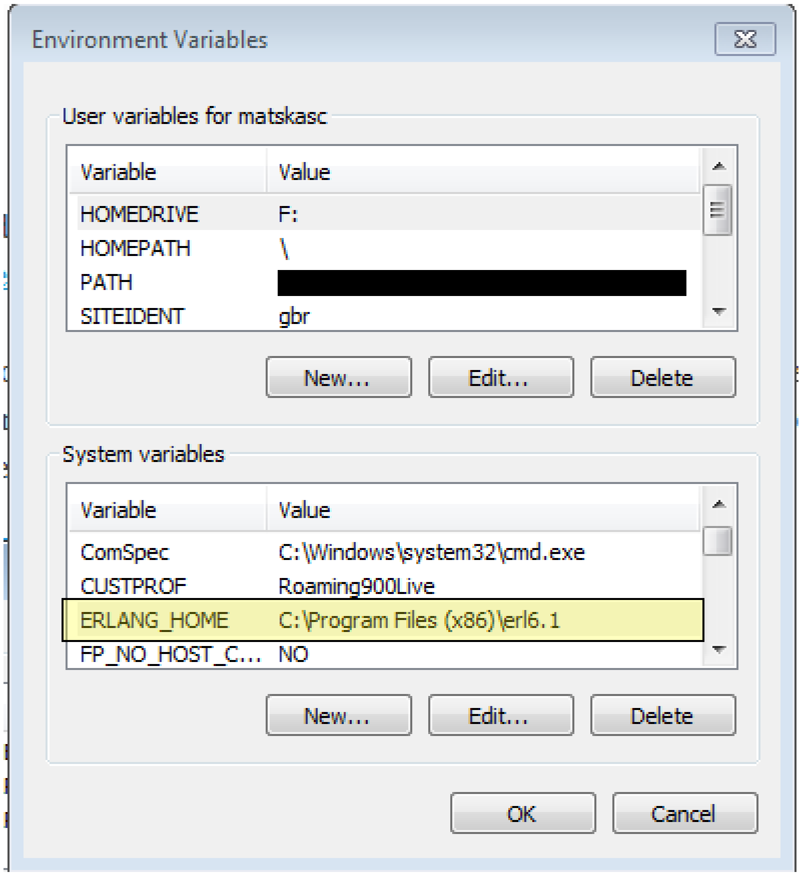
Set up RabbitMQ

1. Download Erlang https://www.erlang.org/downloads
   Set up **ERLANG_HOME**



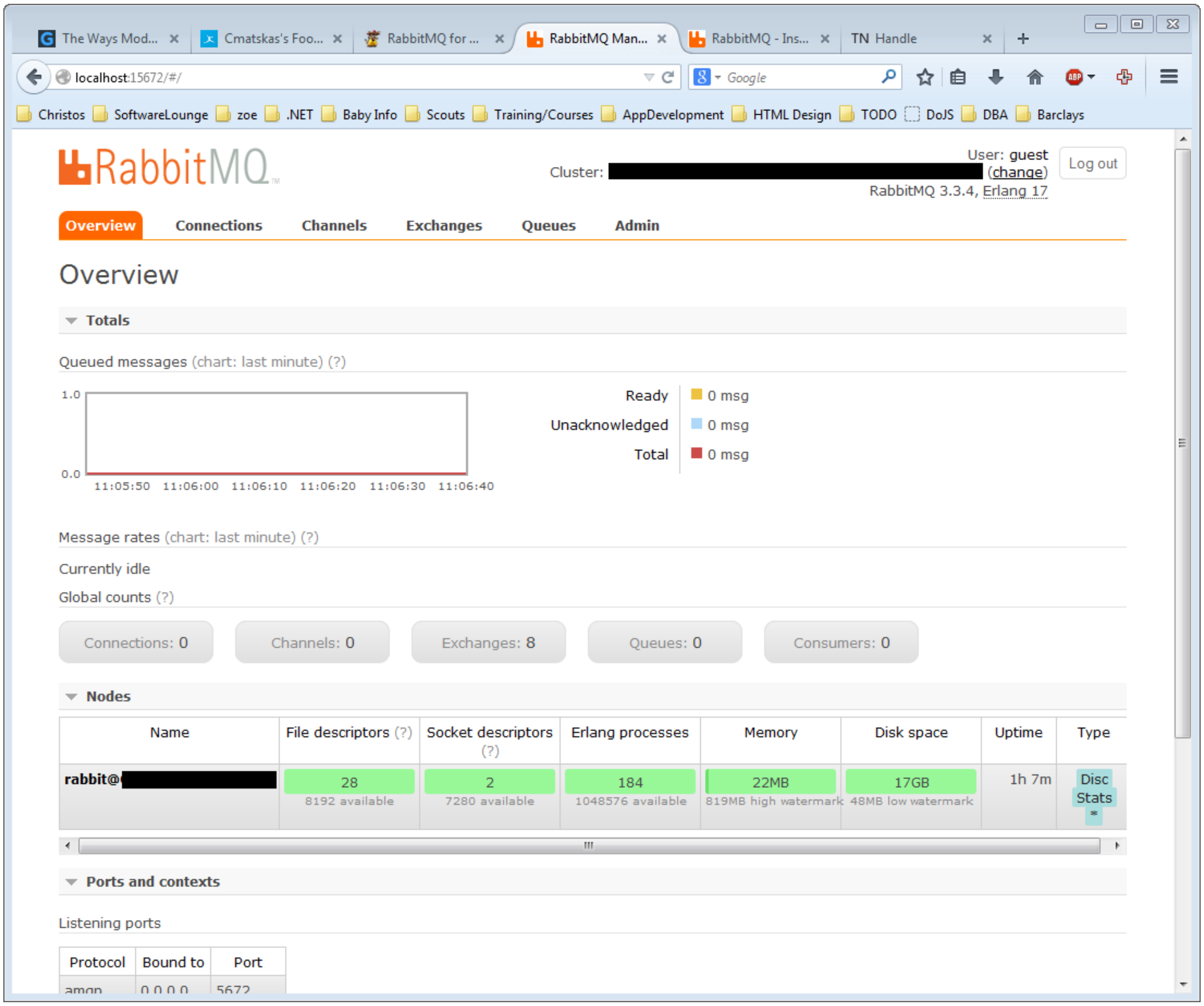
Variable	Value
HOMEDRIVE	F:
HOMEPATH	\
PATH	[redacted]
SITEIDENT	gbr

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
CUSTPROF	Roaming900Live
ERLANG_HOME	C:\Program Files (x86)\erl6.1
FP_NO_HOST_C...	NO

2. Download RabbitMQ https://www.rabbitmq.com/install-windows.html

3. RabbitMQ Web management
   1) Open an command line (Run as Administrator)
   2) Navigate to the **sbin** directory of the RabbitMQ Server installation directory. In my case the path is `C:\Program Files (x86)\RabbitMQ Server\rabbitmq_server-3.3.4\sbin`
   3) Run the following command to enable the plugin `rabbitmq-plugins.bat enable rabbitmq_management`

4. To check if everything worked as expected, navigate to http://localhost:15672 . You will be prompted for username and password. The default credentials are:
   Username: **guest**
   Password: **guest**



User: guest (change)

Cluster: [redacted]

RabbitMQ 3.3.4, Erlang 17

Overview

Totals

Queued messages (chart: last minute) (?)

Ready	🟨 0 msg
Unacknowledged	🟦 0 msg
Total	🟥 0 msg

Message rates (chart: last minute) (?)

Currently idle

Global counts (?)

Connections: 0 Channels: 0 Exchanges: 8 Queues: 0 Consumers: 0

Nodes

Name	File descriptors (?)	Socket descriptors (?)	Erlang processes	Memory	Disk space	Uptime	Type
rabbit@[redacted]	28 8192 available	2 7280 available	184 1048576 available	22MB 819MB high watermark	17GB 48MB low watermark	1h 7m	Disc Stats *

Ports and contexts

Listening ports

Protocol	Bound to	Port
amqp	0.0.0.0	5672

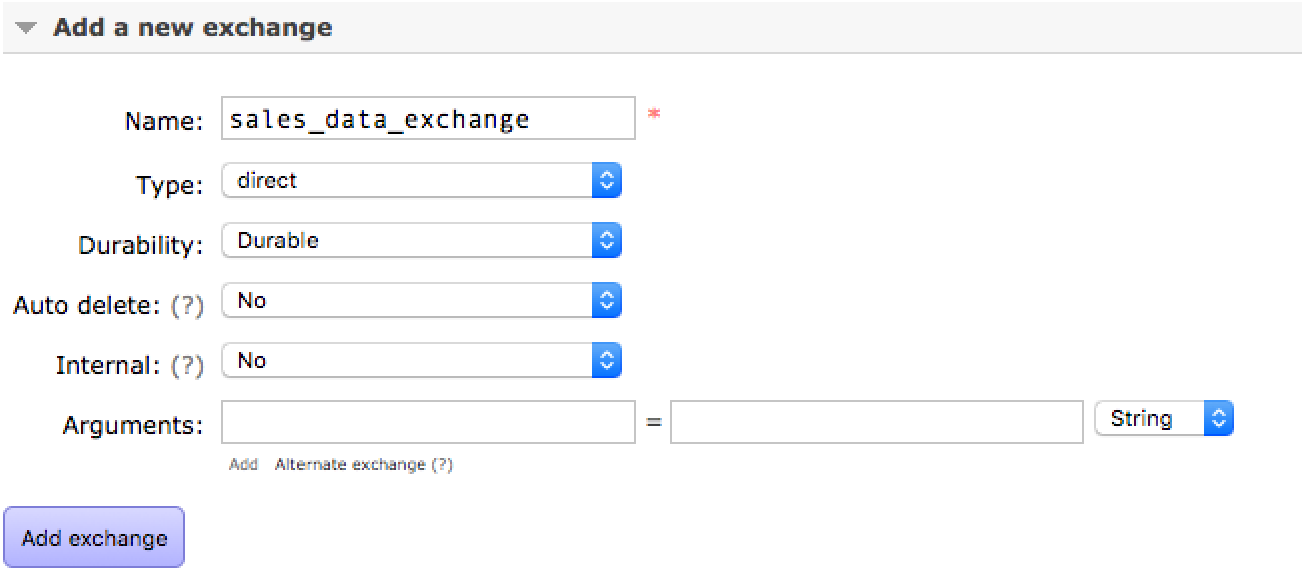
5. Direct Exchange demo

In RabbitMQ server, we need to create a direct exchange and name it as “**sales_data_exchange**“. After that we should create two queues and need to bind those queues with direct exchange using two binding keys (**customer.order** and **customer.deliveries**).

queue name	binding key
orders_queue	customer.order
deliveries_queue	customer.deliveries

1. Creating the direct exchange

Click on the **exchange** tab and add your message exchange. once it is added successfully, it will be listed under **exchanges** with given name.



Add a new exchange

Name: sales_data_exchange *

Type: direct

Durability: Durable

Auto delete: (?) No

Internal: (?) No

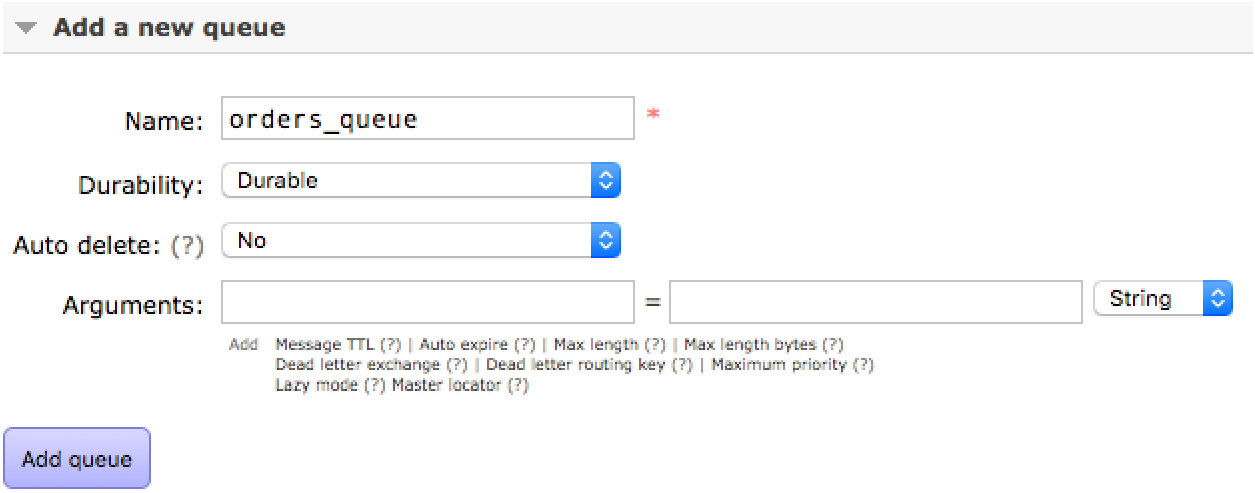
Arguments: = String

You can notice that the type is **direct**. Therefore this is a direct exchange.

2. Creating two queues (known as **orders_queue** and **deliveries_queue**)

Click on the **Queues** tab and add the queue. once it is added successfully, it will be listed under **All Queues**.

- **adding the orders_queue**



Add a new queue

Name: orders_queue *

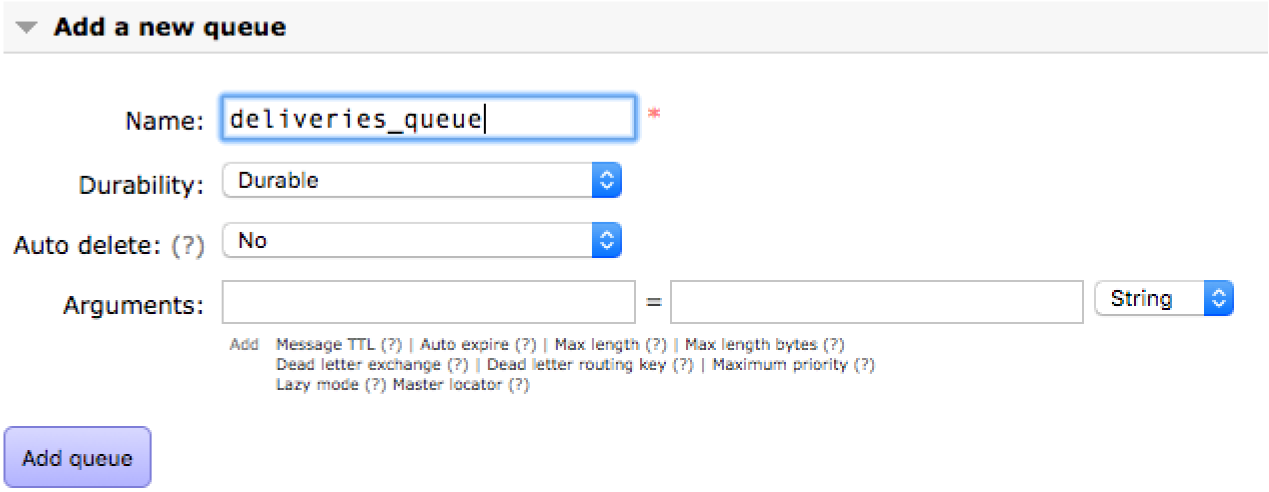
Durability: Durable

Auto delete: (?) No

Arguments: = String

Add Message TTL (?) | Auto expire (?) | Max length (?) | Max length bytes (?)
Dead letter exchange (?) | Dead letter routing key (?) | Maximum priority (?)
Lazy mode (?) Master locator (?)
Add queue

- **adding the deliveries_queue**



Add a new queue

Name: deliveries_queue *

Durability: Durable

Auto delete: (?) No

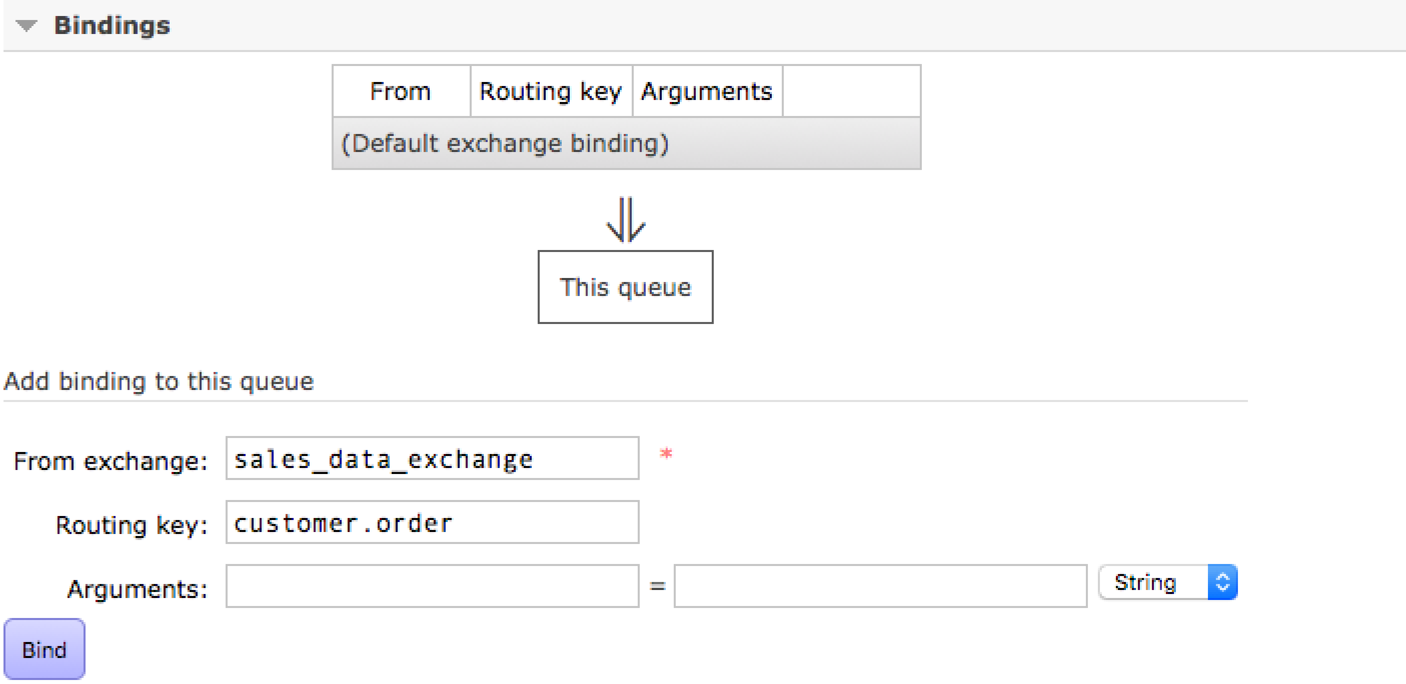
Arguments: = String

Add Message TTL (?) | Auto expire (?) | Max length (?) | Max length bytes (?)
Dead letter exchange (?) | Dead letter routing key (?) | Maximum priority (?)
Lazy mode (?) Master locator (?)

3. Bind the orders_queue for the message exchange (sales_data_exchange) that we have already created.

We will be binding only the **orders_queue** for the **sales_data_exchange**. (As you are already aware, **deliveries_queue** was just created for demonstration purpose)

Click on the **orders_queue** and go to the **Binding** section. Then bind the queue to the message exchange with your desired binding key.



Bindings

From	Routing key	Arguments	
(Default exchange binding)			

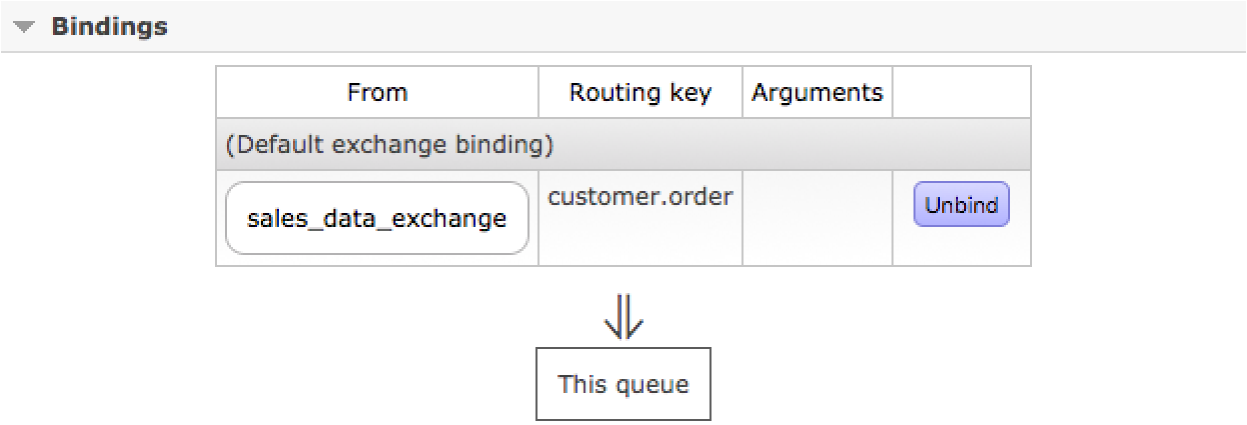
Add binding to this queue

From exchange: sales_data_exchange *

Routing key: customer.order

Arguments: = String

once the queue is successfully bound to the exchange, it will be shown as follows.



From	Routing key	Arguments	
(Default exchange binding)			
sales_data_exchange	customer.order		Unbind

Now we have completed the required RabbitMQ server setup for our article. So Lets look at the project structure and source code. There are two separate spring projects available for representing **Producer** and **Consumer** applications.

Reference:

https://cmatskas.com/getting-started-with-rabbitmq-on-windows/

https://springbootdev.com/2017/09/15/spring-boot-and-rabbitmq-direct-exchange-example-messaging-custom-java-objects-and-consumes-with-a-listener/