

Assignment Two: Wishing Well

Assumptions

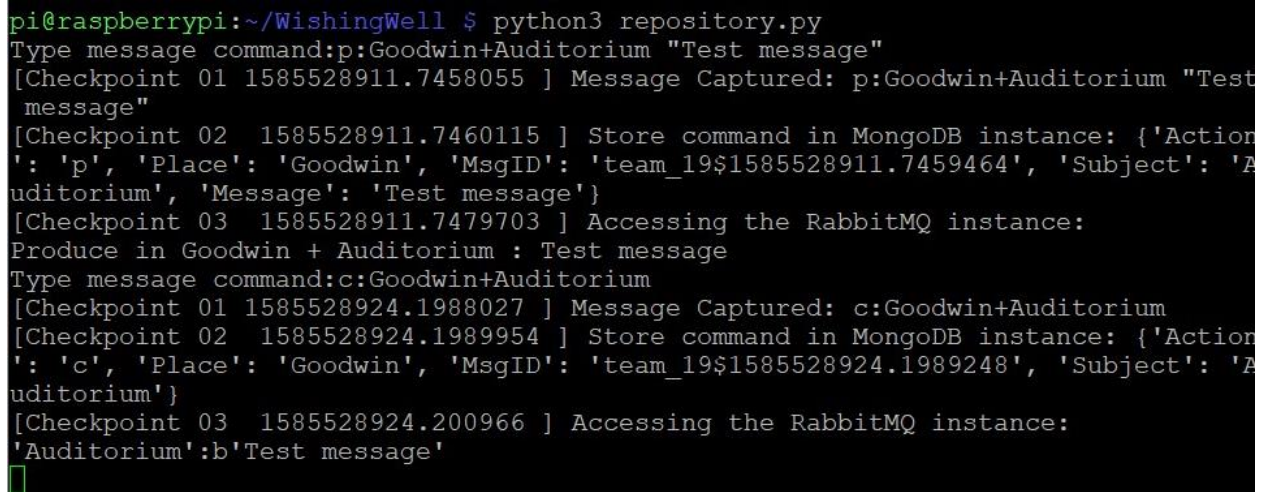
Below is a list of assumptions about the assignment gathered from the PDF:

- Commands are only stored in a single mongo collection
- The “MsgID” field in the mongo pages does replace the “_id” field
- The RabbitMQ cluster is pre-configured using the RabbitMQ management plugin on command line

Implementation

The project was completed in its entirety and works according to the project specifications and the above assumptions.

Evidence of the project working properly on the Raspberry Pi is displayed in **Figure 1**.



```
pi@raspberrypi:~/WishingWell $ python3 repository.py
Type message command:p:Goodwin+Auditorium "Test message"
[Checkpoint 01 1585528911.7458055 ] Message Captured: p:Goodwin+Auditorium "Test message"
[Checkpoint 02 1585528911.7460115 ] Store command in MongoDB instance: {'Action': 'p', 'Place': 'Goodwin', 'MsgID': 'team_19$1585528911.7459464', 'Subject': 'Auditorium', 'Message': 'Test message'}
[Checkpoint 03 1585528911.7479703 ] Accessing the RabbitMQ instance:
Produce in Goodwin + Auditorium : Test message
Type message command:c:Goodwin+Auditorium
[Checkpoint 01 1585528924.1988027 ] Message Captured: c:Goodwin+Auditorium
[Checkpoint 02 1585528924.1989954 ] Store command in MongoDB instance: {'Action': 'c', 'Place': 'Goodwin', 'MsgID': 'team_19$1585528924.1989248', 'Subject': 'Auditorium'}
[Checkpoint 03 1585528924.200966 ] Accessing the RabbitMQ instance:
'Auditorium':b'Test message'
```

Figure 1: Command Line output of repository.py

Messages are then stored in the “commands” collection of the “test” mongo database, as seen below in **Figure 2**.

```

> show collections
> use test
switched to db test
> show collections
commands
system.indexes
utilization
> db.commands.count()
38
> db.commands.find()
{ "_id" : ObjectId("5e7f51ee74fece09eefelbc3"), "Action" : "p", "Place" : "goo",
  "MsgID" : "team_19$1585402350.3974261", "Subject" : "foo", "Message" : "Hiow" }
{ "_id" : ObjectId("5e7f51f974fece09eefelbc4"), "Action" : "c", "Place" : "goo",
  "MsgID" : "team_19$1585402361.4792168", "Subject" : "foo" }
{ "_id" : ObjectId("5e7f555674fece10df355ac1"), "Action" : "p", "Place" : "huhhh",
  "MsgID" : "team_19$1585403222.6804569", "Subject" : "guh", "Message" : "nuhhhh" }
{ "_id" : ObjectId("5e7f558674fece10df355ac2"), "Action" : "c", "Place" : "goo",
  "MsgID" : "team_19$1585403270.7503848", "Subject" : "foo" }
{ "_id" : ObjectId("5e7f626674fece249abfd8ce"), "Action" : "p", "Place" : "Squir",
  "MsgID" : "team_19$1585406566.536453", "Subject" : "Room", "Message" : "Hello" }
{ "_id" : ObjectId("5e7f630174fece2576382a13"), "Action" : "p", "Place" : "Squir",
  "MsgID" : "team_19$1585406721.102897", "Subject" : "Rooms", "Message" : "He

```

Figure 2: Messages shown in Mongo Shell

The messages are then sent to the running instance of RabbitMQ. Queues and message count are listed in **Figure 3**.

```

pi@raspberrypi:~/WishingWell $ sudo rabbitmqctl list_queues
Timeout: 60.0 seconds ...
Listing queues for vhost / ...
Classrooms      0
Wishes          1
Auditorium      0
Meetings        1
Noise           0
Food            0
Seating         0
Rooms           0

```

Figure 3: RabbitMQ queues and message count