MQTT-Client

February 12, 2019

```
In [22]: import time
         import paho.mqtt.client as paho

In [27]: broker="192.168.43.73"
         print(broker)

192.168.43.73


In [28]: #define callback
         def on_message(client, userdata, message):
             time.sleep(1)
             print("received message =",str(message.payload.decode("utf-8")))

In [29]: client= paho.Client("client-001") #create client object client1.on_publish = on_publi
         ######Bind function to callback
         client.on_message=on_message
         #####
         print("connecting to broker ",broker)
         client.connect(broker)#connect
         client.loop_start() #start loop to process received messages
         print("subscribing ")
         client.subscribe("Webservices")#subscribe
         time.sleep(2)
         print("publishing ")
         client.publish("Webservices","Testing Message")#publish
         time.sleep(4)

connecting to broker  192.168.43.73
subscribing
publishing


In [30]: client.disconnect() #disconnect
         client.loop_stop() #stop loop

In [1]: import paho.mqtt.client as mqtt #import the client1
        import time
```

```
In [2]: def on_message(client, userdata, message):
            print("message received " ,str(message.payload.decode("utf-8")))
            print("message topic=",message.topic)
            print("message qos=",message.qos)
            print("message retain flag=",message.retain)


In [3]: broker_address="192.168.43.73"

In [4]: print("creating new instance")
        client = mqtt.Client("P1") #create new instance
        client.on_message=on_message #attach function to callback
        print("connecting to broker")
        client.connect(broker_address) #connect to broker
        client.loop_start() #start the loop

creating new instance
connecting to broker


In [5]: print("Subscribing to topic","Webservices")
        client.subscribe("Webservices")

Subscribing to topic Webservices


Out[5]: (0, 1)

In [6]: print("Publishing message to topic","Webservices")
        client.publish("Webservices","Hello World!!")
        time.sleep(4) # wait
        client.publish("Webservices","This is the second published message")
        time.sleep(4) # wait
        client.publish("Webservices","We can post multiple messages")
        time.sleep(4) # wait

Publishing message to topic Webservices
message received  Hello World!!
message topic= Webservices
message qos= 0
message retain flag= 0
message received  This is the second published message
message topic= Webservices
message qos= 0
message retain flag= 0
message received  We can post multiple messages
message topic= Webservices
message qos= 0
message retain flag= 0
```

```
In [7]: client.loop_stop() #stop the loop
```

```
In [ ]:
```