Example #1: Pub-Sub with Custom Message

Aim

To write a `listener` and `talker` node which should communicate with each other over a ROS Topic called `my_topic` using a custom ROS Message called `myMessage` with the following data fields of the following data types.

1. int32 id
2. string name
3. float32 temperature
4. float32 humidity

Steps

Create Custom ROS Message

- Messages are just simple text files with a field type and field name per line.
- They are stored in the `msg` directory of your package.

1. Create a file and name it `myMessage.msg` and store it in a `msg` folder of `pkg_ros_basics`. If the folder does not exist create it.

2. Now fill the `myMessage.msg` file with the following content.

```
int32 id
string name
float32 temperature
float32 humidity
```

   This is the format of a typical `msg` file.

3. Now open your `package.xml` file of `pkg_ros_basics` package and add in the dependencies for your `message_generation` and `message_runtime` as seen below.

```
<?xml version="1.0"?>
<package format="2">
  <name>pkg_ros_basics</name>
  <version>0.1.0</version>
  <description>The pkg_ey_ros_basics package</description>

  <maintainer email="eyantra@todo.todo">eyantra</maintainer>

  <license>TODO</license>
```

```
  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>geometry_msgs</build_depend>
  <build_depend>rospy</build_depend>
  <build_depend>message_generation</build_depend>

  <build_export_depend>geometry_msgs</build_export_depend>
  <build_export_depend>rospy</build_export_depend>

  <exec_depend>geometry_msgs</exec_depend>
  <exec_depend>rospy</exec_depend>
  <exec_depend>message_runtime</exec_depend>


  <!-- The export tag contains other, unspecified, tags -->
  <export>
    <!-- Other tools can request additional information be placed here -->

  </export>
</package>
```

4. Now open your `CMakeList.txt` file of `pkg_ros_basics` package and navigate to the following block of code in your file.

```
#   FILES
#   Message1.msg
#   Message2.msg
# )
```

Uncomment the Messages and add include the name of your Message files. You can include multiple Message files if required as well.

Now your `CMakeList.txt` should look like this,

```
cmake_minimum_required(VERSION 3.0.2)
project(pkg_ros_basics)


find_package(catkin REQUIRED COMPONENTS
  geometry_msgs
  rospy
  message_generation
)

add_message_files(
  FILES
  myMessage.msg
)

generate_messages(
  DEPENDENCIES
  geometry_msgs
  std_msgs
)


catkin_package(
#  INCLUDE_DIRS include
#  LIBRARIES control_turtle
 CATKIN_DEPENDS geometry_msgs rospy message_runtime

)
```

```
############
## Build ##
############

include_directories(
# include
  ${catkin_INCLUDE_DIRS}
)
```

5. After this build your package.

```
cd ~/catkin_ws
catkin build
```

Once the package is build successfully you can see `myMessage.h` file located at `~/catkin_ws/devel/include/pkg_ros_basics/myMessage.h`. This will be used by ROS Nodes to communicate over a ROS Topic using `myMessage` ROS Message.

Code - ROS Nodes

Listener Node

node_myMsg_listener.py

```
#!/usr/bin/env python

import rospy
from pkg_ros_basics.msg import myMessage


def func_callback_topic_my_topic(myMsg):

    rospy.loginfo("Data Received: (%d, %s, %.2f, %.2f)", myMsg.id,
                  myMsg.name, myMsg.temperature, myMsg.humidity)


def main():

    # 1. Initialize the Subscriber Node.
    rospy.init_node('node_myMsg_listener', anonymous=True)

    # 2. Subscribe to the desired topic and attach a Callback Funtion to it.
    rospy.Subscriber("my_topic", myMessage, func_callback_topic_my_topic)

    # 3. spin() simply keeps python from exiting until this node is stopped
    rospy.spin()


# Python Main
if __name__ == '__main__':
    try:
        main()
    except rospy.ROSInterruptException:
        pass
```

Download

Talker Node

node_myMsg_talker.py

```
#!/usr/bin/env python

import rospy
from pkg_ros_basics.msg import myMessage
```

```
import random


def main():

    # 1. Create a handle to publish messages to a topic.
    var_handle_pub = rospy.Publisher('my_topic', myMessage, queue_size=10)

    # 2. Initializes the ROS node for the process.
    rospy.init_node('node_myMsg_talker', anonymous=True)

    # 3. Set the Loop Rate
    var_loop_rate = rospy.Rate(1) # 1 Hz : Loop will its best to run 1 time in 1 second

    # 4. Write the infinite Loop
    while not rospy.is_shutdown():
        obj_msg = myMessage()

        obj_msg.id = 1
        obj_msg.name = "my_message"
        obj_msg.temperature = 10 + random.random()
        obj_msg.humidity = 20 + random.random()

        rospy.loginfo("Publishing: ")
        rospy.loginfo(obj_msg)

        var_handle_pub.publish(obj_msg)

        var_loop_rate.sleep()



# Python Main
if __name__ == '__main__':
    try:
        main()
    except rospy.ROSInterruptException:
        pass
```

Download

Output

For analyzing the output for these custom messages, you can follow the following steps

1. `roscore` - As seen in previous tutorials, you must have a roscore running for the nodes to communicate. To view the messages between the talker and listener nodes, run an instance of roscore in a separate terminal window

2. listener node - For making the script of your node executable run `chmod +x` within the appropriate directory in a separate terminal window other than where your roscore is running. To run the listener node, run the following commands within your appropriate directory
   `rosrun <package_name> <listener_node.py>`

3. talker node - You can follow the same steps mentioned above for running your talker node using the commands `rosrun <package_name> <talker_node.py>`

If you follow the steps given above, you should see the following output

```
rosrun pkg_ros_basics node_myMsg_talker.py
```

```
[INFO] [1601406458.926833]: Publishing:
[INFO] [1601406458.929838]: id: 1
name: "my_message"
temperature: 10.7699381016
humidity: 20.5365646883
[INFO] [1601406459.928043]: Publishing:
[INFO] [1601406459.931639]: id: 1
name: "my_message"
temperature: 10.510987459
humidity: 20.4537870911
[INFO] [1601406460.927918]: Publishing:
```

```
[INFO] [1601406460.930950]: id: 1
name: "my_message"
temperature: 10.1001665043
humidity: 20.2251078658
[INFO] [1601406461.928028]: Publishing:
[INFO] [1601406461.934640]: id: 1
name: "my_message"
temperature: 10.8662984056
humidity: 20.527154066
[INFO] [1601406462.928025]: Publishing:
[INFO] [1601406462.931750]: id: 1
name: "my_message"
temperature: 10.3771277955
humidity: 20.8530501433
[INFO] [1601406463.927601]: Publishing:
[INFO] [1601406463.928682]: id: 1
name: "my_message"
temperature: 10.6145392272
humidity: 20.8280485139
```

```
rosrun pkg_ros_basics node_myMsg_listener.py
```

```
[INFO] [1601406458.933924]: Data Received: (1, my_message, 10.77, 20.54)
[INFO] [1601406459.936372]: Data Received: (1, my_message, 10.51, 20.45)
[INFO] [1601406460.934484]: Data Received: (1, my_message, 10.10, 20.23)
[INFO] [1601406461.935742]: Data Received: (1, my_message, 10.87, 20.53)
[INFO] [1601406462.936444]: Data Received: (1, my_message, 10.38, 20.85)
[INFO] [1601406463.929752]: Data Received: (1, my_message, 10.61, 20.83)
```

- If you face any problems while following this tutorial you can feel free to reach out to us.