创建一个包含定制Message的任务

建立一个消息传输机制，发布的topic加载定制的机器人位姿信息供接收节点订阅。

实现过程

定制message格式，修改CmakeLists.txt和package.xml，采用发布和接收节点，编程调用该topic

编程实现

创建一个ROS package

首先，我们要创建一个ROS package，它将包含我们的节点。发布节点通过某一个topic发布新的定制message,接收节点接收该新的message。ROS具有内建的命令catkin完成创建ROS package工作。

转到工作空间里的代码子目录:

cd catkin_ws/src

然后跑下面命令创建名为basic_ros_msg工作包（package,名称任意):

catkin_create_pkg basic_ros_msg std_msgs rospy roscpp

然后，我们在catkin_ws/src/basic_ros_msg目录中创建一个新子目录，名为msg. 该msg子目录里含有我们的定制消息（message）文件。进入该msg子目录,创建一个名为RobotPose.msg的文档（文档名任意，但扩展名必须为msg),拷贝黏贴以下4行作为该文档内容（分别是机器人ID，和位姿信息）。

string id

float64 x

float64 y

float64 angle

该定制的消息定义文档让我们可以定制机器人在平面的坐标和朝向信息，创建发布节点把这个信息发布给感兴趣的节点订阅。

转换到本程序包src子目录，即，现在你在该位置：
USERNAME/catkin_ws/src/basic_ros_msg/src

在该src子目录内，我们创建一个python程序user_msg_pub.py。此处程序名可以取任意名字。

以下是你的程序包的结构:

```
catkin_ws
  - build
  - devel
  - src
    - basic_ros_msg
      - CMakeLists.txt
      - package.xml
      - include {This directory contains nothing}
      - CMakeLists.txt
      - src
        - user_msg_pub.py
      - msg
        - basic_ros_msg.msg
```

注意，msg子目录和src子目录同级。现在我们来编写代码。

```
#! /usr/bin python
import rospy
from basic_ros_msg.msg import RobotPose
```

```
def main():
    rospy.init_node("user_msg_pub")
    pub = rospy.Publisher("/robot_pose", RobotPose, queue_size=10)
    msg = RobotPose()
    msg.id = "vbot"
    msg.x = 52.1
    msg.y = 12.6
    msg.angle = 180.0

    while not rospy.is_shutdown():
        pub.publish(msg)
        rospy.loginfo("Pub robot: {}, pose({}, {}, {})".format(msg.id, msg.x, msg.y, msg.angle))
        rospy.sleep(1)

if __name__ == "__main__":
    main()
```

将以上代码存储后，将它变为可执行脚本。

```
sudo chmod +x user_msg_pub.py
```

打开CMakeLists.txt，将我们的python脚本加入CMakeLists.txt适当位置如下。如此将使得catkin会编译我们的脚本。

```
include_directories(
# include
```

```
  ${catkin_INCLUDE_DIRS}

  src/user_msg_pub.py

)
```

然后，我们需要使得我们的msg文档也被编译，为此我们需要将下面内容也加入CMakeLists.txt中：

```
## Generate messages in the 'msg' folder

 add_message_files(

   FILES

   RobotPose.msg

)

## Generate added messages and services with any dependencies listed here

 generate_messages(

   DEPENDENCIES

   std_msgs

 )
```

同时，在 find_package 部分增加依赖项:

```
find_package(catkin REQUIRED COMPONENTS

  roscpp

  rospy

  std_msgs

  message_generation

)
```

我们也需要将上述依赖项加入package.xml文件中的适当位置如下。

```
  <build_depend>message_generation</build_depend>
```

```
  <exec_depend>message_runtime</exec_depend>
```

现在返回catkin_ws进行编译，运行catkin_make。编译完以后运行以下命令：

```
source devel/setup.bash
```

如果成功编译，则大部分工作已经完成了。

用命令行运行程序包

启动ROS Master

```
roscore
```

现在，打开一个新终端，运行以下命令：

```
rosrun basic_ros_msg user_msg_pub.py
```

你可查看运行节点：

```
rosnode list
```

我们可见 user_msg_pub节点。

用接收节点程序接收结果

将以下脚本存为文件名：user_msg_sub.py。

```
#! /usr/bin python

import rospy

from basic_ros_msg.msg import RobotPose


def robotPoseCallback(msg):

    rospy.loginfo("Sub robot: {}, pose({}, {}, {})".format(msg.id, msg.x, msg.y, msg.angle))
```

```
def main():
    rospy.init_node("user_msg_sub")
    rospy.Subscriber("/robot_pose", RobotPose, robotPoseCallback, queue_size=10)
    rospy.spin()


if __name__ == "__main__":
    main()
```

结语：

通过此练习，你学会了如何创建新的定制消息，并采用发布、接收节点传递新消息。你也学会了如何定制message.最后，你也获知了如何修改CMakeLists.txt和package.xml文件。

注：发布、接收节点分别有一个typo，可通过搜索网络改正。