基本概念

msg消息 是ROS topic通讯节点间传递的内容。

msg消息描述的是业务间传递数据，也可以理解为业务间数据的抽象化。

常见Msg消息

std消息

std消息属于基本数据类型消息，和python类似，描述的是数字，字符串，布尔类型等。

!!!note

- `std_msgs/Bool`
- `std_msgs/Byte`
- `std_msgs/ByteMultiArray`
- `std_msgs/Char`
- `std_msgs/ColorRGBA`
- `std_msgs/Duration`
- `std_msgs/Empty`
- `std_msgs/Float32`
- `std_msgs/Float32MultiArray`
- `std_msgs/Float64`
- `std_msgs/Float64MultiArray`
- `std_msgs/Header`
- `std_msgs/Int16`
- `std_msgs/Int16MultiArray`
- `std_msgs/Int32`
- `std_msgs/Int32MultiArray`
- `std_msgs/Int64`
- `std_msgs/Int64MultiArray`
- `std_msgs/Int8`
- `std_msgs/Int8MultiArray`
- `std_msgs/MultiArrayDimension`
- `std_msgs/MultiArrayLayout`
- `std_msgs/String`
- `std_msgs/Time`
- `std_msgs/UInt16`
- `std_msgs/UInt16MultiArray`
- `std_msgs/UInt32`
- `std_msgs/UInt32MultiArray`
- `std_msgs/UInt64`
- `std_msgs/UInt64MultiArray`
- `std_msgs/UInt8`
- `std_msgs/UInt8MultiArray`

geometry消息

空间数据描述

!!!note
- `geometry_msgs/Accel`
- `geometry_msgs/AccelStamped`
- `geometry_msgs/AccelWithCovariance`
- `geometry_msgs/AccelWithCovarianceStamped`
- `geometry_msgs/Inertia`
- `geometry_msgs/InertiaStamped`
- `geometry_msgs/Point`
- `geometry_msgs/Point32`
- `geometry_msgs/PointStamped`
- `geometry_msgs/Polygon`
- `geometry_msgs/PolygonStamped`
- `geometry_msgs/Pose`
- `geometry_msgs/Pose2D`
- `geometry_msgs/PoseArray`
- `geometry_msgs/PoseStamped`
- `geometry_msgs/PoseWithCovariance`
- `geometry_msgs/PoseWithCovarianceStamped`
- `geometry_msgs/Quaternion`
- `geometry_msgs/QuaternionStamped`
- `geometry_msgs/Transform`
- `geometry_msgs/TransformStamped`
- `geometry_msgs/Twist`
- `geometry_msgs/TwistStamped`
- `geometry_msgs/TwistWithCovariance`
- `geometry_msgs/TwistWithCovarianceStamped`
- `geometry_msgs/Vector3`
- `geometry_msgs/Vector3Stamped`
- `geometry_msgs/Wrench`
- `geometry_msgs/WrenchStamped`

sensor消息

传感器消息

!!!note
- `sensor_msgs/BatteryState`
- `sensor_msgs/CameraInfo`
- `sensor_msgs/ChannelFloat32`
- `sensor_msgs/CompressedImage`
- `sensor_msgs/FluidPressure`
- `sensor_msgs/Illuminance`
- `sensor_msgs/Image`
- `sensor_msgs/Imu`
- `sensor_msgs/JointState`
- `sensor_msgs/Joy`
- `sensor_msgs/JoyFeedback`
- `sensor_msgs/JoyFeedbackArray`

`sensor_msgs/LaserEcho`
`sensor_msgs/LaserScan`
`sensor_msgs/MagneticField`
`sensor_msgs/MultiDOFJointState`
`sensor_msgs/MultiEchoLaserScan`
`sensor_msgs/NavSatFix`
`sensor_msgs/NavSatStatus`
`sensor_msgs/PointCloud`
`sensor_msgs/PointCloud2`
`sensor_msgs/PointField`
`sensor_msgs/Range`
`sensor_msgs/RegionOfInterest`
`sensor_msgs/RelativeHumidity`
`sensor_msgs/Temperature`
`sensor_msgs/TimeReference`

一些需求

已知有两个节点，其中一个节点是`Publisher`，另外一个节点是`Subscriber`。

`Publisher`发布的消息是学生数据，包含了学生的`姓名`和`年龄`。

要求实现这个两个节点，并且模拟数据发布和订阅！

分析

`Publisher`和`Subscriber`的编写我们已经会了。

但是中间传输的数据类型，默认系统是没有提供的。

这个时候我们需要自己定制消息格式。

总的来说，这个自定义的消息格式中，需要包含`name`和`age`。

通常在编程语言中，我们还是要去确定消息字段中的类型。

例如，Python程序，表达数据类型是这样的:

```
name = ''
age = 0
```

python中省掉了声明类型，我们用值来表示类型。

实现步骤

- 首先自定义消息
- 实现publisher
- 实现subscriber
- 模拟数据发布

自定义消息

在Ros中，如果没有现成的消息类型来描述要去传递的消息时，我们会自定义消息。

通常我们会新建一个Package来去自定义消息，这个Package一般不去写任何的业务逻辑，只是用来声明自定义的消息类型，可以只定义一种消息类型，也可以定义多种消息类型，根据业务需求来定。

所以，我们会单独的创建一个package，用来定义消息.

创建package

包名取名也是有讲究的，`业务名_msgs`。大家可以看看std_msgs，geometry_msgs，都是遵循这个规则。

创建msg目录

在package包下新建msg文件夹

创建msg文件

在msg文件夹下创建`.msg`文件. `.msg`文件就是自定义消息文件，用来描述消息格式的。

例如案例中, 我们会去创建`Student.msg`文件, 内容如下:

```
string name
int64 age
```

这个自定义的消息包含两个数据形式，`name`和`age`，`name`的类型 是`string`，`age`的类型是`int64`。

这个msg文件其实遵循一定规范的，每一行表示一种数据。前面是类型，后面是名称。

ros不只是提供了`int64`和`string`两种基本类型供我们描述，其实还有很多，具体可以参考附录的内容

配置package.xml

在package.xml种添加如下配置:

```
<build_depend>message_generation</build_depend>
<exec_depend>message_runtime</exec_depend>
```

配置CMakeLists.txt

find_package配置

在`find_package`添加`message_generation`，结果如下:

```
find_package(catkin REQUIRED COMPONENTS
  roscpp
  rosmsg
  rospy
  message_generation
)
```

add_message_file配置

添加`add_message_file`，结果如下:

```
add_message_files(
        FILES
        Student.msg
)
```

!!!tip
    这里的`Student.msg`要和你创建的msg文件名称一致，且必须时在msg目录下，否则编译会出现问题

generation_msg配置

添加`generation_msg`，结果如下:

```
generate_messages(
        DEPENDENCIES
        std_msgs
)
```

!!!tip
    这个配置的作用是添加生成消息的依赖，默认的时候要添加`std_msgs`

catkin_package配置

修改`catkin_package`，结果如下:

```
catkin_package(
#  INCLUDE_DIRS include
#  LIBRARIES demo_msg
   CATKIN_DEPENDS roscpp rosmsg rospy message_runtime
#  DEPENDS system_lib
)
```

!!!tip
    为catkin编译提供了依赖`message_runtime

编译项目

```
cd 工作空间
catkin_make
```

校验

devel文件夹校验

来到`devel`的`lib/python2.7/dist-package`目录下，查看是否生成和package名称相同的目录，以及目录内是否生成对应的py文件。

命令校验

```
rosmsg show hello_msgs/Student
```

需求升级

已知有两个节点，其中一个节点是`Publisher`，另外一个节点是`Subscriber`。

`Publisher`发布的消息是 团队数据，包含了以下内容:

名称	类型	描述
name	`string`	团队名称
leader	`Student`	团队领导
intro	`std_msgs/String`	团队介绍
location	`geometry_msgs/Pose`	位置
members	多个元素	团队成员

要求实现这个两个节点，并且模拟数据发布和订阅!