创建一个包含定制Service的任务

输入二个数值，经计算返回这二个数值的乘积。

实现过程

采用一个节点，编程调用service,步骤如下。

1. 获取输入数值

2. 计算乘积

3. 通过topic返回乘积值

编程实现

创建一个ROS package

首先，我们要创建一个ROS package，它将是我们的节点。该节点包含通过topic发布message的代码。ROS具有内建的命令catkin完成该工作。

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工作包（package):

catkin_create_pkg basic_ros_service std_msgs rospy roscpp

然后，我们在catkin_ws/src/basic_ros_service目录中创建一个新子目录，名为srv. 该srv子目录里含有我们的定制服务文件。进入该srv子目录,创建一个名为Product.srv的文档（文档名任意，但扩展名必须为srv),拷贝黏贴以下4行作为该文档内容。

float64 a

float64 b

---

float64 product

前二个浮点定义输入，用三个横线分隔，后一个浮点是乘积值。该服务定义文档让我们可以定义定制的输入输出数据服务。

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_service/src

在该src子目录内，我们创建一个python程序ProductServer.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_service
      - CMakeLists.txt
      - package.xml
      - include {This directory contains nothing}
      - CMakeLists.txt
      - src
        - ProductServer.py
      - srv
        - Product.srv
```

现在我们来编写代码。首先程序中必须包含程序库rospy及std_msgs。

```
#!/usr/bin/env python
from __future__ import print_function
from basic_ros_service.srv import Product, ProductResponse
```

```
import rospy

import math

def product_func(a, b):

    return a*b

def compute(req): # Find the product from the inputs req.a and req.b and return the product.

    product = product_func(req.a, req.b)

    print("Returning [%s]" % (product))

    return ProductResponse(product)

def product_server(): # Start the node and server.

    rospy.init_node('basic_ros_service')

    s = rospy.Service('basic_ros_service', Product, compute)

    print("Ready to compute.")

    rospy.spin()

if __name__ == "__main__":

    product_server()
```

将以上代码存储后，将它变为可执行脚本。

```
sudo chmod +x ProductServer.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/ProductServer.py
```

)

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

```
## Generate services in the 'srv' folder
 add_service_files(
   FILES
   Product.srv
)
## 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_service ProductServer.py

你可查看运行节点：

rosnode list

我们可见 basic_ros_service节点。

查看服务：

rosservice list

可见basic_ros_service程序包的service及一些记录服务。下面我们采用命令行调用服务。

rosservice call /basic_ros_service "a: 2.0 b: 2.0"

提示: 如你忘记命令格式，可双击TAB键，ROS会自动提示命令。

我们可见计算结果 4.0。

用程序计算结果

将以下脚本存为文件名：ProductClient.py。

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

from __future__ import print_function

import sys
```

```
import rospy
from basic_ros_service.srv import *
def product_client(a, b):
    rospy.wait_for_service('basic_ros_service') # Find our service
    try:
        # Connect to service
        solver = rospy.ServiceProxy('basic_ros_service', Product)
        resp1 = solver(a, b) # Make a request
        return resp1.product # Return the result
    except rospy.ServiceException as e:
        print("Service call failed: %s"%e)
def usage():
    return "%s [a b]"%sys.argv[0]
if __name__ == "__main__":
    if len(sys.argv) == 3:
        a = float(sys.argv[1])
        b = float(sys.argv[2])
    else:
        print(usage())
        sys.exit(1)
    print("Requesting %s*%s"%(a, b))
    print("%s * %s = %s"%(a, b, product_client(a, b)))
```

存储该pytho脚本后,也要记得将其属性改为可执行。然后像以前运行publisher节点一样运行它：

```
rosrun basic_ros_service ProductClient.py 2 2
```

结语：

通过此练习，你学会了如何创建Service的服务器和客户端。你也学会了如何定制服务的message.最后，你也获知了如何修改CMakeLists.txt和package.xml文件。