ROS Service 练习：

此练习将巩固你关于使用 ROS Services 的技巧。我们将讨论从生成一个功能包（package），直到使用 Services 以及打印结果的全流程。

- ROS 服务是对同期远程调用的回应
- ROS 服务含有输入输出，即请求应答，其输入输出的定义和消息类似
- 通常在需要应答的情况下使用 ROS 服务。比如，要求机器人在遇到障碍物时拍照等。

此例子将使用随机数作为输入，ON/OFF 作为输出。

第一步，在你的某一个 catkin 空间的 src 子目录里生成一个 ROS 功能包（你也可选择使用已经存在的 ROS 功能包而省略此 4 步）

```
catkin_create_pkg ros_service rospy roscpp std_msgs

cd ..

catkin_make

source devel/setup.bash
```

完成后，你将创建了一个新的名为 ros_service 的功能包。

第二步， 在功能包内建立一个名为 srv 的子目录，在此子目录里定义你的 service 的输入和输出数据种类。

```
roscd ros_service

mkdir srv
```

用你的文本编辑器，产生一个名为 ServiceExample.srv 的文档（可任意取文档名，但一定要使用.srv 作为扩展名），在此文档内输入如下内容。

```
Int32 onezero
---
string turn
```

三个横线表明输入定义结束，输出定义开始。以上定义好了 ROS 服务。

第三步，现在开始，需要修改二个文件，使得新的 ROS 服务可被编译，从而自动产生相应

的处理程序和类定义。打开此功能包的 package.xml 文档，确保以下内容在此文档中。

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

做出以上修改后，存储此文档。

打开此功能包的 CMakeLists.txt 文档，我们需要加入依赖项和新的服务文件（此文档内有很多行被 comment out，找到下面一些被 comment out 的行恢复使用并根据下面的内容修改，非常方便）。确保红色内容被包括。

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

add_service_files (
   FILES
   ServiceExample.srv
)

generate_messages (
   DEPENDENCIES
   std_msgs
)

catkin_package(
   INCLUDE_DIRS include
   LIBRARIES ros_service
   CATKIN_DEPENDS roscpp rospy std_msgs
   DEPENDS system_lib
)
```

做出以上修改后，存储此文档。

退回功能包子目录，运行以下命令编译产生库程序，即可生成服务。

```
catkin_make

source devel/setup.bash
```

第四步, 在功能包的 scripts 或 src 子目录里, 新建一个 Python 程序作为调用新服务的 ROS 节点。任意给出程序名, callService.py, 记得使此程序变为可执行程序(chmod +x 程序名.py)。

程序内容如下，请分析理解。

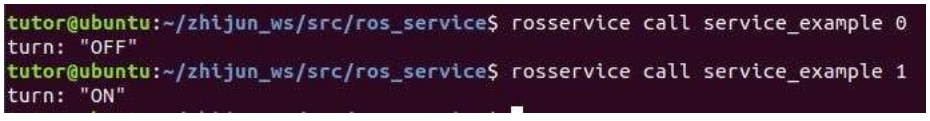
```
#!/usr/bin/env python

import rospy
#import the code generated by catkin.
#we need ServiceExample for the first message type,
#and the ServiceExampleResponse from the second message type from the  ServiceExample.srv file
from ros_service.srv import ServiceExample, ServiceExampleResponse

def turn_on_off(mess):
  if mess.onezero==1:
    return  ServiceExampleResponse('ON')
  else:
    return  ServiceExampleResponse('OFF')

rospy.init_node('service_respond')

service=rospy.Service('service_example',ServiceExample,turn_on_off)

rospy.spin()
```

第五步，测试服务
在确信 rosmaster 已经工作的情况下，打开一个新终端，运行如下命令:

```
rosrun ros_service callService.py
```

此时，新的 ROS 服务已经运行，为了看到输出，必须提供输入。

```
rosservice call service_example 0
```



第六步，编程使用新的 ROS 服务

当你测试完确保新的 service 可以工作后，你可在你的程序中使用它，简单例子如下。新建立一个名为 useService.py 程序，输入以下内容存储后，记得 chmod +x useService.py，使得

它可执行。

```
#!/usr/bin/env python
import rospy
from ros_service.srv import ServiceExample
import sys

rospy.init_node('use_service')

#wait the service to be advertised, otherwise the service use will fail
rospy.wait_for_service('service_example')

#setup a local proxy for the service
srv=rospy.ServiceProxy('service_example',ServiceExample)

#use the service and send it a value. In this case, I can send 1 or 0
service_example=srv(0)

#print the result from the service
print(service_example)
```

课后如有时间，请参照网址 http://wiki.ros.org/rospy/Overview/Services 复习 ROS Services 概念。