AI & Robotics

ROS: Simple Service & Client

PXL AI & ROBOTICS LAB

Goals

The **junior-colleague**

- can create and run their own ROS service using Python
- can create and run their own ROS client using Python

Creating a Service

```
$ catkin_create_pkg first_service \
> message_generation message_runtime std_msgs rospy

   . . .

$ mkdir first_service/srv
$ gedit first_service/srv/AddTwoInts.srv
   int64 a
   int64 b
   ---
   int64 sum
```

Creating a Service

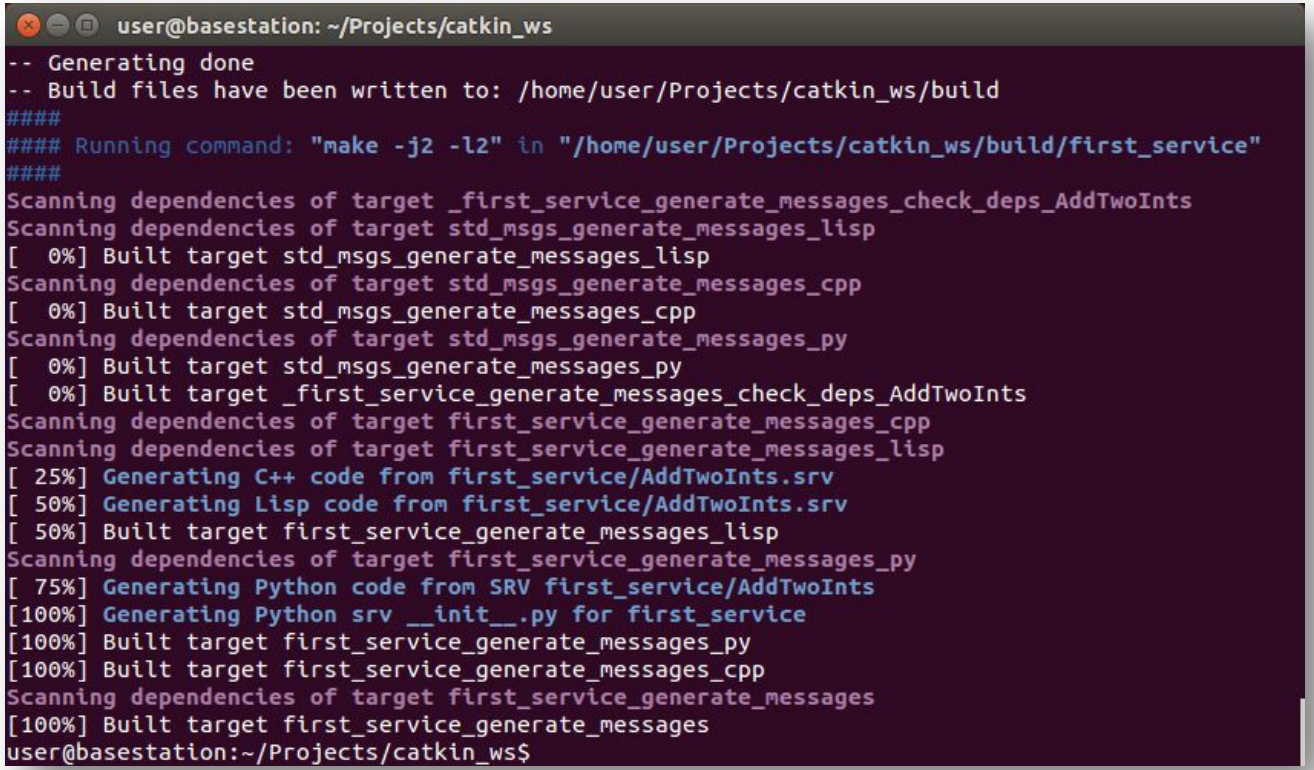
```
$ gedit CMakeLists.txt
add_service_files(
     FILES
     AddTwoInts.srv
)

generate_messages(
     DEPENDENCIES
     std_msgs
)
$ rossrv show first_service/AddTwoInts
int64 a
int64 b
---
int64 sum
$
```

Creating a Service

```
$ # In your catkin_ws, not in src!
$ catkin_make --pkg first_service
$
```

Build


user@basestation: ~/Projects/catkin_ws
-- Generating done
-- Build files have been written to: /home/user/Projects/catkin_ws/build
####
Running command: "make -j2 -l2" in "/home/user/Projects/catkin_ws/build/first_service"
####
Scanning dependencies of target _first_service_generate_messages_check_deps_AddTwoInts
Scanning dependencies of target std_msgs_generate_messages_lisp
[0%] Built target std_msgs_generate_messages_lisp
Scanning dependencies of target std_msgs_generate_messages_cpp
[0%] Built target std_msgs_generate_messages_cpp
Scanning dependencies of target std_msgs_generate_messages_py
[0%] Built target std_msgs_generate_messages_py
[0%] Built target _first_service_generate_messages_check_deps_AddTwoInts
Scanning dependencies of target first_service_generate_messages_cpp
Scanning dependencies of target first_service_generate_messages_lisp
[25%] Generating C++ code from first_service/AddTwoInts.srv
[50%] Generating Lisp code from first_service/AddTwoInts.srv
[50%] Built target first_service_generate_messages_lisp
Scanning dependencies of target first_service_generate_messages_py
[75%] Generating Python code from SRV first_service/AddTwoInts
[100%] Generating Python srv __init__.py for first_service
[100%] Built target first_service_generate_messages_py
[100%] Built target first_service_generate_messages_cpp
Scanning dependencies of target first_service_generate_messages
[100%] Built target first_service_generate_messages
user@basestation:~/Projects/catkin_ws$

Service Node

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

from first_service.srv import *
import rospy

def handle_add_two_ints(req):
     print "Returning [%s + %s = %s]"%(req.a, req.b, (req.a + req.b))
     return AddTwoIntsResponse(req.a + req.b)

def add_two_ints_server():
     rospy.init_node('add_two_ints_server')
     s = rospy.Service('add_two_ints', AddTwoInts, handle_add_two_ints)
     print "Ready to add two ints."
    rospy.spin()

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

service.py

Client Node

```
#!/usr/bin/env python
import roslib
import sys
import rospy
from first_service.srv import *

def add_two_ints_client(x, y):
    rospy.wait_for_service('add_two_ints')
    try:
        add_two_ints = rospy.ServiceProxy('add_two_ints', AddTwoInts)
        resp1 = add_two_ints(x, y)
        return resp1.sum
    except rospy.ServiceException, e:
        print "Service call failed: %s"%e

def usage():
    return "%s [x y]"%sys.argv[0]
```

client.py

Client Node

```
if __name__ == "__main__":
   if len(sys.argv) == 3:
       x = int(sys.argv[1])
       y = int(sys.argv[2])
   else:
       print usage()
       sys.exit(1)
   print "Requesting %s+%s"%(x, y)
   print "%s + %s = %s"%(x, y, add_two_ints_client(x, y))
```

client.py

Running

Terminal 1

```
$ roscore
```

Terminal 2

```
$ rosrun first_service service.py
Ready to add two ints.
Returning [1 + 1 = 2]
Returning [1 + 112 = 113]
Returning [333 + 765 = 1098]
```

Terminal 3

```
$ rosrun first_service client.py
/home/user/Projects/catkin_ws/src/first_service/src/client.py [x y]
$ rosrun first_service client.py 1 1
Requesting 1+1
1 + 1 = 2
$ rosrun first_service client.py 1 112
Requesting 1+112
1 + 112 = 113
$ rosrun first_service client.py 333 765
Requesting 333+765
333 + 765 = 1098
```