

9. Custom service messages and usage

9. Custom service messages and usage

9.1 Custom service message

9.1.1 define srv file

9.1.2 Add function package dependencies in package.xml

9.1.3 Add compile options in CMakeLists.txt

9.1.4 Compile and generate language-related files

9.1.5 C++ language implementation

9.1.6 Python language implementation

9.1 Custom service message

Switch to the `~/catkin_ws/src/learning_server` function package directory, and then create a new folder named `srv` to store custom service messages.

9.1.1 define srv file

Switch to the `srv` directory, create a new blank `srv` file, and use `srv` as the suffix to indicate that it is a `srv` file. Here we take `IntPlus.srv` as an example, and copy the following code into the `srv` file just created.

```
uint8    a
uint8    b

---
uint8    result
```

The composition of the `srv` file is described here, which is divided into upper and lower parts by the symbol `---`, the upper part indicates the request and the lower part is the response.

9.1.2 Add function package dependencies in package.xml

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

9.1.3 Add compile options in CMakeLists.txt

```
Add message_generation to find_package
add_service_files(FILES IntPlus.srv)
generate_messages(DEPENDENCIES std_msgs)
```

9.1.4 Compile and generate language-related files

```
cd ~/catkin_ws
catkin_make
```

9.1.5 C++ language implementation

1. switch to `~/catkin_ws/src/learning_server/src`, create two new cpp files, named `IntPlus_server.cpp` and `IntPlus_client.cpp`, and copy the following codes into them respectively,

`IntPlus_server.cpp`

```
/**
 * This routine will execute /Two_Int_Plus service, service data type
 learning_service::IntPlus
 */

#include <ros/ros.h>
#include "learning_server/IntPlus.h"

// service callback function, input parameter req, output parameter res
bool IntPlusCallback(learning_service::IntPlus::Request & req,
                     learning_service::IntPlus::Response & res)
{
    ROS_INFO("number 1 is:%d,number 2 is:%d ", req.a, req.b); // Display request
    data

    res.result = req.a + req.b; // The feedback result is the sum of the two
    numbers

    return res.result;
}

int main(int argc, char ** argv)
{
    ros::init(argc, argv, "IntPlus_server"); // ROS node initialization

    ros::NodeHandle n; // create node handle

    // Create a server and register the callback function IntPlusCallback
    ros::ServiceServer Int_Plus_service = n.advertiseService("/Two_Int_Plus",
IntPlusCallback);

    // loop waiting for the callback function
    ROS_INFO("Ready to caculate.");
    ros::spin();

    return 0;
}
```

`IntPlus_client.cpp`

```
/*
 This routine will request /Two_Int_Plus service, service data type
 learning_service::IntPlus
 Add and sum two integers
 */
```

```

#include <ros/ros.h>
#include "learning_server/IntPlus.h"
#include <iostream>
using namespace std;
int main(int argc, char** argv)
{
    int i, k;
    cin >> i;
    cin >> k;

    ros::init(argc, argv, "IntPlus_client"); // initialize the ROS node

    ros::NodeHandle node; // create node handle

    // After discovering the /Two_Int_Plus service, create a service client
    ros::service::waitForService("/Two_Int_Plus");
    ros::ServiceClient IntPlus_client = node.serviceClient <
learning_server::IntPlus >("/Two_Int_Plus");

    // Initialize the request data for learning_service::IntPlus
    learning_server::IntPlus srv;
    srv.request.a = i;
    srv.request.b = k;

    ROS_INFO("Call service to plus %d and %d", srv.request.a, srv.request.b); //
request service call

    IntPlus_client.call(srv);

    // Display the result of the service call
    ROS_INFO("Show the result : %d", srv.response.result); // show the result of
the service call

    return 0;
}

```

2. modify the CMakeLists.txt file

```

add_executable(IntPlus_server src/IntPlus_server.cpp)
target_link_libraries(IntPlus_server ${catkin_LIBRARIES})
add_dependencies(IntPlus_server ${PROJECT_NAME}_generate_messages_cpp)

add_executable(IntPlus_client src/IntPlus_client.cpp)
target_link_libraries(IntPlus_client ${catkin_LIBRARIES})
add_dependencies(IntPlus_client ${PROJECT_NAME}_generate_messages_cpp)

```

3. the core part

The implementation process here is the same as before, the main difference is the introduction of header files and the use of custom service files:

The import header file is

```
#include "learning_server/IntPlus.h"
```

The front learning_server is the function package name, and the latter IntPlus.h is the header file name generated by the srv file just created

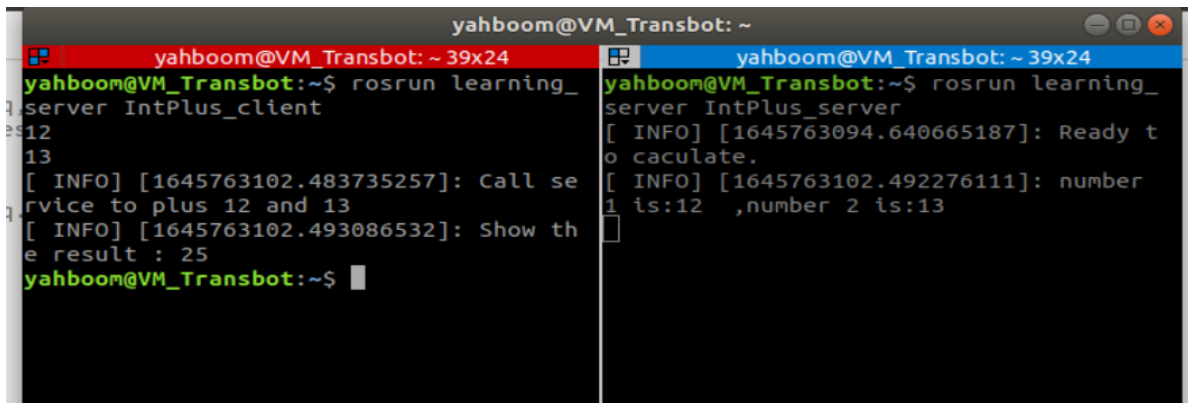
Using a custom service file is

```
client:
learning_server::IntPlus srv;
srv.request.a = i;
srv.request.b = k;
#i, k is the addend input by the terminal
ros::ServiceClient IntPlus_client = node.serviceClient <
learning_server::IntPlus >("/Two_Int_Plus");
IntPlus_client.call(srv);
server:
ros::ServiceServer Int_Plus_service = n.advertiseService("/Two_Int_Plus",
IntPlusCallback);
bool IntPlusCallback(learning_server::IntPlus::Request & req,
learning_server::IntPlus::Response & res)
```

4. run the program

```
roscore
roslaunch learning_server IntPlus_client
roslaunch learning_server IntPlus_server
```

5. run the screenshot

The screenshot shows two terminal windows side-by-side. The left window, titled 'yahboom@VM_Transbot: ~ 39x24', shows the execution of 'roslaunch learning_server IntPlus_client'. It displays the input of numbers 12 and 13, followed by log messages from the client: '[INFO] [1645763102.483735257]: Call service to plus 12 and 13' and '[INFO] [1645763102.493086532]: Show the result : 25'. The right window, also titled 'yahboom@VM_Transbot: ~ 39x24', shows the execution of 'roslaunch learning_server IntPlus_server'. It displays log messages from the server: '[INFO] [1645763094.640665187]: Ready to calculate.' and '[INFO] [1645763102.492276111]: number 1 is:12 ,number 2 is:13'.

6. program description

After running IntPlus_server, you will be prompted to prepare for calculation; after running IntPlus_client, the terminal will input two integer numbers, then IntPlus_server will calculate the result, return it to IntPlus_client, and then print the result.

9.1.6 Python language implementation

1. switch to ~/catkin_ws/src/learning_server/script, create two new py files, named

IntPlus_server.py and IntPlus_client.py, and copy the following codes into them respectively,

IntPlus_server.py

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

```

# -*- coding: utf-8 -*-
import rospy

from learning_server.srv import IntPlus, IntPlusResponse

def IntPlusCallback(req):

    rospy.loginfo("Ints: a:%db:%d", req.a, req.b) # show request data

    return IntPlusResponse(req.a + req.b) # Feedback data

def IntPlus_server():

    rospy.init_node ' IntPlus_server(') # ROS node initialization

    # Create a server and register the callback function IntPlusCallback
    s = rospy.Service('/Two_Int_Plus', IntPlus, IntPlusCallback)

    print "Ready to caculate two ints." # Loop waiting for the callback
function

    rospy.spin()

if __name__ == "__main__" :
    IntPlus_server()

```

IntPlus_client.py

```

#!/usr/bin/env python
# -*- coding: utf-8 -*-
import sys
import rospy
from learning_server.srv import IntPlus, IntPlusRequest

def Plus_client():

    # ROS node initialization
    rospy.init_node('IntPlus_client')

    rospy.wait_for_service('/Two_Int_Plus')
    try :
        Plus_client = rospy.ServiceProxy('/Two_Int_Plus', IntPlus)

        response = Plus_client(22, 20) # Request service call, enter request
data

        return response.result
    except rospy.ServiceException, e :
        print "failed to call service : %s" % e

if __name__ == "__main__" :
    #Service call and display the call result
    print "Show two_int_plus result : %s" %(Plus_client())

```

2. the core part

Here is mainly to explain how to import the custom service message module and use it:

import

```
server :
from learning_server.srv import IntPlus, IntPlusResponse
client :
from learning_server.srv import IntPlus, IntPlusRequest
```

use

```
server:
s = rospy.Service('/Two_Int_Plus', IntPlus, IntPlusCallback)
return IntPlusResponse(req.a + req.b) # Feedback data
client:
response = Plus_client(12, 20) # Request service call, enter request data
return response.result
```

3. run the program

Before running the program, first add executable permissions to the py file

```
sudo chmod a+x IntPlus_server.py
sudo chmod a+x IntPlus_client.py
```

run the program

```
roscore
roslaunch learning_server IntPlus_client.py
roslaunch learning_server IntPlus_server.py
```

4. program operation instructions

What is inconsistent with the C++ version here is that the addend here is set in the program(12 and 20) so after the service is started, the result can be returned immediately.