

Note: This tutorial assumes that you have completed the previous tutorials: examining the simple publisher and subscriber (/ROS/Tutorials/ExaminingPublisherSubscriber).

💡 Please ask about problems and questions regarding this tutorial on answers.ros.org (<http://answers.ros.org>). Don't forget to include in your question the link to this page, the versions of your OS & ROS, and also add appropriate tags.

Writing a Simple Service and Client (C++)

Description: This tutorial covers how to write a service and client node in C++.

Tutorial Level: BEGINNER

Next Tutorial: Examining the simple service and client (/ROS/Tutorials/ExaminingServiceClient)

catkin

roscd

Sommaire

1. Writing a Service Node
 1. The Code
 2. The Code Explained
2. Writing the Client Node
 1. The Code
 2. The Code Explained
3. Building your nodes
4. Building your nodes
5. Running the nodes
 1. Running the Server
 2. Running the Client
6. Video Tutorial

1. Writing a Service Node

Here we'll create the service ("add_two_ints_server") node which will receive two ints and return the sum.

Change directories to your beginner_tutorials package you created in your catkin workspace previous tutorials:

```
roscd beginner_tutorials
```

Please make sure you have followed the directions in the previous tutorial for creating the service needed in this tutorial, creating the AddTwoInts.srv (/ROS/Tutorials/CreatingMsgAndSrv#Creating_a_srv) (be sure to choose the right version of build tool you're using at

the top of wiki page in the link).

1.1 The Code

Create the `src/add_two_ints_server.cpp` file within the `beginner_tutorials` package and paste the following inside it:

Afficher/masquer les numéros de lignes

```
1 #include "ros/ros.h"
2 #include "beginner_tutorials/AddTwoInts.h"
3
4 bool add(beginner_tutorials::AddTwoInts::Request &req,
5          beginner_tutorials::AddTwoInts::Response &res)
6 {
7     res.sum = req.a + req.b;
8     ROS_INFO("request: x=%ld, y=%ld", (long int)req.a, (long int)req.b);
9     ROS_INFO("sending back response: [%ld]", (long int)res.sum);
10    return true;
11 }
12
13 int main(int argc, char **argv)
14 {
15     ros::init(argc, argv, "add_two_ints_server");
16     ros::NodeHandle n;
17
18     ros::ServiceServer service = n.advertiseService("add_two_ints", add);
19     ROS_INFO("Ready to add two ints.");
20     ros::spin();
21
22     return 0;
23 }
```

1.2 The Code Explained

Now, let's break the code down.

Afficher/masquer les numéros de lignes

```
1 #include "ros/ros.h"
2 #include "beginner_tutorials/AddTwoInts.h"
3
```

`beginner_tutorials/AddTwoInts.h` is the header file generated from the `srv` file that we created earlier.

Afficher/masquer les numéros de lignes

```
4 bool add(beginner_tutorials::AddTwoInts::Request &req,
5          beginner_tutorials::AddTwoInts::Response &res)
```

This function provides the service for adding two ints, it takes in the request and response type defined in the srv file and returns a boolean.

Afficher/masquer les numéros de lignes

```
6 {  
7   res.sum = req.a + req.b;  
8   ROS_INFO("request: x=%ld, y=%ld", (long int)req.a, (long int)req.b);  
9   ROS_INFO("sending back response: [%ld]", (long int)res.sum);  
10  return true;  
11 }
```

Here the two ints are added and stored in the response. Then some information about the request and response are logged. Finally the service returns true when it is complete.

Afficher/masquer les numéros de lignes

```
18   ros::ServiceServer service = n.advertiseService("add_two_ints", add);
```

Here the service is created and advertised over ROS.

2. Writing the Client Node

2.1 The Code

Create the src/add_two_ints_client.cpp file within the beginner_tutorials package and paste the following inside it:

Afficher/masquer les numéros de lignes

```
1 #include "ros/ros.h"
2 #include "beginner_tutorials/AddTwoInts.h"
3 #include <cstdlib>
4
5 int main(int argc, char **argv)
6 {
7     ros::init(argc, argv, "add_two_ints_client");
8     if (argc != 3)
9     {
10         ROS_INFO("usage: add_two_ints_client X Y");
11         return 1;
12     }
13
14     ros::NodeHandle n;
15     ros::ServiceClient client = n.serviceClient<beginner_tutorials::AddTw
oInts>("add_two_ints");
16     beginner_tutorials::AddTwoInts srv;
17     srv.request.a = atoll(argv[1]);
18     srv.request.b = atoll(argv[2]);
19     if (client.call(srv))
20     {
21         ROS_INFO("Sum: %ld", (long int)srv.response.sum);
22     }
23     else
24     {
25         ROS_ERROR("Failed to call service add_two_ints");
26         return 1;
27     }
28
29     return 0;
30 }
```

2.2 The Code Explained

Now, let's break the code down.

Afficher/masquer les numéros de lignes

```
15     ros::ServiceClient client = n.serviceClient<beginner_tutorials::AddTw
oInts>("add_two_ints");
```

This creates a client for the `add_two_ints` service. The `ros::ServiceClient` object is used to call the service later on.

Afficher/masquer les numéros de lignes

```
16     beginner_tutorials::AddTwoInts srv;
17     srv.request.a = atoll(argv[1]);
18     srv.request.b = atoll(argv[2]);
```

Here we instantiate an autogenerated service class, and assign values into its request member. A service class contains two members, `request` and `response`. It also contains two class definitions, `Request` and `Response`.


Afficher/masquer les numéros de lignes

```
19     if (client.call(srv))
```

This actually calls the service. Since service calls are blocking, it will return once the call is done. If the service call succeeded, `call()` will return true and the value in `srv.response` will be valid. If the call did not succeed, `call()` will return false and the value in `srv.response` will be invalid.

3. Building your nodes

Again edit the `beginner_tutorials CMakeLists.txt` located at `~/catkin_ws/src/beginner_tutorials/CMakeLists.txt` and add the following at the end:

 https://raw.githubusercontent.com/ros/catkin_tutorials/master/create_package_srvclient/catkin_ws/src/beginner_tutorials/CMakeLists.txt (https://raw.githubusercontent.com/ros/catkin_tutorials/master/create_package_srvclient/catkin_ws/src/beginner_tutorials/CMakeLists.txt)

Afficher/masquer les numéros de lignes

```
27 add_executable(add_two_ints_server src/add_two_ints_server.cpp)
28 target_link_libraries(add_two_ints_server ${catkin_LIBRARIES})
29 add_dependencies(add_two_ints_server beginner_tutorials_gencpp)
30
31 add_executable(add_two_ints_client src/add_two_ints_client.cpp)
32 target_link_libraries(add_two_ints_client ${catkin_LIBRARIES})
33 add_dependencies(add_two_ints_client beginner_tutorials_gencpp)
```

This will create two executables, `add_two_ints_server` and `add_two_ints_client`, which by default will go into package directory of your devel space (`/catkin/workspaces#Development_28Devel.29_Space`), located by default at `~/catkin_ws/devel/lib/<package name>`. You can invoke executables directly or you can use `roslaunch` to invoke them. They are not placed in '`<prefix>/bin`' because that would pollute the `PATH` when installing your package to the system. If you wish for your executable to be on the `PATH` at installation time, you can setup an install target, see: `catkin/CMakeLists.txt` (`/catkin/CMakeLists.txt`)

For more detailed description of the `CMakeLists.txt` (`/catkin/CMakeLists.txt`) file see: `catkin/CMakeLists.txt` (`/catkin/CMakeLists.txt`)

Now run `catkin_make`:

```
# In your catkin workspace
cd ~/catkin_ws
catkin_make
```

If your build fails for some reason:

- make sure you have followed the directions in the previous tutorial: creating the `AddTwoInts.srv` (`/ROS/Tutorials/CreatingMsgAndSrv#Creating_a_srv`).

4. Running the nodes

Running nodes requires you have a ROS core started. Open a new shell, and type:

```
roscore
```

If all goes well, you should see an output that looks something like this:

```
... logging to /u/takayama/.ros/logs/83871c9c-934b-11de-a451-001d927076eb/roslaunch-ads-31831.log
... loading XML file
[/wg/storla/rosbuild/shared_installation/ros/tools/roslaunch/roscore.xml]
Added core node of type [rosout/rosout] in namespace [/]
started roslaunch server http://ads:54367/

SUMMARY
=====

NODES

changing ROS_MASTER_URI to [http://ads:11311/] for starting master locally
starting new master (master configured for auto start)
process[master]: started with pid [31874]
ROS_MASTER_URI=http://ads:11311/
setting /run_id to 83871c9c-934b-11de-a451-001d927076eb
+PARAM [/run_id] by /roslaunch
+PARAM [/roslaunch/uris/ads:54367] by /roslaunch
process[rosout-1]: started with pid [31889]
started core service [/rosout]
+SUB [/time] /rosout http://ads:33744/
+SERVICE [/rosout/get_loggers] /rosout http://ads:33744/
+SERVICE [/rosout/set_logger_level] /rosout http://ads:33744/
+PUB [/rosout_agg] /rosout http://ads:33744/
+SUB [/rosout] /rosout http://ads:33744/
```

Now everything is set to run server and client.

4.1 Running the Server

Start by running the server. Open a new shell and type:

```
roslaunch beginner_tutorials add_two_ints_server
```

You should see something similar to:

```
Ready to add two ints.
```

4.2 Running the Client

Now let's run the client with the necessary arguments, in another shell:

```
$ rosrun beginner_tutorials add_two_ints_client 1 3
```

In the client's shell, you should see something similar to:

```
Sum: 4
```

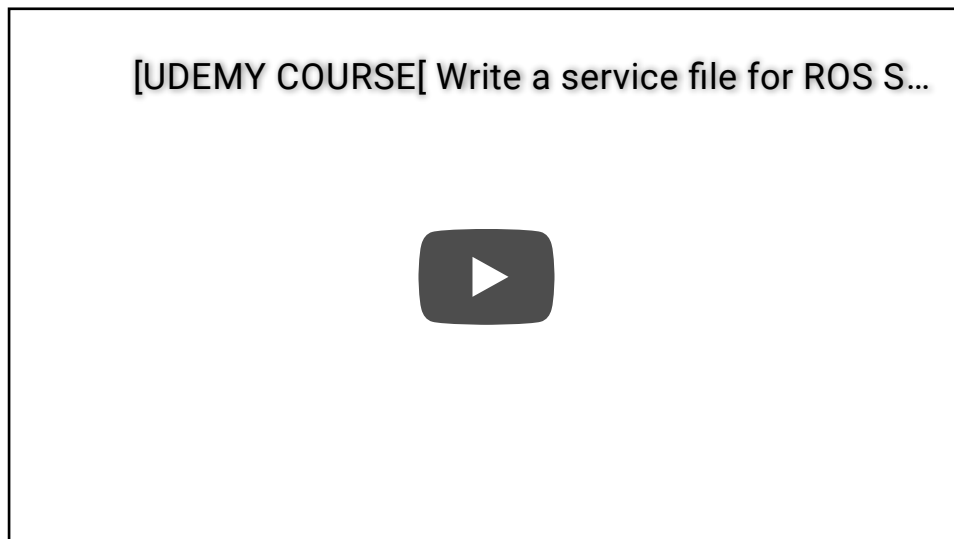
In the server's shell, instead, you should see something similar to:

```
request: x=1, y=3  
sending back response: [4]
```

Now that you have written a simple service and client, let's examine the simple service and client (/ROS/Tutorials/ExaminingServiceClient).

5. Video Tutorial

The following video presents a small tutorial on ROS services.



Except where
otherwise noted,

Wiki: ROS/Tutorials/WritingServiceClient(c++) (dernière édition le 2019-07-18 19:13:24 par AnisKoubaa (/AnisKoubaa))

the ROS wiki is licensed under the

Creative Commons Attribution 3.0 (<http://creativecommons.org/licenses/by/3.0/>)

Brought to you by:  Open Source Robotics Foundation

(<http://www.osrfoundation.org>)