ROS: services

Services in `ROS` provide a synchronous communication mechanism between two nodes: one offers a service (server), and the other calls it (client). Unlike topics (which stream data continuously), services are used for request-response interactions, ideal when a one-time action or data exchange is needed.

> Along the following tutorial, the `Linux` `gedit` text editor will be used for adding or modifying text file. It is not mandatory to use this editor, feel free to chose the one of preference (`vim` , `nano` , etc.).

Important concepts

- Synchronous: the client waits for the server to respond before continuing.
- Two-way communication: clients send a request, and servers return a response.
- Defined structure: services use `.srv` files to define input and output formats.

Service structure

A service is defined by:

- A request part (inputs).
- A response part (outputs).

When to use services

Use services when:

- An action needs to be triggered once (e.g., reset a sensor, move to a specific pose).
- A direct result or response is expected.
- Continuous streaming data is not needed (use topics for that).

Steps to follow:

> `Catkin` is a build system for the Robot Operating System (`ROS`) that extends `CMake` to manage dependencies between packages. It simplifies the development process for large, complex, and heterogeneous code ecosystems, making it easier to integrate various programming languages and tools within a single workspace.

1. Create the package

From the home directory `(~)`, run:

```
mkdir catkin_ws
cd catkin_ws
mkdir src
```

This creates the directories `catkin_ws` and `src` .

From `~/catkin_ws/src` , run:

```
catkin_create_pkg my_service_pkg rospy std_msgs
```

This creates the `my_service_pkg` folder package with basic dependencies.

---

2. Create the `srv` folder and the service file

In the newly created package folder:

```
cd my_service_pkg
mkdir srv
gedit srv/AddTwoInts.srv
```

Copy and paste this into the `AddTwoInts.srv` file:

```
int64 a
int64 b
---
int64 sum
```

Save and close.

Mind that when a `*.srv` file is created, there is an automatically created response to that method, so if there is an `AddTwoInts.srv`, there will be two classes associated to it: `AddTwoInts` and `AddTwoIntsResponse`. The second class is in charge of handling the response of the message.

When defining the `AddTwoInts.srv`, it is necessary to know:

- Everything **above** the `---` is part of the **Request** message.
- Everything **below** the `---` is part of the **Response** message.

---

3. Edit `CMakeLists.txt`

Open the package's `CMakeLists.txt`:

```
gedit CMakeLists.txt
```

Make sure it has the following lines (edit if necessary):

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

add_service_files(
  FILES
  AddTwoInts.srv
)

generate_messages(
  DEPENDENCIES
  std_msgs
)

catkin_package(
  CATKIN_DEPENDS rospy std_msgs message_runtime
)
```

4. Edit `package.xml`

Open `package.xml` and make sure to have these lines (or add them):

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

5. Create the server and client scripts

Inside `my_service_pkg`, create a `scripts` folder:

```
mkdir scripts
cd scripts
gedit add_two_ints_server.py
```

The server:

> ⚠ Make sure to have the correct version of `Python` on the first code line or it will not work. In this cased, `Python v3.X` has been selected as the preferred option as stated in the first line of the following scripts.

```
#!/usr/bin/env python3

from my_service_pkg.srv import AddTwoInts, AddTwoIntsResponse
import rospy

def handle_add_two_ints(req):
    print(f"Adding {req.a} + {req.b}")
    return AddTwoIntsResponse(req.a + req.b)

def server():
    rospy.init_node('add_two_ints_server')
    s = rospy.Service('add_two_ints', AddTwoInts, handle_add_two_ints)
    print("Server ready.")
    rospy.spin()

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

The client:

```
gedit add_two_ints_client.py
```

Paste this:

```
#!/usr/bin/env python3

import rospy
import sys
from my_service_pkg.srv import AddTwoInts

def client(x, y):
    rospy.init_node('add_two_ints_client')
    rospy.wait_for_service('add_two_ints')
    try:
        add_two_ints = rospy.ServiceProxy('add_two_ints', AddTwoInts)
```

```
        resp = add_two_ints(x, y)
        print(f"Result: {resp.sum}")
    except rospy.ServiceException as e:
        print(f"Error: {e}")

if __name__ == "__main__":
    if len(sys.argv) != 3:
        print("Please pass two numbers as arguments.")
        sys.exit(1)
    x = int(sys.argv[1])
    y = int(sys.argv[2])
    client(x, y)
```

Make both executables by adding execution permissions:

```
chmod +x add_two_ints_*.py
```

---

6. Go back to `catkin_ws` and compile

```
cd ~/catkin_ws
catkin_make
source devel/setup.bash
```

---

7. Run in 3 terminals

1. The first terminal will run the `ROS` daemon so there will be a process in the background listening to create the required `ROS` structure (server, client, services, topics, etc.).
2. The second terminal will execute the server which will be waiting for a signal from a client to execute the code. In this case, the code consists in calculate the `sum` of `a` and `b`, which are the expected parameters set in the message passed to the server.
3. The third terminal will be used as a client, signalling the server to execute the operation and set the content in the expected message format (fill out the `sum` variable).

Terminal 1 (roscore):

```
roscore

... logging to /root/.ros/log/0e56c18a-3177-11f0-ab77-c5f36d6c7d6d/roslaunch-HP-190-935.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://HP-190:42139/
ros_comm version 1.17.0


SUMMARY
========

PARAMETERS
 * /rosdistro: noetic
 * /rosversion: 1.17.0

NODES
```

```
auto-starting new master
process[master]: started with pid [945]
ROS_MASTER_URI=http://HP-190:11311/

setting /run_id to 0e56c18a-3177-11f0-ab77-c5f36d6c7d6d
process[rosout-1]: started with pid [955]
started core service [/rosout]
```

Terminal 2 (server):

```
source ~/catkin_ws/devel/setup.bash
rosrun my_service_pkg add_two_ints_server.py
Server ready.
Adding 10 + 14
```

Terminal 3 (client):

```
source ~/catkin_ws/devel/setup.bash
rosrun my_service_pkg add_two_ints_client.py 10 14
Result: 24
```