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 (Python)

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

Tutorial Level: BEGINNER

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

catkin | rosbuild

Sommaire

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 directory into the beginner_tutorials package, you created in the earlier tutorial, creating a package (/ROS/Tutorials/CreatingPackage%3Fbuildsystem%3Dcatkin):

```
$ 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 **scripts/add_two_ints_server.py** file within the beginner_tutorials package and paste the following inside it:

Afficher/masquer les numéros de lignes

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

from beginner_tutorials.srv import AddTwoInts,AddTwoIntsResponse
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()
```

Don't forget to make the node executable:

```
chmod +x scripts/add_two_ints_server.py
```

1.2 The Code Explained

Now, let's break the code down.

There's very little to writing a service using rospy (/rospy). We declare our node using `init_node()` and then declare our service:

Afficher/masquer les numéros de lignes

```
    s = rospy.Service('add_two_ints', AddTwoInts, handle_add_two_ints)
```

This declares a new service named `add_two_ints` with the `AddTwoInts` service type. All requests are passed to `handle_add_two_ints` function. `handle_add_two_ints` is called with instances of `AddTwoIntsRequest` and returns instances of `AddTwoIntsResponse`.

Just like with the subscriber example, `rospy.spin()` keeps your code from exiting until the service is shutdown.

2. Writing the Client Node

2.1 The Code

Create the **scripts/add_two_ints_client.py** file within the beginner_tutorials package and paste the following inside it:

Afficher/masquer les numéros de lignes

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

import sys
import rospy
from beginner_tutorials.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]

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))
```

Don't forget to make the node executable:

```
$ chmod +x scripts/add_two_ints_client.py
```

2.2 The Code Explained

Now, let's break the code down.

The client code for calling services is also simple. For clients you don't have to call `init_node()`. We first call:

Afficher/masquer les numéros de lignes

```python
    rospy.wait_for_service('add_two_ints')
```

This is a convenience method that blocks until the service named `add_two_ints` is available. Next we create a handle for calling the service:

Afficher/masquer les numéros de lignes

```
        add_two_ints = rospy.ServiceProxy('add_two_ints', AddTwoInts)
```

We can use this handle just like a normal function and call it:

Afficher/masquer les numéros de lignes

```
        resp1 = add_two_ints(x, y)
        return resp1.sum
```

Because we've declared the type of the service to be `AddTwoInts`, it does the work of generating the `AddTwoIntsRequest` object for you (you're free to pass in your own instead). The return value is an `AddTwoIntsResponse` object. If the call fails, a `rospy.ServiceException` may be thrown, so you should setup the appropriate `try/except` block.

3. Building your nodes

We use CMake as our build system and, yes, you have to use it even for Python nodes. This is to make sure that the autogenerated Python code for messages and services (http://www.ros.org/wiki/ROS/Tutorials/CreatingMsgAndSrv#Creating_a_srv) is created.

Go to your catkin workspace and run `catkin_make`.

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

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

4. Video Tutorial

The following video presents a small tutorial on ROS services

Except where otherwise noted,