理解 ROS 服务和参数

AI 航 团队

1. ROS Service

本教程假设从前一教程启动的 turtlesim_node 仍在运行，现在我们来看看 turtlesim 提供了什么服务：

服务（services）是节点之间通讯的另一种方式。服务允许节点发送请求（request） 并获得一个响应（response）

2. 使用 rosservice

rosservice 可以很轻松的使用 ROS 客户端/服务器框架提供的服务。rosservice 提供了很多可以在 topic 上使用的命令，如下所示：

```
rosservice list      输出可用服务的信息
rosservice call      调用带参数的服务
rosservice type      输出服务类型
rosservice find      依据类型寻找服务 find services by service type
rosservice uri       输出服务的 ROSRPC uri
```

2.1 rosservice list

```
$ rosservice list
```

list 命令显示 turtlesim 节点提供了 9 个服务：重置（reset）, 清除（clear）, 再生（spawn）, 终（kill）, turtle1/set_pen, /turtle1/teleport_absolute, /turtle1/teleport_relative, turtlesim/get_loggers, and turtlesim/set_logger_level. 同时还有另外两个 rosout 节点提供的服务: /rosout/get_loggers and /rosout/set_logger_level.

```
yt@yt-UNO-2483G-453AE:~$ rosservice list
/clear
/kill
/my_turtle/get_loggers
/my_turtle/set_logger_level
/reset
/rosout/get_loggers
/rosout/set_logger_level
/rostopic_7955_1587957076060/get_loggers
/rostopic_7955_1587957076060/set_logger_level
/rostopic_8943_1587958496697/get_loggers
/rostopic_8943_1587958496697/set_logger_level
/rqt_gui_py_node_7783/get_loggers
/rqt_gui_py_node_7783/set_logger_level
/rqt_gui_py_node_9256/get_loggers
/rqt_gui_py_node_9256/set_logger_level
/spawn
/teleop_turtle/get_loggers
/teleop_turtle/set_logger_level
/turtle1/set_pen
/turtle1/teleport_absolute
/turtle1/teleport_relative
```

2.2 rosservice type

我们使用 rosservice type 命令更进一步查看 clear 服务:使用方法如下：

```
$ rosservice type  clear
```

```
/turtle1/teleport_relative
yt@yt-UNO-2483G-453AE:~$ rosservice type clear
std_srvs/Empty
yt@yt-UNO-2483G-453AE:~$ rosservice call clear
```

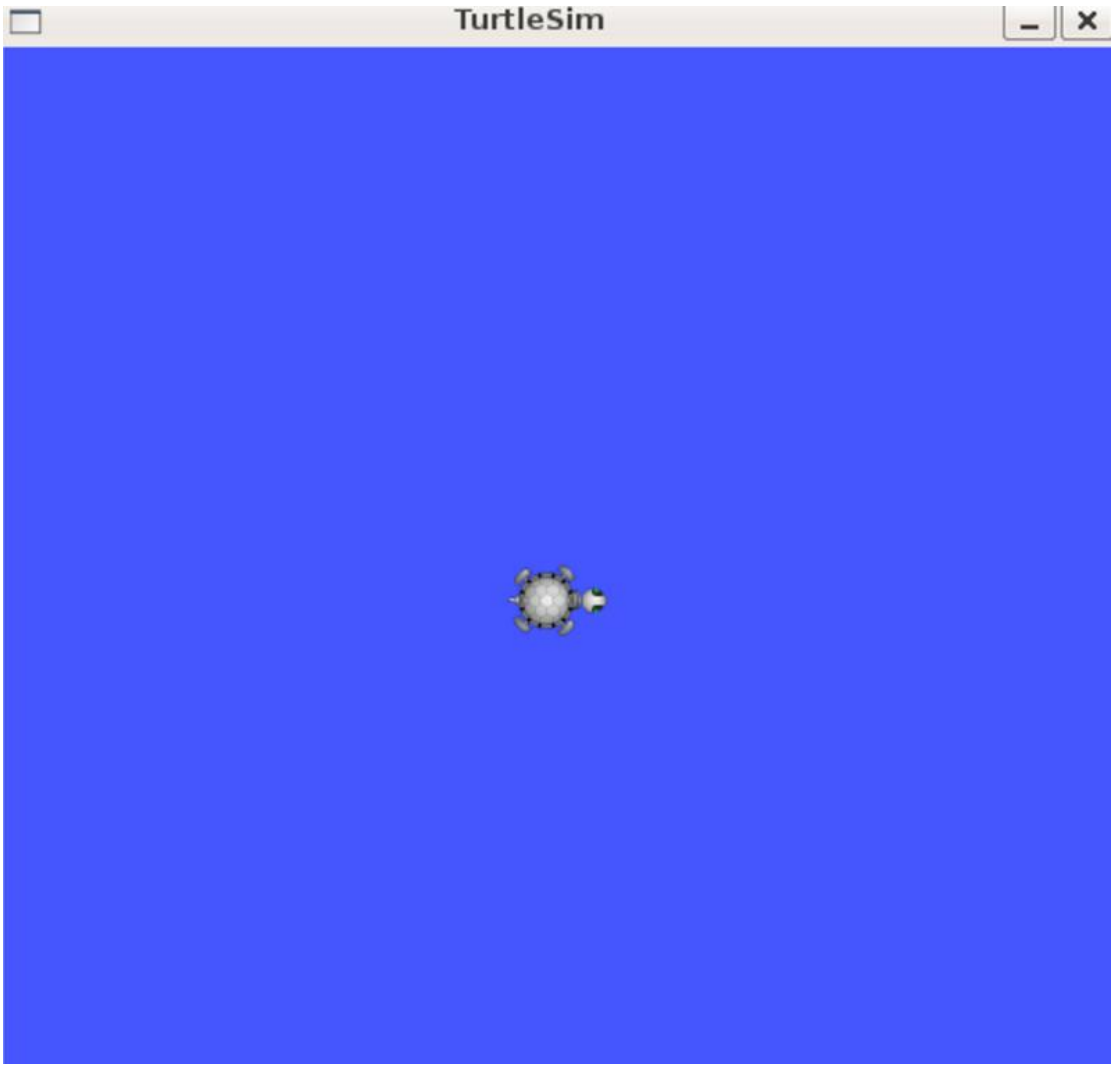
服务的类型为空（empty)，这表明在调用这个服务是不需要参数（比如，请求不需要发送数据，响应也没有数据）。下面我们使用 rosservice call 命令调用服务：

2.3 rosservice call

使用方法：

```
std_srvs/Empty
yt@yt-UNO-2483G-453AE:~$ rosservice call clear

yt@yt-UNO-2483G-453AE:~$ rosservice type spawn|rossrv show
float32 x
float32 y
float32 theta
string name
---
string name

yt@yt-UNO-2483G-453AE:~$ rosservice call spawn 2 2 0.2 ""
```

正如我们所期待的，服务清除了 turtlesim_node 的背景上的轨迹。



通过查看再生（spawn）服务的信息，我们来了解带参数的服务:

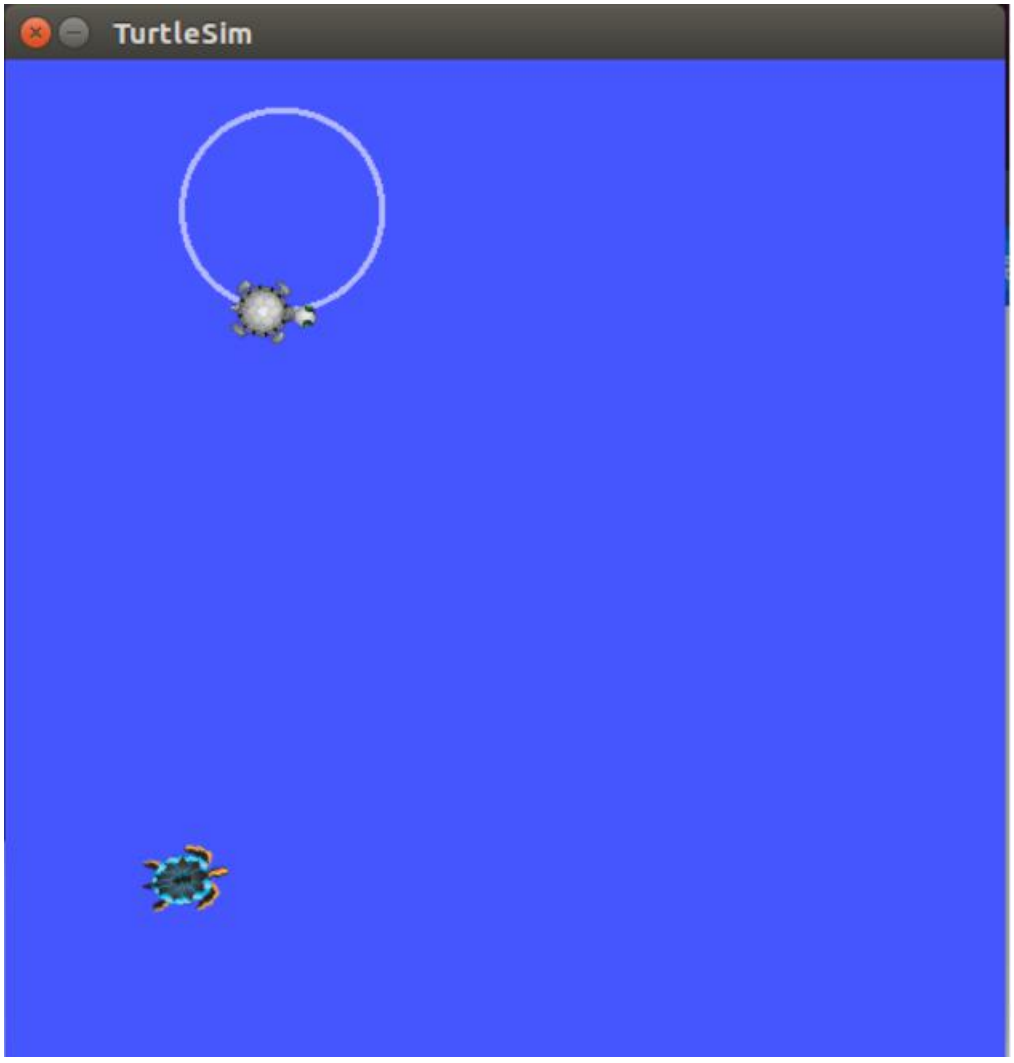
```
$ rosservice type spawn| rossrv show
```

```
std_srvs/Empty
yt@yt-UNO-2483G-453AE:~$ rosservice call clear

yt@yt-UNO-2483G-453AE:~$ rosservice type spawn|rossrv show
float32 x
float32 y
float32 theta
string name
---
string name

yt@yt-UNO-2483G-453AE:~$ rosservice call spawn 2 2 0.2 ""
```

这个服务使得我们可以在给定的位置和角度生成一只新的乌龟。名字参数是可选的，这里我们不设具体的名字，让 turtlesim 自动创建一个。

```
$ rosservice call spawn 2 2 0.2 ""
```

```
yt@yt-UNO-2483G-453AE:~$ rosservice call spawn 2 2 0.2 ""
name: "turtle2"
```



3. Using rosparam

rosparam 使得我们能够存储并操作 ROS 参数服务器（Parameter Server）上的数据。参数服务器能够存储整型、浮点、布尔、字符串、字典和列表等数据类型。rosparam 使用 YAML 标记语言的语法。一般而言，YAML 的表述很自然：1 是整型, 1.0 是浮点型, one 是字符串, true 是布尔, [1, 2, 3]是整型列表, {a: b, c: d}是字典. rosparam 有很多指令可以用来操作参数，如下所示:

使用方法:

```
rosparam set          设置参数
rosparam get          获取参数
rosparam load         从文件读取参数
rosparam dump         向文件中写入参数
rosparam delete       删除参数
rosparam list         列出参数名
```

我们来看看现在参数服务器上都有哪些参数：

3.1 rosparam list

```
$ rosparam list
```

我们可以看到 turtlesim 节点在参数服务器上有 3 个参数用于设定背景颜色：

```
/background_b
/background_g
/background_r
/roslaunch/uris/aqy:51932
/run_id
```

3.2 rosparam set and rosparam get

现在我们修改背景颜色的红色通道：

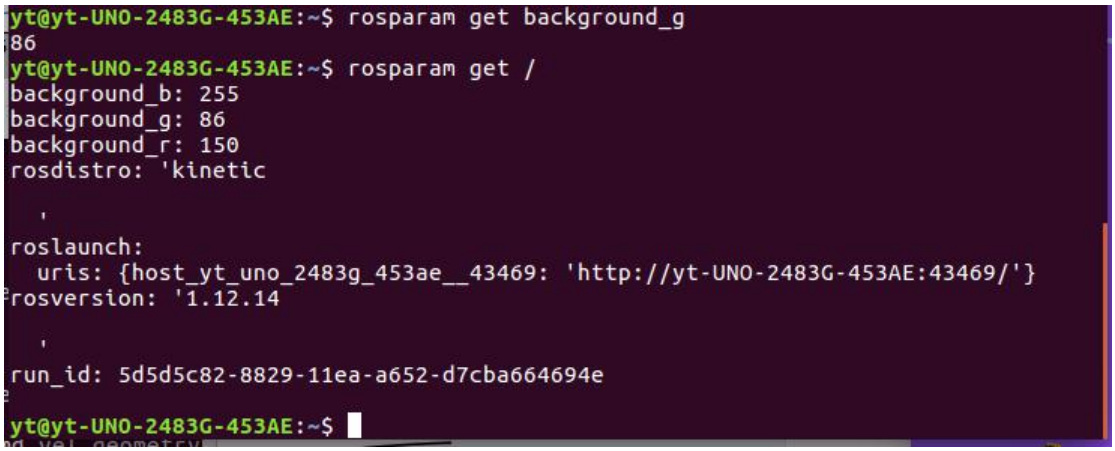
```
$ rosparam set background_r 150
```

上述指令修改了参数的值，现在我们调用清除服务使得修改后的参数生效：

```
$ rosservice call clear
```

现在 我们的小乌龟看起来应该是像这样：

现在我们来查看参数服务器上的参数值——获取背景的绿色通道的值：
我们可以使用 rosparam get /来显示参数服务器上的所有内容：



3.3 rosparam dump and rosparam load

你可能希望存储这些信息以备今后重新读取。这通过 rosparam 很容易就可以实现: 现在我们将所有的参数写入 params.yaml 文件：

```
$ rosparam dump params.yaml
```

你甚至可以将 yaml 文件重载入新的命名空间，比如说 copy 空间:

```
$ rosparam load params.yaml copy
$ rosparam get copy/background_b
```

```
255
```