ROS Get Started Guide

Youngjoon Han
young@ssu.ac.kr

Service

- Service
  - Service Server, Client 예제 작성
- Action
  - Action Server, Client 예제 작성
- Parameter
  - Getparam, setparam 예제 작성
- Launch
  - launch 예제 작성
- 수업과제
  - parameter 예제 class로 수정하기

1. Service

- service_tutorials 패키지 생성

```
## catkin_create_pkg service_tutorials
```

- srv 파일 생성

```
## roscd service_tutorials
## mkdir srv
## vim SrvTutorials.srv
```

```
1   int64 a
2   int64 b
3   ---
4   int64 result
```

1. Service

- service client 코드 작성

```
#include "ros/ros.h"
#include "service_tutorials/SrvTutorial.h"

int main(int argc, char *argv[]){
  ros::init(argc, argv, "service_client");

  if(argc != 3){
    ROS_INFO("cmd : rosrun ros_tutorial_service service_client arg0 arg1");
    ROS_INFO("arg0 : double number, arg1 : double number");
    return 1;
  }

  ros::NodeHandle nh;

  ros::ServiceClient service_client =
    nh.serviceClient<service_tutorials::SrvTutorial>("ros_tutorial_srv");

```

1. Service

- service client 코드 작성

```
18    service_tutorials::SrvTutorial srv;
19
20    srv.request.a = atoll(argv[1]);
21    srv.request.b = atoll(argv[2]);
22
23    if(service_client.call(srv)){
24      ROS_INFO("send srv, srv.Request.a and b : %ld, %ld",static_cast<long int>(srv.request.a),
25        static_cast<long int>(srv.request.b));
26      ROS_INFO("receive srv, srv.Response.result: %ld", (long int)srv.response.result);
27    }
28    else{
29      ROS_ERROR("Faild to call service ros_tutorial_srv");
30      return 1;
31    }
32    return 0;
33  }
```

1. Service

- service server 코드 작성

```cpp
#include "ros/ros.h"
#include "service_tutorials/SrvTutorial.h"


bool calculation(service_tutorials::SrvTutorial::Request &req,
  service_tutorials::SrvTutorial::Response &res){
    res.result = req.a + req.b;

    ROS_INFO("request: x=%ld, y=%ld",static_cast<long int>(req.a), static_cast<long int>(req.b));
    ROS_INFO("sending back response : %ld",static_cast<long int>(res.result));

    return true;
  }

int main(int argc, char *argv[]){
  ros::init(argc, argv, "service_server");
  ros::NodeHandle nh;

  ros::ServiceServer service_server = nh.advertiseService("service_tutorial", calculation);

  ROS_INFO("ready srv server!");
  ros::spin();
}
```

1. Service

- CMakeLists.txt 수정

```
cmake_minimum_required(VERSION 2.8.3)
project(service_tutorials)

find_package(catkin REQUIRED)

find_package(catkin REQUIRED COMPONENTS
  roscpp
  message_generation
)
add_service_files(FILES
  SrvTutorial.srv
)

generate_messages(DEPENDENCIES
  std_msgs
)

catkin_package(
  CATKIN_DEPENDS
  message_runtime
)

include_directories(
  include ${catkin_INCLUDE_DIRS}
)
```

1. Service

- CMakeLists.txt 수정

```
27  add_executable(service_server
28    src/service_server.cpp
29  )
30  add_dependencies(service_server ${${PROJECT_NAME}_EXPORTED_TARGETS}
31  ${catkin_EXPORTED_TARGETS})
32  target_link_libraries(service_server
33    ${catkin_LIBRARIES}
34  )
35
36  add_executable(service_client
37    src/service_client.cpp
38  )
39  add_dependencies(service_client ${${PROJECT_NAME}_EXPORTED_TARGETS}
40  ${catkin_EXPORTED_TARGETS})
41  target_link_libraries(service_client
42    ${catkin_LIBRARIES}
43  )
```

1. Service

- package.xml 수정

```xml
<?xml version="1.0"?>
<package format="2">
  <name>service_tutorials</name>
  <version>0.0.0</version>
  <description>The service_tutorials package</description>

  <maintainer email="whiteherv@todo.todo">whiteherv</maintainer>

  <license>TODO</license>

  <buildtool_depend>catkin</buildtool_depend>
  <export>
  </export>
<build_depend>message_generation</build_depend>
<build_export_depend>message_generation</build_export_depend>
<exec_depend>message_runtime</exec_depend>
</package>
```

1. Service

- 서비스 서버 실행

```
## rosrun service_tutorials service_server
```

- 서비스 클라이언트 실행(서비스 요청)

```
## rosrun service_tutorials service_client 실수1 실수2
```

2. action

- action_tutorials 패키지 생성

```
## catkin_create_pkg action_tutorials
```

- action 파일 생성

```
## roscd action_tutorials
## mkdir action
## vim fibonacci.action
```

```
#goal definition
int32 order
---
#result definition
int32[] sequence
---
#feedback
int32[] sequence
```

2. action

- action client 코드 작성

```cpp
#include "ros/ros.h"
#include "actionlib/client/simple_action_client.h"
#include "actionlib/client/terminal_state.h"
#include "action_tutorials/fibonacciAction.h"

int main(int argc, char **argv){
  ros::init(argc, argv, "action_client");
  actionlib::SimpleActionClient<action_tutorials::fibonacciAction> ac("action_tutorials", true);

  ROS_INFO("Waiting for action server to start.");
  ac.waitForServer();

  ROS_INFO("Action server started, sending goal.");
  action_tutorials::fibonacciGoal goal;
  goal.order = 20;
  ac.sendGoal(goal);
```

2. action

- action client 코드 작성

```
18    bool finished_before_timeout = ac.waitForResult(ros::Duration(30.0));
19
20    if(finished_before_timeout)
21    {
22      actionlib::SimpleClientGoalState state = ac.getState();
23      ROS_INFO("Action finished : %s",state.toString().c_str());
24    }
25    else {
26      ROS_INFO("Actiondid not finish before the time out.");
27    }
28
29    return 0;
30  }
```

2. action

- action server 코드 작성

```
#include "ros/ros.h"
#include "actionlib/server/simple_action_server.h"
#include "action_tutorials/fibonacciAction.h"

class FibonacciAction{
protected:
  ros::NodeHandle nh_;

  actionlib::SimpleActionServer<action_tutorials::fibonacciAction> as_;

  std::string action_name_;

  action_tutorials::fibonacciFeedback feedback_;
  action_tutorials::fibonacciResult result_;

public:
  FibonacciAction(std::string name):
  as_(nh_, name, boost::bind(&FibonacciAction::executeCB, this, _1),false),
    action_name_(name)
  {
        as_.start();
  }
  ~FibonacciAction(void){}
```

2. action

- action server 코드 작성

```
25    void executeCB(const action_tutorials::fibonacciGoalConstPtr &goal)
26    {
27        ros::Rate r(1);
28        bool success = true;
29
30        feedback_.sequence.clear();
31        feedback_.sequence.push_back(0);
32        feedback_.sequence.push_back(1);
33
34        ROS_INFO("%s : Executing, creating fibonacci sequence of order %i with seeds %i, %i",
35          action_name_.c_str(), goal->order, feedback_.sequence[0], feedback_.sequence[1]);
36
37        for(int i=1; i <= goal->order; ++i){
38          if(as_.isPreemptRequested() || !ros::ok()){
39            ROS_INFO("%s: Preempted", action_name_.c_str());
40            as_.setPreempted();
41            success = false;
42            break;
43          }
44
45          feedback_.sequence.push_back(feedback_.sequence[i] + feedback_.sequence[i-1]);
46          as_.publishFeedback(feedback_);
47          r.sleep();
48        }
```

2. action

- action server 코드 작성

```
49          if(success){
50            result_.sequence = feedback_.sequence;
51            ROS_INFO("%s: Secceeded", action_name_.c_str());
52            as_.setSucceeded(result_);
53          }
54      }
55  };
56
57  int main(int argc, char **argv){
58    ros::init(argc, argv, "action_server");
59    FibonacciAction fibonacci("action_tutorials");
60    ros::spin();
61    return 0;
62  }
```

2. action

- CMakeLists.txt 수정

```
1  cmake_minimum_required(VERSION 2.8.3)
2  project(action_tutorials)
3
4  find_package(catkin REQUIRED)
5  find_package(catkin REQUIRED COMPONENTS
6    roscpp
7    actionlib
8    actionlib_msgs
9    message_generation
10 )
11 add_action_files(FILES
12   fibonacci.action
13 )
14
15 generate_messages(DEPENDENCIES
16   actionlib_msgs
17   std_msgs
18 )
19 catkin_package(
20   CATKIN_DEPENDS
21   message_runtime
22 )
```

2. action

- CMakeLists.txt 수정

```
23
24  include_directories(
25    include ${catkin_INCLUDE_DIRS}
26  )
27  add_executable(action_client
28    src/action_client.cpp
29  )
30  add_dependencies(action_client ${${PROJECT_NAME}_EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})
31  target_link_libraries(action_client
32    ${catkin_LIBRARIES}
33  )
34
35  add_executable(action_server
36    src/action_server.cpp
37  )
38  add_dependencies(action_server ${${PROJECT_NAME}_EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})
39  target_link_libraries(action_server
40    ${catkin_LIBRARIES}
41  )
```

2. action

- package.xml 수정

```
1  <?xml version="1.0"?>
2  <package format="2">
3    <name>action_tutorials</name>
4    <version>0.0.0</version>
5    <description>The action_tutorials package</description>
6
7    <maintainer email="whiteherv@todo.todo">whiteherv</maintainer>
8
9    <license>TODO</license>
10
11   <buildtool_depend>catkin</buildtool_depend>
12
13   <export>
14
15   </export>
16 <build_depend>message_generation</build_depend>
17 <build_export_depend>message_generation</build_export_depend>
18 <exec_depend>message_runtime</exec_depend>
19 </package>
```

3. Parameter

- 실습 전 ~/catkin_ws/src에서 service_tutorials 폴더를 parameter_tutorials 이름으로 복사한다.
- package 이름 정보를 parameter_tutorials에 맞게 수정한다.
  - package.xml
  - CMakeLists.txt
- service_client 노드 이름을 paramter_client로 수정한다.
  - cpp 파일 이름 수정, 코드 내 ros::init 코드 수정, 코드 내 헤더파일 수정
  - CMakeLists.txt 하단 add_executable, add_dependency 수정

3. Parameter

- 수정된 클라이언트 코드

```cpp
#include "ros/ros.h"
#include "parameter_tutorials/SrvTutorial.h"

int main(int argc, char *argv[]){
  ros::init(argc, argv, "parameter_client");

  if(argc != 3){
    ROS_INFO("cmd : rosrun service_tutorials service_client arg0 arg1");
    ROS_INFO("arg0 : double number, arg1 : double number");
    return 1;
  }

  ros::NodeHandle nh;

  ros::ServiceClient parameter_client =
    nh.serviceClient<parameter_tutorials::SrvTutorial>("parameter_tutorial");

  parameter_tutorials::SrvTutorial srv;

  srv.request.a = atoll(argv[1]);
  srv.request.b = atoll(argv[2]);

  if(parameter_client.call(srv)){
    ROS_INFO("send srv, srv.Request.a and b : %ld, %ld",static_cast<long int>(srv.request.a),
      static_cast<long int>(srv.request.b));
    ROS_INFO("receive srv, srv.Response.result: %ld", (long int)srv.response.result);
  }
  else{
    ROS_ERROR("Faild to call service ros_tutorial_srv");
    return 1;
  }
  return 0;
}
```

3. Parameter

- service_server 노드 이름을 paramter_server로 수정한다.
  - cpp 파일 이름 수정, 코드 내 ros::init 코드 수정
  - CMakeLists.txt 하단 add_executable, add_dependency 수정
- parameter_server노드가 "caculation_method" 파라미터 값을 참조해 service_server와는 달리 +, -, *, -를 수행하도록 코드를 수정한다.

3. Parameter

- 수정된 서버 코드

```
#include "ros/ros.h"
#include "parameter_tutorials/SrvTutorial.h"


enum {PLUS=1, MINUS, MULTIPLICATION, DIVISION};

int g_operator = PLUS;

bool calculation(parameter_tutorials::SrvTutorial::Request &req,
  parameter_tutorials::SrvTutorial::Response &res){
    switch(g_operator){
      case PLUS:
        res.result = req.a + req.b; break;
      case MINUS:
        res.result = req.a - req.b; break;
      case MULTIPLICATION:
        res.result = req.a * req.b; break;
      case DIVISION:
        res.result = (req.b == 0) ? 0 : req.a / req.b; break;
      default:
        ROS_WARN("UNKNOWN OPERATOR TYPE. It must be 1, 2, 3 ,4. The value of operator is %d", g_operator);
        break;
    }

      ROS_INFO("request: x=%ld, y=%ld",static_cast<long int>(req.a), static_cast<long int>(req.b));
      ROS_INFO("sending back response : %ld",static_cast<long int>(res.result));
    return true;
}
```

3. Parameter

- 수정된 서버 코드

```
30  int main(int argc, char *argv[]){
31    ros::init(argc, argv, "parameter_server");
32    ros::NodeHandle nh;
33
34    nh.setParam("calculation_method", g_operator);
35    ros::ServiceServer parameter_server = nh.advertiseService("parameter_tutorial", calculation);
36
37    ROS_INFO("ready srv server!");
38    ros::Rate loop_rate(10);
39    while(ros::ok()){
40      nh.getParam("calculation_method", g_operator);
41      ros::spinOnce();
42      loop_rate.sleep();
43    }
44    return 0;
45  }
```

4. launch

- parameter_tutorials 패키지로 디렉토리 변경

```
## roscd parameter_tutorials
```

- 런치파일 생성

```
## mkdir launch
## cd launch
## vim sample_launch.launch
```

```
<launch>
    <node pkg="parameter_tutorials" type="parameter_server" name="parameter_server" output="screen"/>
    <param name="calculation_method" value="1"/>

    <node pkg="beginner_tutorials" type="talker" name="talker"/>
    <node pkg="beginner_tutorials" type="listener" name="listener"/>
</launch>
```

5. 수업과제

parameter 예제 class로 수정하기

- parameter_tutorials 패키지의 parameter_server.cpp 노드를 다음 조건에 맞춰 re-writing하라.
  - 노드의 입력과 출력은 기존과 같다.
  - 메인함수의 loop문을 삭제하고 대신 ros::spin()을 호출한다.
  - int g_operator를 삭제한다.
  - 새로운 클래스 SimpleCalculator를 정의하며, 적절한 메소드를 정의해 기존 서버의 역할을 수행하도록 한다. 이때 콜백 함수가 호출될때마다 “calculation_method”파라미터 값을 읽어 서버의 동작을 수행하도록 한다.