Understanding ROS 2 Topics

Estimated time to completion: **15 minutes**

3.8 How to Create a Custom Interface

It is always recommended to use the `interfaces` that ROS2 already provides. Remember you can check all ROS2 available `interfaces` using the **`ros2 interface list`**) command. However, if none fits your needs, you can create a new one.

Create a new package named **`custom_interfaces`** . This package, however, has to be a **`CMake` package**. Currently, there is no way to generate custom interfaces in a pure Python package. However, you can create a custom `interface` in a CMake package and then use it in a Python node.

▶ Execute in Terminal #1

In [ ]:

```
cd ~/ros2_ws/src
```

In [ ]:

```
ros2 pkg create --build-type ament_cmake custom_interface
```

- Notes -

Now, specify that this is a CMake package with the **`--build-type`** flag set to **`ament_cmake`** .

- End of Notes -

MISSION 4: Create a `topic` `interface` message for information about locations of detections

- **NASA** has asked us that our Mars rover has to publish information about the objects detected, issues that happened, actions taken... All of this information has to be associated with a location.
- Therefore we need two fields:
  - `info` : This will be a `std_msgs/msg/String` that documents anything we want.
  - `rover_location` : This will be a message of type `geometry_msgs/Pose` extracted from the `/odom` on the moment of the information generation.
- For example:
  - For example: If the Mars rover detects a plant in a certain location, the `interface` would be something like this:

```
info = "Detected plant, detected as tropical_plant"
rover_location =
    position =
           x = 0.6
           y = 2.3
           z = 0.1
    orientation =
           x 0
           y 0
           z 0
           w 1
```

1. Create a directory

Create a directory `msg` in your package.

In [ ]:

```
cd ~/ros2_ws/src/custom_interfaces
```

In [ ]:

```
mkdir msg
touch msg/RoverEvents.msg
```

2. Create a topic message file

The `RoverEvents.msg` file must contain the following:

RoverEvents.msg

In [ ]:

```
std_msgs/String info # Info about discovery, issue, actio
geometry_msgs/Pose rover_location # Location of the mars
```

3. Modify the CMakeLists.txt file:

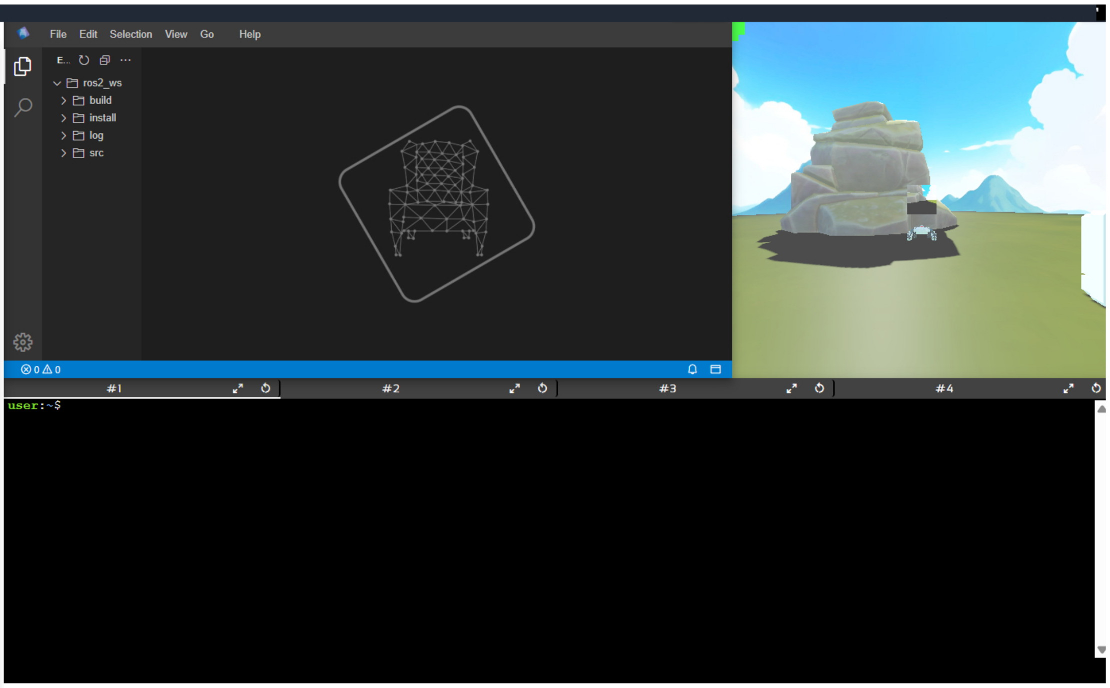
You have to edit two functions inside `CMakeLists.txt` :

`find_package()`

This is where all the packages required to **COMPILE** the messages for the `topics` , services, and actions go. In `package.xml` , state them as **`build_depend`** and **`exec_depend`** .

In [ ]:

```
find_package(ament_cmake REQUIRED)
find_package(rclcpp REQUIRED)
find_package(std_msgs REQUIRED)
```



```
user:~$
```

```
find_package(geometry_msgs REQUIRED)
find_package(rosidl_default_generators REQUIRED)
```

`rosidl_generate_interfaces()`

This function includes all of the messages for this package (in the `msg` folder) to be compiled. The function should look similar to the following:

In [ ]:

```
rosidl_generate_interfaces(${PROJECT_NAME}
  "msg/RoverEvents.msg"
   DEPENDENCIES std_msgs geometry_msgs
)
```

In summary, this is the minimum expression of what is needed for the `CMakeLists.txt` to generate a new `interface`:

CMakeLists.txt

In [ ]:

```
cmake_minimum_required(VERSION 3.8)
project(custom_interfaces)

if(CMAKE_COMPILER_IS_GNUCXX OR CMAKE_CXX_COMPILER_ID MATC
  add_compile_options(-Wall -Wextra -Wpedantic)
endif()

# find dependencies
find_package(ament_cmake REQUIRED)
find_package(rclcpp REQUIRED)
find_package(std_msgs REQUIRED)
find_package(geometry_msgs REQUIRED)
find_package(rosidl_default_generators REQUIRED)

if(BUILD_TESTING)
  find_package(ament_lint_auto REQUIRED)
  # the following line skips the linter which checks for
  # comment the line when a copyright and license is adde
  set(ament_cmake_copyright_FOUND TRUE)
  # the following line skips cpplint (only works in a git
  # comment the line when this package is in a git repo a
  # a copyright and license is added to all source files
  set(ament_cmake_cpplint_FOUND TRUE)
  ament_lint_auto_find_test_dependencies()
endif()

rosidl_generate_interfaces(${PROJECT_NAME}
  "msg/RoverEvents.msg"
   DEPENDENCIES std_msgs geometry_msgs
)

ament_package()
```

4. Modify the package.xml file:

Add the following lines to the `package.xml` file:

In [ ]:

```
<build_depend>rosidl_default_generators</build_depend>

<exec_depend>rosidl_default_runtime</exec_depend>

<member_of_group>rosidl_interface_packages</member_of_gro
```

This is the minimum expression of the `package.xml`.

package.xml

In [ ]:

```
<?xml version="1.0"?>
<?xml-model href="http://download.ros.org/schema/package_
<package format="3">
  <name>custom_interfaces</name>
  <version>0.0.0</version>
  <description>TODO: Package description</description>
  <maintainer email="ubuntu@todo.todo">ubuntu</maintainer
  <license>TODO: License declaration</license>

  <buildtool_depend>ament_cmake</buildtool_depend>

  <depend>rclcpp</depend>
  <depend>std_msgs</depend>
  <depend>geometry_msgs</depend>

  <build_depend>rosidl_default_generators</build_depend>
  <exec_depend>rosidl_default_runtime</exec_depend>
  <member_of_group>rosidl_interface_packages</member_of_g

  <test_depend>ament_lint_auto</test_depend>
  <test_depend>ament_lint_common</test_depend>

  <export>
    <build_type>ament_cmake</build_type>
  </export>

</package>
```

5. Compile

Now, compile the package to generate the new `interface`.

▶ Execute in Terminal #1

In [ ]:

```
cd ~/ros2_ws
```

In [ ]:

```
colcon build --packages-select custom_interfaces
```

In [ ]:

```
source install/setup.bash
```

- VERY IMPORTANT -

VERY IMPORTANT: When you compile new messages, there is an extra step before using the messages. You have to type in the Terminal, in the `ros2_ws`, the following command: `source install/setup.bash`.

This executes the bash file that sets, among other things, the newly generated messages created through the build process.

If you do not do this, it might give you an import error, saying it does not find the message generated.

- END -

To verify that your message has been created successfully, type into your Terminal: `ros2 interface show custom_interfaces/msg/Age`.

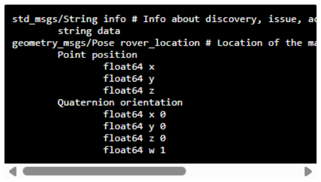
If the structure of the age message appears, it means that your message has been created successfully and is ready to be used in your ROS2 programs.

Execute in Terminal #1

In [ ]:

```
ros2 interface show custom_interfaces/msg/RoverEvents
```

Terminal #1 Output



```
std_msgs/String info # Info about discovery, issue, ac
        string data
geometry_msgs/Pose rover_location # Location of the ma
        Point position
                float64 x
                float64 y
                float64 z
        Quaternion orientation
                float64 x 0
                float64 y 0
                float64 z 0
                float64 w 1
```

REMEMBER: To Create a custom `interface`:

To create a new `interface`, you have to follow the next steps:

1. Create a new package or use any existing one.
2. Create a directory named `msg` inside your package
3. Inside this directory, create a file named `name_of_your_message.msg` (more information below)
4. Modify the `CMakeLists.txt` file (more information below)
5. Modify `package.xml` file (more information below)
6. Compile and source
7. Use it in your node

24/09/2024