ROS2: Setting up Workspaces and Packages

In this tutorial we're going to learn how to create a workspace and a package inside it. For more details on the same, read/follow the official ROS tutorial Creating a Workspace and Creating-Your-First-ROS2-Package

We shall just list out all the major tasks that one needs to perform while creating and workinmg with workspaces and packages in general.

Source ROS 2 environment

If this is not already included in the bashrc file (explained during installation) you'll need to source the ROS 2 environment everytime you open the terminal during this tutorial.

Depending on how you installed ROS 2 (from source or binaries), and which platform you're on, your exact source command will vary:

```
source /opt/ros/humble/setup.bash
```

Creating a workspace

Create a new directory For example:

```
mkdir -p ~/hb_task1b_ws/src
cd ~/hb_task1b_ws/src
```

What makes up a ROS 2 package?

- `package.xml` file containing meta information about the package
- `resource/<package_name>` marker file for the package
- `setup.cfg` is required when a package has executables, so ros2 run can find them
- `setup.py` containing instructions for how to install the package
- `<package_name>` - a **directory** with the same name as your package, used by ROS 2 tools to find your package, contains **init**.py

The simplest possible package may have a file structure that looks like:

```
my_package/
      package.xml
      resource/my_package
      setup.cfg
      setup.py
      my_package/
```

Packages in a workspace

A single workspace can contain as many packages as you want, each in their own folder. You can also have packages of different build types in one workspace (CMake, Python, etc.). You cannot have nested packages. Best practice is to have a src folder within your workspace, and to create your packages in there. This keeps the top level of the workspace “clean”. A trivial workspace might look like:

```
workspace_folder/
    src/
        py_package_1/
          package.xml
          resource/py_package_1
          setup.cfg
          setup.py
          py_package_1/

        py_package_2/
          package.xml
          resource/py_package_2
          setup.cfg
          setup.py
          py_package_2/
```

Create a package

Make sure you are in the src folder before running the package creation command.

```
cd ~/hb_task1b_ws/src
```

The command syntax for creating a new package in ROS 2 is:
For example:

```
ros2 pkg create --build-type ament_python hb_task_1b
```

Build a package

Putting packages in a workspace is especially valuable because you can build many packages at once by running colcon build in the workspace root. Otherwise, you would have to build each package individually.
Return to the root of your workspace:

```
cd ~/hb_task1b_ws
colcon build
```

To build only the my_package package next time, you can run:

```
colcon build --packages-select my_package
```

Source the setup file

To use your new package and executable, first open a new terminal and source your main ROS 2 installation. (as explained above)

Then, from inside the hb_task1b_ws directory, run the following command to source your workspace:

```
source install/local_setup.bash
cd ~/hb_task1b_ws/src/hb_task_1b/
```

Customize package.xml

You may have noticed in the return message after creating your package that the fields description and license contain TODO notes. That's because the package description and license declaration are not automatically set, but are required if you ever want to release your package.

From hb_task1b_ws/src/hb_task_1b, open package.xml using your preferred text editor and add these above test_depend :

```
<depend>xacro</depend>
<depend>robot_state_publisher</depend>
<depend>joint_state_publisher</depend>
<exec_depend>ros2launch</exec_depend>
<exec_depend>gazebo_ros_pkgs</exec_depend>
<depend>rclpy</depend>
```

setup the setup.py file

If you know what a CMakeLists.txt file is, well the setup.py is basically the same but for Python. When you compile your package it will tell what to install, where to install it, how to link dependencies, etc.

From hb_task1b_ws/src/hb_task_1b, open setup.py using your preferred text editor and add these in `data_files` :

```
data_files=[
('share/ament_index/resource_index/packages',
['resource/' + package_name]),
('share/' + package_name, ['package.xml']),
(os.path.join('share', package_name, 'launch'), glob('launch/*.launch.py')),
(os.path.join('share', package_name, 'urdf'), glob('urdf/*')),
(os.path.join('share', package_name, 'meshes'), glob('meshes/*')),
(os.path.join('share', package_name, 'config'), glob('config/*')),
(os.path.join('share', package_name, 'worlds'), glob('worlds/*'))
],
```

Finally build the package and source the setup file to use the package.

That should be sufficient to get you started on the tasks, but as mentioned earlier, it is good idea to go through the official ROS tutorial on Creating a Workspace and Creating-Your-First-ROS2-Package for a better understanding.

Infact we strongly recommend completing the beginner tutorials on docs.ros.org to get a good hold of ROS 2.
