ROS Actions Cheat Sheet

Create Action File

Example: ActionName.action

```
# Goal
geometry_msgs/Point my_goal
---
# Result
float32 my_result
---
# Feedback
float32 my_feedback
```

Modify package.xml

Include actionlib And actionlib_msgs Dependencies

```
  <build_depend>actionlib</build_depend>
  <build_depend>actionlib_msgs</build_depend>
  <exec_depend>actionlib</exec_depend>
  <exec_depend>actionlib_msgs</exec_depend>
```

Modify CMakeLists.txt

Include genmsg, actionlib_msgs, actionlib

```
find_package(catkin REQUIRED COMPONENTS
  roscpp
  rospy
  sensor_msgs
  std_msgs
  message_generation
  genmsg
  actionlib_msgs
  actionlib
)
```

Include Action Files In add_action_files()

```
## Generate actions in the 'action' folder
add_action_files(
  FILES
  ActionName.action
)
```

Include Generate Message Dependencies

```
## Generate added messages and services with any dependencies listed here
generate_messages(
  DEPENDENCIES
  sensor_msgs
  std_msgs
  actionlib_msgs
)
```

Create C++ Nodes

Import Actionlib Server, Client, And Custom Action

```
#include "pkg_name/ActionNameAction.h"
#include "actionlib/server/simple_action_server.h"
#include "actionlib/client/simple_action_client.h"
```

Create Action Server Typedef

```
typedef actionlib::SimpleActionServer<pkg_name::ActionNameAction> MyServer;
```

Create Action Server

```
MyServer server(node_handle_var, "action_topic_name",
                boost::bind(&myCallback, _1, &server), false);
server.start();

ros::spin();
```

Create Action Server Callback Function

```
void myCallback(const pkg_name::ActionNameGoalConstPtr& goal_point,
                MyServer* my_server)
{

  ros::Rate feedback_rate(2);
  pkg_name::ActionNameFeedback feedback_msg;
  pkg_name::ActionNameResult res_msg;
  float feedback_var

  while(condition)
  {
    feedback_var = doSomething(goal_point->point.x, goal_point->point.y);
    feedback_msg.my_feedback = feedback_var;
    my_server->publishFeedback(feedback_msg);
    feedback_rate.sleep();
  }

  res_msg.my_result = doSomething2();
  my_server->setSucceeded(res_msg);
}
```

Create Action Client Typedef

```
typedef actionlib::SimpleActionClient<pkg_name::ActionNameAction> MyClient;
```

Create Action Client

```
MyClient client("action_topic_name", true); // true -> don't need ros::spin()
client.waitForServer();

pkg_name::ActionNameGoal goal;
goal.point.x = 0;
goal.point.y = 0;
goal.point.z = 0;
// Dependant on goal msg type, in this example, geometry_msgs/Point

client.sendGoal(goal, &resultCallback, &activeCallback, &feedbackCallback);
client.waitForResult();
```

Create Action Client Callback Functions

```
void activeCallback()
{
  std::cout << "Action Activated" << std::endl;
}

void feedbackCallback(const pkg_name::ActionNameFeedback::ConstPtr& feedback_var)
{
  std::cout << feedback_var->my_feedback << std::endl;
  // Based off feedback attribute name in action file
}

void resultCallback(const actionlib::SimpleClientGoalState& state_var,
                    const pkg_name::ActionNameResult::ConstPtr& result_var)
{
  std::cout << result_var->my_result << std::endl;
  // Based off result attribute name in action file
}
```