function创建流程

1.从demo工程copy一个副本。重命名,这里我称new_func。

2.修改new_func目录下include和src里的文件名为new_func.h, new_func.cpp

3.修改package.xml中name和description标签为你想要的功能名，这里我依然叫它new_func,

将你自己的package.xml里配置的标签copy过来。

4.修改cmakelist.txt，修改里面的`project`为new_func，`set(INSTALL_DEVEL_PATH collect_node)` 这里把`collect_node`名字改成你这个func所属的node名，`add_definitions(-DFUNCTION_NAME="new_func")`，别的标签`add_library`，`target_link_libraries`，`set_target_properties`这些标签都改成自己，我这里是new_func，添加你自己代码里cmakelist.txt所需要的别的宏。

5.1修改new_func.h，将`namespace collect_node_ns`修改为你自己的命名空间，这里推介用节点名。

5.2重写`class Demo`，这里我改成class new_func。你在初始化中用到的pub，sub，client，server，timer都需要再这个private中定义。可参考代码中给出格式。

6.1，修改new_func.cpp，将`#include "demo.h"` 改成new_func.h,`将factory.registerCreater<collect_node_ns::Demo>`，这个改成你的命名空间下的类(5中重写的类new_func)。同样将`namespace collect_node_ns` 改成你的（5步骤中的）

6.2重写`Demo::Demo构造函数，将你原本初始化节点的逻辑放在这里。`（这里不允许有死循环，一律改造成定时器或者线程方式。）。`json_conf`入参可以拿到你在节点处配置的参数。

6.3补充你的析构函数，比如你的一些需要关闭文件描述符，new的空间等等。

7.将你自己定义的msg， srv等消息定义添加到hj_interface中。并将hj_interface做catkin_make编译，编译完后再执行一次catkin_make -DCATKIN_WHITELIST_PACKAGES="hj_interface" install_inc 。

所以这里你需要将你代码中用到的消息头文件和消息域名等都要改到hj_interface这个位置去。

8.在你的node目录（我这里用collect_node），修改相关平台的config.json，添加如下：

```
,
      {
        "ID": "new_func",
        "name": "new_func"
      }
```

9.执行catkin_make你的包，就能正常运行了