Регистрация агента

- **Заходим в** ~/ostis-example-app/problem-solver/cxx/exampleModule/keynodes

Открываем keynodes.cpp

```cpp
#include "keynodes.hpp"

namespace exampleModule
{
ScAddr Keynodes::question_find_subdividing;
ScAddr Keynodes::nrel_subdividing;
ScAddr Keynodes::question_find_isomorphic_structures;
ScAddr Keynodes::nrel_search_result;
ScAddr Keynodes::empty_set;

ScAddr Keynodes::question_silly_agent;  // <- request node
ScAddr Keynodes::nrel_kills_people;
ScAddr Keynodes::smoll_kitty;
```

Вставляем туда узел запроса нашему агенту(этот обязателен), и все остальные узлы, которые понадобятся нам для работы и формирования результатов нашего агента(по своему усмотрению).

Открываем keynodes.hpp

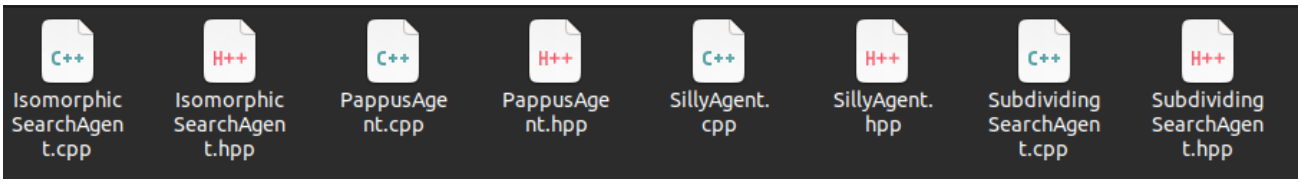
```cpp
public:
  SC_PROPERTY(Keynode("question_find_subdividing"), ForceCreate)
  static ScAddr question_find_subdividing;

  SC_PROPERTY(Keynode("nrel_subdividing"), ForceCreate)
  static ScAddr nrel_subdividing;

  SC_PROPERTY(Keynode("question_find_isomorphic_structures"), ForceCreate)
  static ScAddr question_find_isomorphic_structures;

  SC_PROPERTY(Keynode("nrel_search_result"), ForceCreate)
  static ScAddr nrel_search_result;

  SC_PROPERTY(Keynode("empty_set"), ForceCreate)
  static ScAddr empty_set;



  SC_PROPERTY(Keynode("question_silly_agent"), ForceCreate)
  static ScAddr question_silly_agent;;

  SC_PROPERTY(Keynode("nrel_kills_people"), ForceCreate)
  static ScAddr nrel_kills_people;

  SC_PROPERTY(Keynode("smoll_kitty"), ForceCreate)
  static ScAddr smoll_kitty;
```

Ручками записываем создание наших узлов как в примере

Заходим в ~/ostis-example-app/problem-solver/cxx/exampleModule/agents

Создаём непосредственно нашего агента

1) Создаем НашегоАгента.hpp и НашегоАгента.cpp (Вместо НашегоАгента название вашего агента(за тавтологию сори))



В НашегоАгента.hpp записываем такое содержимое(скопируйте):

```
#pragma once

#include <sc-memory/kpm/sc_agent.hpp>

#include "keynodes/keynodes.hpp"
#include "SillyAgent.generated.hpp"   // <- Замените на название вашего агента

namespace exampleModule
{

class SillyAgent : public ScAgent     // <- Замените на название вашего агента
{
  SC_CLASS(Agent, Event(Keynodes::question_find_pappus_graph, ScEvent::Type::AddOutputEdge))   // <- Замените на ваш узел запроса, тип события, на которое реагирует агент не меняем, оно у всех будет одинаковым
  SC_GENERATED_BODY()
};

}
```

В НашегоАгента.cpp записываем такое содержимое(скопируйте):

```
#include <sc-agents-common/utils/GenerationUtils.hpp>
#include <sc-agents-common/utils/AgentUtils.hpp>
#include <sc-agents-common/utils/IteratorUtils.hpp>

#include "SillyAgent.hpp"   // <- Замените на название вашего агента

using namespace std;
using namespace utils;

namespace exampleModule
{

SC_AGENT_IMPLEMENTATION(SillyAgent)  // <- Замените на название вашего агента
{
  SC_LOG_ERROR("Weee Woooo");
  return SC_RESULT_OK;
}

}
```

Заходим в ~/ostis-example-app/problem-solver/cxx/exampleModule

Открываем exampleModule.cpp
В подключенные библиотеки добавляем заголовочный файл нашего агента:

```
#include "exampleModule.hpp"
#include "keynodes/keynodes.hpp"
#include "agents/SubdividingSearchAgent.hpp"
#include "agents/IsomorphicSearchAgent.hpp"

#include "agents/SillyAgent.hpp"
```

Добавляем строчку с регистрацией нашего агента в код:

```
sc_result ExampleModule::InitializeImpl()
{
  if (!exampleModule::Keynodes::InitGlobal())
    return SC_RESULT_ERROR;

  SC_AGENT_REGISTER(SubdividingSearchAgent)
  SC_AGENT_REGISTER(IsomorphicSearchAgent)

  SC_AGENT_REGISTER(SillyAgent)
```

Добавляем строчку с отключением нашего агента после завершения работы платформы в код:

```
sc_result ExampleModule::ShutdownImpl()
{
  SC_AGENT_UNREGISTER(SubdividingSearchAgent)
  SC_AGENT_UNREGISTER(IsomorphicSearchAgent)

  SC_AGENT_UNREGISTER(SillyAgent)
```

exampleModule.hpp не трогаем

Впринципе, всё, наш агент зарегистрирован(на сдачу достаточно), нужно только перебилдить платформу.
Если же хотите, чтобы в вебе появилась кнопочка вызова вашего агента, то идём дальше

Заходим в ~/ostis-example-app/problem-solver/cxx/exampleModule/specifications
Создаем тут папку нашего агента(название желательно как у агента)
Создаем два файлика с расширением scs(на название все равно они все равно просто как в kb забилдятся)
В первый фалик записываем:

lib_component_ui_menu_silly_agent // <- Заменить silly_agent на что угодно, но запомнить, нам это ещё понадобится в следующем файле(коммент мой удалите иначе не заработает)
=> nrel_main_idtf:
[Компонент библиотеки. Команда пользовательского интерфейса для делания глустей(сюда что душе угодно вписываем, это просто описание для меню, что делает наш агент)]
(* <- lang_ru;; *);
[Library component. User interfaces command of doing silly things]
(* <- lang_en;; *);

<- library_of_platform_independent_reusable_components;
<- library_of_atomic_reusable_components;;

Во второй файлик записываем:

```
lib_component_ui_menu_silly_agent = [*
ui_menu_silly_agent <- ui_user_command_class_atom; ui_user_command_class_view_kb;
ui_one_argument_command_class;;

ui_menu_silly_agent
=> nrel_main_idtf:
        [Тут то, что будет на кнопке]
        (* <- lang_ru;; *);
=> nrel_idtf:
        [Запрос на глупости]
        (* <- lang_ru;; *);;

ui_menu_silly_agent
=> nrel_main_idtf:
        [?]
        (* <- lang_en;; *);
=> nrel_idtf:
        [Request?]
        (* <- lang_en;; *);;

ui_menu_silly_agent => ui_nrel_command_template:
        [*
                question_silly_agent _-> ._question_silly_agent_instance
                        (*
                                _-> ui_arg_1;;
                        *);;
                ._question_silly_agent_instance _<- question;;
        *];;

ui_menu_silly_agent => ui_nrel_command_lang_template: [А может это ты $ui_arg_1?   (Вопрос для вывода результата в качестве ответа)] (* <- lang_ru;; *);;
ui_menu_silly_agent => ui_nrel_command_lang_template: [$ui_arg_1?] (* <- lang_en;; *);;
*];;
```

Всё, что жёлтое заменяем на то, что вставили в первом файлике
Всё, что зелёное заменяем на название нашего узла запроса из keynodes

- **Заходим в** ~/ostis-example-app
  Открываем repo.path
  Дописываем путь к нашей папке с scs файлами

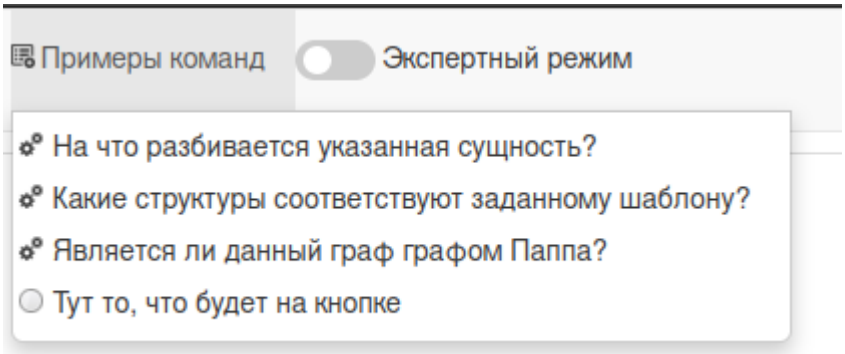
```
# specifications
problem-solver/cxx/exampleModule/specifications/agent_of_isomorphic_search
problem-solver/cxx/exampleModule/specifications/agent_of_subdividing_search
problem-solver/cxx/exampleModule/specifications/SillyAgent
```

- **Заходим в** ~/ostis-example-app/kb/ui_menu
  Открываем файлик ui_menu_na_example_commands.scs
  Добавляем там строчку с узлом из тех scs файликов
  P.S точка с запятой после последнего не ставятся

```
<= nrel_ui_commands_decomposition:
        {
                ui_menu_subdividing_search;
                ui_menu_isomorphic_search;
                ui_menu_pappus_graph_search;
                ui_menu_silly_agent
        };;
```

Теперь билдим вааще всё

Запускаем



Проверяем(пкм на любой узел и находим там название кнопки, либо пкм по любому названию→булавка, Примеры команд→Наш мегакрутой агент)

Нас отправит в бесконечную загрузку, но это потому что реализация нашего агента пустая, а веб слегка корявый.

В консоль ничего не выведется. Ну как бы всё.